**A project report submitted in partial fulfilment for the degree of**

**BSc (Hons) Software Engineering**

**School of Psychology and Computer Science**

**University of Central Lancashire**

Adventure Walk:   
The Gamification of Walking

Muhammad Katib Hussain

**20th April 2022**

# Abstract

My project attempts to encourage users of my project more aerobically active by gamifying features of an ordinary walk. Many people from time to time want to get out of their house and take a walk, either by themselves or with friends, others may want to reach a goal and others do it for exercise. Yet the hardest part for many is getting out of the house in the first place to take a walk; during the pandemic and especially during lockdown, many stayed at home and stopped going out altogether at the time. My hope is that with my project I could encourage more individuals to get out the house and work towards a goal with my walking app and its gamified features. To make sure that development of the project was both efficient and working out smoothly, I used techniques such as MoSCoW prioritisation to enumerate my objectives and their feasibility of making it into the final app as well differentiating the most critical of tasks from those that would be considered a luxury later in the project. This was used in conjunction with a Kanban board of which I used to keep track of what has been completed and what is yet to be worked on. As well as those, every other week I would also use Sprint Retrospectives to step back and evaluate what I have done so far and what I could improve on to ensure I am not divulging too much of my time in a problem that shouldn’t require it. The final product is a fully functional app written in Java whose core feature is choosing a location for you to walk towards and upon completion, the user would receive experience points and in-game currency which can be used to redeem rewards. The app would also give statistics on your walks along the week such as a step counter for those with an interest on their performance. Among all this, I used a thin-client architecture meaning my server application written in C# does the bulk of the processing to conserve the user’s battery. Considering my initial goal for the app was to have a functioning client-server architecture that made users walk from point A to point B, I would consider what I have accomplished an absolute success.

# Attestation

I understand the nature of plagiarism, and I am aware of the University’s policy on this.

I certify that this document reports original work by me during my University project. I also confirm that I adhere to the University’s legal and ethical guidelines for undergraduate projects in Computing.

Signature: Muhammad Katib Hussain

Date: 03/03/2022

# Acknowledgements

I’d like to my friends in the current year and in the years previous while I have attended University. Many times, I have doubted my performance and doubted my ability to be a good peer and friend, worried too much that what I have done is not enough and not worked hard enough but I have always been supported by peers and friends alike to keep pushing forward. Their unrelenting support is why I am here completing this project.

I’d like to also thank Matt Horton for being a great supervisor both personality and work wise. As well as Jonathan Edwards for being extremely supportive, well spoken and just in general chill person to talk to since I started university. I am also grateful to Nicholas Mitchell for being a superb lecturer who always tries to make the lectures a lot more fun with his wit and sarcasm, yet a great portion of my gratitude comes from knowing he tries his best with students given the little time he has to tend to them outside of labs and lectures.

From my family, I would like to thank my parents for being supportive of my career path I have taken and listening to my little successes in programming despite not understanding a single thing other than my sheer excitement of getting over an issue. Lastly, I would like to thank my brother who has helped me every now and then with programming despite it not being in his field and brainstorming solutions to numerous issues both to do with work in University and mental health.

# Table of Contents

The table of contents below is automatically generated from the paragraphs of style ‘Heading 1’ to ‘Heading 3’ and ‘Heading (Unnumbered)’. To update this after revisions, right-click in the table and choose ‘Update Field’ for the entire table. It is also possible to left-click the table and press F9 to update the entire table. Delete this paragraph and the one below before submission.

Sometimes Word does not automatically update references to other parts of the document, such as captions, Table of Contents and cross-references. To update everything, select the whole document by pressing Ctrl+A and then either press F9 or right-click anywhere and click on ‘Update Field’. Confirm all prompts by selecting ‘Update entire table’ and clicking OK. Sometimes the font changes on update so check it is the correct font face before submitting your work. You need to set it to Calibri.

[Abstract i](#_Toc101331498)

[Attestation ii](#_Toc101331499)

[Acknowledgements iii](#_Toc101331500)

[Table of Contents iv](#_Toc101331501)

[List of Figures ix](#_Toc101331502)

[List of Tables xi](#_Toc101331503)

[List of Listings xii](#_Toc101331504)

[1 Introduction 1](#_Toc101331505)

[1.1 Background And Context 1](#_Toc101331506)

[1.2 Scope and Objective 1](#_Toc101331507)

[1.3 Achievements 1](#_Toc101331508)

[1.4 Overview of Report 2](#_Toc101331509)

[2 Background and Related Work 3](#_Toc101331510)

[2.1 Introduction 3](#_Toc101331511)

[2.2 Gamification 3](#_Toc101331512)

[2.2.1 Effectiveness of Gamification 3](#_Toc101331513)

[2.2.2 Issues related to gamification of physical activity 4](#_Toc101331514)

[2.3 Thin-Client & Mobile Computing 6](#_Toc101331515)

[2.3.1 Thin-Client versus Thick-Client 6](#_Toc101331516)

[2.3.2 Thin-Client disadvantages with Mobile & Cloud Computing issues. 7](#_Toc101331517)

[2.4 User Experience Design 11](#_Toc101331518)

[2.4.1 Effectiveness of UX design methods in practice 13](#_Toc101331519)

[2.5 Summary 14](#_Toc101331520)

[3 Project Planning 15](#_Toc101331521)

[3.1 Introduction 15](#_Toc101331522)

[3.2 Methodology 15](#_Toc101331523)

[3.3 Requirements 16](#_Toc101331524)

[3.3.1 Must 16](#_Toc101331525)

[3.3.2 Should 16](#_Toc101331526)

[3.3.3 Could 17](#_Toc101331527)

[3.3.4 Would 17](#_Toc101331528)

[3.4 Development Decisions 17](#_Toc101331529)

[3.4.1 Android or iOS 17](#_Toc101331530)

[3.5 Tools and Techniques 18](#_Toc101331531)

[3.5.1 No-IP 18](#_Toc101331532)

[3.5.2 GitHub 18](#_Toc101331533)

[3.6 Legal, Social, and Ethical Issues 18](#_Toc101331534)

[3.6.1 Legal Issues 18](#_Toc101331535)

[3.6.2 Ethical Issues 19](#_Toc101331536)

[3.7 Kanban board for project management 19](#_Toc101331537)

[3.8 Summary 20](#_Toc101331538)

[4 Design 21](#_Toc101331539)

[4.1 Introduction 21](#_Toc101331540)

[4.2 System Design 21](#_Toc101331541)

[4.2.1 Model-View-Controller design pattern 21](#_Toc101331542)

[4.2.2 Database Design 23](#_Toc101331543)

[4.2.3 Client Application Design 24](#_Toc101331544)

[4.2.4 Server Application Design 25](#_Toc101331545)

[4.3 User Interface Design 26](#_Toc101331546)

[4.3.1 Maintaining consistency 26](#_Toc101331547)

[4.3.2 Showing lots of information succinctly and elegantly 27](#_Toc101331548)

[4.3.3 Keeping the UI simple 28](#_Toc101331549)

[4.3.4 UI Error Handling and packages in Android Studio 29](#_Toc101331550)

[4.4 Summary 30](#_Toc101331551)

[5 Implementation 31](#_Toc101331552)

[5.1 Introduction 31](#_Toc101331553)

[5.2 Server Application 31](#_Toc101331554)

[5.2.1 Controller and data manipulation 31](#_Toc101331555)

[5.2.2 Quest System 32](#_Toc101331556)

[5.2.3 Reward System 36](#_Toc101331557)

[5.2.4 Login System 37](#_Toc101331558)

[5.3 Client Application 40](#_Toc101331559)

[5.3.1 Using Retrofit to communicate with my web API 40](#_Toc101331560)

[5.3.2 Login and Sign-up Screen 40](#_Toc101331561)

[5.3.3 Quest/Main Screen 42](#_Toc101331562)

[5.3.4 Account Screen 44](#_Toc101331563)

[5.4 Summary 46](#_Toc101331564)

[6 Investigation 48](#_Toc101331565)

[6.1 Introduction 48](#_Toc101331566)

[6.2 Functional Testing 48](#_Toc101331567)

[6.3 Non-Functional Testing 50](#_Toc101331568)

[6.4 User Testing 51](#_Toc101331569)

[6.5 Summary 52](#_Toc101331570)

[7 Evaluation, Conclusions and Future Work 53](#_Toc101331571)

[7.1 Project Objectives 53](#_Toc101331572)

[7.2 Self-Evaluation 53](#_Toc101331573)

[7.3 Project Evaluation 54](#_Toc101331574)

[7.4 Applicability of Findings to the Commercial World 55](#_Toc101331575)

[7.5 Conclusions 56](#_Toc101331576)

[7.6 Future Work 56](#_Toc101331577)

[References 57](#_Toc101331578)

[Appendix 1 61](#_Toc101331579)

[Project Proposal 61](#_Toc101331580)

[Project Context 61](#_Toc101331581)

[Specific Objectives 61](#_Toc101331582)

[Potential Ethical or Legal Issues 62](#_Toc101331583)

[Resources 62](#_Toc101331584)

[Potential Commercial Considerations - Estimated costs and benefits 62](#_Toc101331585)

[Proposed Approach 63](#_Toc101331586)

[References 63](#_Toc101331587)

[Appendix 2 65](#_Toc101331588)

[Technical Plan 65](#_Toc101331589)

[Title 65](#_Toc101331590)

[Summary 65](#_Toc101331591)

[Deliverables 65](#_Toc101331592)

[Constraints 65](#_Toc101331593)

[Key Problems 66](#_Toc101331594)

[System and Work Outline 66](#_Toc101331595)

[Project Activities 67](#_Toc101331596)

[Risk Analysis 67](#_Toc101331597)

[Risk 67](#_Toc101331598)

[Severity 67](#_Toc101331599)

[Likelihood 67](#_Toc101331600)

[Action 67](#_Toc101331601)

[Options 68](#_Toc101331602)

[Potential Ethical or Legal Issues 69](#_Toc101331603)

[Commercial Analysis 69](#_Toc101331604)

[Factor name 70](#_Toc101331605)

[Description 70](#_Toc101331606)

[Is this a cost 70](#_Toc101331607)

[or a benefit 70](#_Toc101331608)

[Estimated 70](#_Toc101331609)

[Amount 70](#_Toc101331610)

[Estimate 70](#_Toc101331611)

[of when 70](#_Toc101331612)

[paid 70](#_Toc101331613)

[Employability Contribution 71](#_Toc101331614)

[References 71](#_Toc101331615)

[Appendix 3 76](#_Toc101331616)

[Project Trello Board 76](#_Toc101331617)

[Appendix 4 78](#_Toc101331618)

[Overview of the prototype 78](#_Toc101331619)

[Appendix 5 79](#_Toc101331620)

[Interactable prototype 79](#_Toc101331621)

[Appendix 6 80](#_Toc101331622)

[All endpoints on the server listed in their entirety 80](#_Toc101331623)

[Appendix 7 82](#_Toc101331624)

[Client application’s full architecture illustrated in a Class Diagram 82](#_Toc101331625)

[Appendix 8 83](#_Toc101331626)

[Functional tests using Postman on my server 83](#_Toc101331627)

[Appendix 9 84](#_Toc101331628)

[Trello page updates with a dedicated bug testing section 84](#_Toc101331629)

# List of Figures

[Figure 1 - Diagram of the Fogg Behaviour Model which has three factors: motivation, ability (simplicity) and prompts which contribute to the occurrence of an action (behaviour) (Fogg, 2009). 5](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331630)

[Figure 2 - Diagram of client-server model showing interaction between clients and a server (Client–Server Model - Wikipedia, 2011). 6](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331631)

[Figure 3 - Diagram of client-server model showing interaction between clients and a server using thick-client architecture (Haroon-Sulyman & Shakirat Oluwatosin, 2014). 7](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331632)

[Figure 4 – Diagram of different network appliances, showing the bandwidth throughput they require along with the amount of latency they operate with (Xu, 2017). 8](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331633)

[Figure 5 - Diagram of the amount of cell phone related injuries between years 2004 to 2010. (Nasar & Troyer, 2013). 11](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331634)

[Figure 6 - Diagram of the varying age-groups that incidents related to cell phone usage had occurred to between the years 2004 and 2010 cumulatively (Nasar & Troyer, 2013). 11](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331635)

[Figure 7 - Low-fidelity paper prototype of the app I plan to create with planned potential features and basic layout defined. 12](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331636)

[Figure 8 – Example of using Trello to manage my project timeline. 20](#_Toc101331637)

[Figure 9 – Illustation of the Model-View-Controller design pattern (Rhoudunda, 2020) 22](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331638)

[Figure 10 – Example file structure within server application 22](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331639)

[Figure 11- Example of file structure within client application 23](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331640)

[Figure 12 - ER Diagram of database design so far 23](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331641)

[Figure 13 – Activity Diagram displaying the process of logging a user into the application. 25](#_Toc101331642)

[Figure 14 – Class Diagram showing the proposed server architecture (prone to changes depending on future implementation) 26](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331643)

[Figure 15 - Back button at the top left of each of my sub-screens 27](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331644)

[Figure 16 - Depiction of graph (Accounts screen), friends list (Friends screen) and global leader-board (Leader-board screen) from high-fidelity prototype 27](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331645)

[Figure 17 - Demonstration of the equivalence between the main feature on the main screen and the logo image 29](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331646)

[Figure 18 - Separate states of the main screen exemplifying the difference between having the navigation drawer open and hidden during gameplay 29](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331647)

[Figure 19 – Android SDK showing server error and Material Design showing user error respectively. 30](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331648)

[Figure 20 – Illustration of concepts used and end result of getting a quest location 33](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331649)

[Figure 21 – Depiction of the Mercator projection and its distortion pattern across the entire map. 33](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331650)

[Figure 22 – Design of the login screen shown on the left and sign -up screen on the right. 41](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331651)

[Figure 23 – Validation on each field appropriate to the meaning of each field in the sign-up screen. 41](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331652)

[Figure 24 – Variety of login error messages that exist for specific situations. 42](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331653)

[Figure 25 – Screen shown upon logging into the app. 42](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331654)

[Figure 26 – Process of starting a quest and showing the UI change when a quest has been started. 43](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331655)

[Figure 27 – Notification upon completion of a quest 43](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331656)

[Figure 28 – Full view of the user interface within the Account screen 44](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331657)

[Figure 29 – Current week data represented on all graphical options within the Account screen. 46](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331658)

[Figure 30 – Average steps from all 52 weeks on all graphical options within the Account screen to show handling of larger sets of data. 46](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331659)

[Figure 31 – A list of parameters associated with non-functional testing presented by Hamilton (2022) 50](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331660)

[Figure 32 – Snippet of a bug card with user comments on steps taken surmised with progress shown in the comments of the card. 52](file:///E:\Desktop\Third%20Year%20Material\CO3808%20Project%20Module\ProjectReport%20-%20Muhammad%20Katib%20Hussain.docx#_Toc101331661)

# List of Tables

[Table 1– Amount of studies using particular motivational affordances (Hamari et al., 2014) 4](#_Toc101331662)

[Table 2 – Snippet of the list of data within the UserActivities table. 39](#_Toc101331663)

[Table 3 – Snippet of rows dedicated to each setting within the UserSettings table. 39](#_Toc101331664)

[Table 4 – Sample conducted Sign-up screen tests from an earlier iteration of the project 48](#_Toc101331665)

[Table 5 – List of requirements for the project and their completion 54](#_Toc101331666)

# List of Listings

[Listing 1 - [QuestController.cs] Endpoint to get a new quest location (C#) 31](#_Toc100896038)

[Listing 2 – [QuestRandomiser.cs] Segment of method showing how a quest location is set (C#) 32](#_Toc100896039)

[Listing 3 – [QuestController.cs] Most critical code in the endpoint for checking the completion of a quest (C#) 34](#_Toc100896040)

[Listing 4 – Implementation of method that calculates distances between two points. 35](#_Toc100896041)

[Listing 5 – [EarnableCalculator.cs] A view of the properties and methods associated with earnable types within the game. 37](#_Toc100896042)

[Listing 6 – [ILoginApi.java] Setup for a request to the server attempting to log in a user (Java). 40](#_Toc100896043)

[Listing 7 – [LoginExecutor.java] Sending the request to the server with the login details as the parameter (Java). 40](#_Toc100896044)

[Listing 8 – [LoginExecutor.java] Class inheriting from a deprecated object before refactoring (Java) 49](#_Toc100896045)

[Listing 9 – [LoginExecutor.java] Class inheriting from refactored object to satisfy reusability and reliability tests (Java) 49](#_Toc100896046)

# Introduction

## Background And Context

In the recent decades, video games have become very mainstream. From the release of the very first Nintendo consoles and arcades, this new form of digital interaction quickly took over the world within just a few decades since the 1970’s and 1980’s. With an estimated 3.24 billion video game players worldwide in 2021 (Statista, 2021), this form of entertainment seems to only attract more attention each passing year. Unfortunately, though, a vast majority of the global populace does not have enough time to play games each day and have varying responsibilities in the outside world. My project aims to provide a functioning app that combines the activities in the outside world with subtle aspects of everyday games and in effect, merge the two for the benefit of the user.

## Scope and Objective

The project I am undertaking is extremely scalable and while there are many ideas I have for features to include within this project, I realised there is simply not enough time to have a satisfactory implementation for all of them. Disregarding features, this projects architecture dictates the performance of the final app and server. Without a good backbone, all the features would be exponentially harder to implement and even more so time-consuming. That said a list of my objectives are as follows:

* Creating a mobile application by which a user can get an objective to walk to a google maps implementation and receive rewards for completion
* Creating a server to handle heavy processor workloads for the thin-client mobile application
* A server which connects to a live database to perform CRUD operations in real-time.
* Present statistics to users of their activity.

## Achievements

At present, I was successful in creating a stable mobile application which achieved the original objectives and then some. The final app allowed users, to sign up, (encrypting their credentials using BCrypt as a password-hashing function) and sign in. It allowed a user to set a quest (an objective) on an implementation of Google Maps and walk to a destination which upon completion provided experience points and currency. The user has a step counter recording their steps given they have an appropriate step-counter sensor within their device to record steps shown in various charts. Finally, the user could tailor their experience by customising the distances they would like to walk and would accordingly be awarded for the distance they managed to cover to get to the objective.

As for the server implementation, that was also successful. Most of the processor heavy tasks are computed on the server. Alongside that, all database calls are made by the server and not by the mobile application. Database calls were made in an object-oriented fashion which made connecting to the database and sending queries considerably easier thanks to Entity Framework Core (EFCore) which is a Microsoft owned package used specifically for querying a database with ease. On top of that, I managed to implement the RESTful architecture into my server application which will help in scalability of my project long term.

## Overview of Report

**Chapter 2:** Contains a reviewal of relevant literature to my project including the thin-client architecture, gamification, user experience considerations and other topics related to computing in general.

**Chapter 3:** Explains my choice of methodology used to develop the project as well as specifying the requirements set during the planning of my project, ordered by a prioritisation technique to specify the importance of completing each task. It also describes the type of mobile operating system I plan to develop for, what tools I will use to assist my development and legal and ethical issues surrounding the scope and intention of the project.

**Chapter 4:** Discusses how guidelines linked associated with human computer interactions have influenced my design decisions as well as showing a portfolio of my planned designs that my implementation will embody by using a high-fidelity prototype. Specific system designs are also mentioned that are not focused on front-end UI such as a Entity Relationship Diagram to show the associations between tables in my database as well as go into detail about using design patterns within my system architecture to facilitate an easier development experience,

**Chapter 5:** Presents the major and minor implementation details within both the server and client applications while also discussing the use of geometry and mathematics within my implementation with justifications on design decisions carried out. It sequentially narrates the steps taken to ensure the implementation of the chosen system architecture pattern works as intended and describes how the server and the application communicate and interpret the data that they both receive and send between each other.

**Chapter 6:** Details the importance of testing and how it has been carried out during and after development of my project using real world testing by myself and the target demographic. It also notes the variety of tests in place to test my project, both functionally and non-functionally with examples of how the testing strategies used has affected the material presented in earlier chapters.

**Chapter 7:** Displays the progress made by me working on the project and consequently, each of the initial requirements I set in Chapter 3. It goes into further details into an honest reflection of my own self-improvement ever since the undertaking of the project and the mountain of knowledge procured following Android development for the client application and C# development for the server-side application. Finally, it provides an indication of improvements that could me made when working on the project in the future, post-university.

# Background and Related Work

## Introduction

My project’s purpose is to have the user engage in physical activity while having fun making their efforts count toward a goal provided by the mobile application that my project provides. Physical activity for some takes a lot of effort, and discipline to take time out for something they see as a boring task. That is why I will gamify their physical activity, to make sure they are engaged consistently with the app while also going about their task. It is also very important that the app is not only engaging, but also simple and easy to use; this is the responsibility of user interface design and is one of the key aspects to making the application itself successful. However, while both the design and the game are key to the project, neither are going to be appealing enough to the user if the app itself is slow or a battery drainer. That’s why I am going for a thin-client architecture, to make sure most the resource-intensive tasks are instead delegated to a central server in which will send a response back to their device with a result.

## Gamification

### Effectiveness of Gamification

In the last decade, gamification has become an extremely popular way of improving user engagement with a service as well as the positive patterns associated with the service in ways such as socialising with the service or increasing overall user activity (Hamari et al., 2014). I plan to use gamification to my advantage on my app for those reasons, however the effectiveness has shown to vary depending on the use case. In the case of Eickhoff et al. (2012), their research showed a substantial positive impact in user motivation and efficiency when given an incentive such as money or entertainment to workers. Due to the gamified methods, they also massively improved cost effectiveness within their study. Whereas in the case of Denny (2013), upon testing the effect of virtual achievements on student engagement there was a positive impact when students were answering questions but not when they were authoring questions despite incentives (badges) being present in both. Denny (2013) theorised that answering questions is an activity students find immediate value in seeing as they are more familiar with answering questions than authoring them and that designers should probably cater towards rewards that their target audience already see value in for the best impact. Regardless, both studies have found only positive impacts in user engagement and at worst, no effect with their gamified elements. That is not to say the gamified elements alone will have a positive impact on any product, their design and implementation are key to making them effective and purposeful to the product. My app will primarily fixate on activities involving walking and therefore in accordance with the idea Denny (2013) proposed, I will prioritise creating randomly generated locations to walk to within their area rather than the user having to set their own location to walk to, incentives could be made for both, but I believe doing the journey has more immediate value than configuring the journey for my target audience.

There are many ways of gamifying a product, as shown below in Table 1. Some motivational affordances are used a lot more frequently than others. The most common from the table below are core concepts that add game aspects to a product, but each affordance carries a degree of importance to each use case. An example of this is in 2013, the app “Flappy Bird” simply had a point system as its motivation affordance, while another game such as “Minecraft” used achievements as its motivational affordance, but both became extremely popular. This was not due to these games just having these motivational affordances, but their implementations of it is what made them popular. I plan to add a point, achievement, and a challenge system, however for my case, a leader board system is quite a hard choice as I must consider long term effects.

Table 1– Amount of studies using particular motivational affordances (Hamari et al., 2014)

|  |  |
| --- | --- |
| **Affordance** | **Number included in studies** |
| Points | 9 |
| Leaderboards | 10 |
| Achievements/Badges | 9 |
| Levels | 6 |
| Story/Theme | 6 |
| Clear goals | 4 |
| Feedback | 6 |
| Rewards | 4 |
| Progress | 4 |
| Challenge | 7 |

### Issues related to gamification of physical activity

I wish to make this game playable with friends and therefore that means having a social system and to add to that, a global leader board of sorts to compare themselves with others on distances walked or steps done within a day or week. Introducing competition into the app can lead to competition anxiety, which can be seen as a negative correlation between anxiety and achievement motivation (Khan et al., 2011). In effect, that means an individual with a high-level of anxiety also means that individual will have low level of motivation to work towards an achievement. Competition can enhance or worsen the user experience and its implementation within the app can cause positive or negative effects to the user depending on their personality (Epstein & Harackiewicz, 2016). This would be unidealistic as those in the user-base that are athletic and enjoy being on the move would inevitably rank higher on the leader board than those who just want to go for a walk every now and then. For others with a higher motivation to achieve, this feature may inspire them to surpass another individual on the leader board as a personal goal. Though I believe that overall, for my app, the minority of users will look at the global leader board and feel inspired and the majority would not. I feel as if this feature would bring little value to the app if it were just a global leader board as achievements of a few that are dedicated to the app will overshadow the achievements of most individuals. This would result in a reduced usage of the feature as the user has no incentive to check the results on the leader board other than for curiosity’s sake. Therefore, if I were to add a leader board along with the social system, I will add 2 separate leader boards. There will be one for global users and one for friends. This will hopefully result in a more realistic comparison between one individual and another that they relate to while also displaying global stats for those that are curious.

Diagram

Description automatically generatedAnother issue to consider is whether the app will motivate people enough to do what the app asks them to do. The app is centralised over physical activity, but for the gamification to work, motivational and simplicity factors must be considered. Physical activity tends to be much more of a chore than just playing games, so careful design is required for the gamification to become successful. Below in Figure 1, the Fogg Behaviour Model demonstrates that for an individual to perform an action, they must be adequately motivated, have the ability to do the task (based on simplicity) and be prompted to perform the behaviour, all at the same time (Fogg, 2009). This has been an effective approach for persuasive design along with gamification as seen in case of O’donovan et al. (2013), where they used the Fogg Behaviour Model to carefully design their gamification of a university course. They introduced an experience system, a secret group with a storyline, clues, as well as justifiable rewards to go with it and various other game elements to gamify the course while following the model shown in Figure 1. From their results they deemed that students significantly improved on their understanding of the course and their engagement with the course as well as students having better lecture attendance and course grades overall. If the model is used correctly, this would lead to a higher chance of the gamification of a product to become successful. In my case though, physical activity is much more optional than a university course. There is not as much incentive to do physical activity that you can do anytime, unlike with a university course where you pay a lot of money to get a degree within a period of a few years. The individual must be committed to do the physical activity and to maximise their engagement within the app, I will utilise this model to determine whether some features are worth adding in, which features should be prioritised and fine-tune each feature based on their prompts, the motivation needed, and the task’s simplicity.

Figure 1 - Diagram of the Fogg Behaviour Model which has three factors: motivation, ability (simplicity) and prompts which contribute to the occurrence of an action (behaviour) (Fogg, 2009).

## Thin-Client & Mobile Computing

### Thin-Client versus Thick-Client

Graphical user interface, application

Description automatically generatedThe thin-client/server computing model involves having a thin-client software or hardware device that communicates to a server over a network protocol. This server has the ability manage and process calls made to the server from the remote software/hardware (Kanter, 1997), resulting in a two-way communication between client and server. This is essentially how the internet works. This model is used often in the current day by many apps and websites, but examples of the concept aren’t limited to the digital world. For instance, a doctor could be the server with the client being the patient; the patient wants results from their diagnosis that the doctor made, which is the same as a client asking for data back from the server. Below in Figure 2, the client-server model is depicted, which also depicts thin-client computing where the devices on the left communicate with the server over the internet to get data.

Figure 2 - Diagram of client-server model showing interaction between clients and a server (Client–Server Model - Wikipedia, 2011).

A thick-client (or 2-tier client-server architecture) in contrast, is a client that typically has many to most of its functions remain usable independent of the server, therefore it needs no intermediary to receive data from a database. Some of these functions would run a portion of the coding and business logic that would otherwise have been ran on the server if it were a thin client (Haroon-Sulyman & Shakirat Oluwatosin, 2014). As shown in Figure 3, the client doesn’t necessarily require a server, but it still needs a separate database that it needs to communicate with.

Diagram

Description automatically generatedRelating back to my app, the features I plan to add require some heavy computation which can take varying amounts of time. To add to that, social systems, and features such as the leader board would not be feasible on a thick-client based architecture as not only would that make the implementation tedious, but it would have many security risks too. Following that, I have decided that I will utilise the thin-client architecture as opposed to a thick-client. A thick-client is resource-intensive and for mobile applications, this will drain more battery, reduce the performance of the app and as a result, worsen the user experience. Seeing as my app will be developed for Android devices and is particularly targeting mobile devices, hardware limitations is an element I must consider when designing my app. An advantage of thin-client computing over thick-client is that it reduces the total cost of ownership (Tang et al., 2011) which can be defined as getting as much as you can out of your investment in the technology without additional effort (Kanter, 1997). In my case, this means that any improvements I make to my app that could either make my app more computationally complex or simple, I would only need to worry about my server’s hardware as its hardware is what directly affects the performance of the app for each user. Since all the resource-intensive tasks would be delegated to the server, performance of the server needs to be optimal as a minimum to attain the best user experience on the client-side. The user’s hardware is of little concern when making changes for the app on the server-side, but *only* for server-side. The user’s device is only a concern when debating the minimum specifications for their device to run the app without issue or when changes are made for the client-side that has potential to impact performance. For a thick-client though, you would always need to be concerned with how it would impact performance as there really isn’t a server to compute the resource intensive tasks. In summary, thin-clients would cause less worry and provide more freedom to me as a developer to produce features, while the users experience few to zero issues with app on the client-side.

Figure 3 - Diagram of client-server model showing interaction between clients and a server using thick-client architecture (Haroon-Sulyman & Shakirat Oluwatosin, 2014).

### Thin-Client disadvantages with Mobile & Cloud Computing issues.

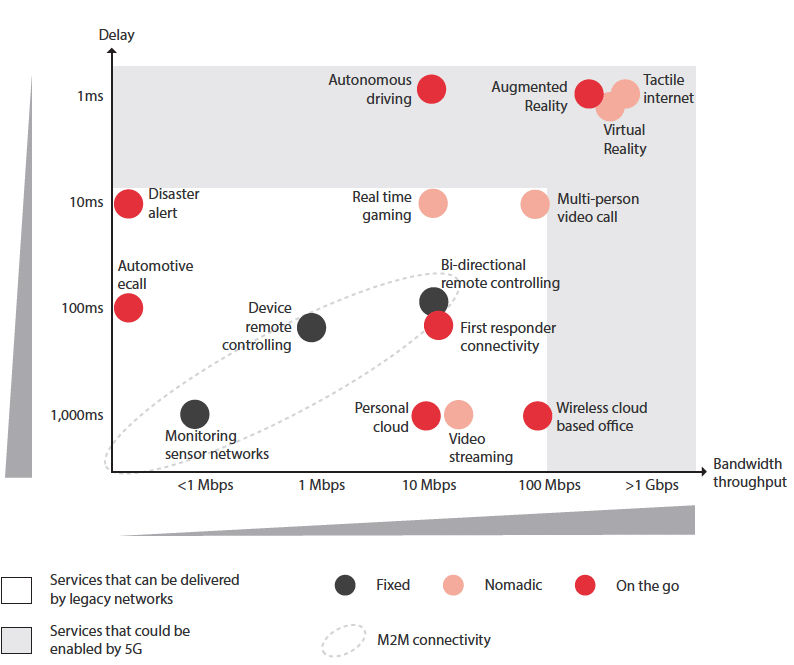
Even though the thin-client architecture seems far better than the thick client architecture in my situation, it also comes with its downsides. As a result of following the client-server model shown in Figure 2, the user requires an internet connection, and a relatively decent one at that. Figure 4 shows the varying amount of bandwidth is required for different network applications and along with that, each application has a different amount of latency attributed to it. In my case, the client side will be doing routine calls to the server for data related to each feature, the feature that will take the most bandwidth is the implementation of google maps on the client-side. The ideal user would have a have a good internet provider, with a decent amount of mobile data on their device. Considering that, I cannot design my app around what is just ideal, I must find ways to make the client-side as efficient as possible to lower the criterium for the minimum specifications needed to be able to use the app. Uptime of the server is also just as important, without the server, the app is useless. I need to make sure that the server is always connected to the internet with minimal interruptions. For this, I could use a backup server in a remote location, using another internet service provider (ISP). This would be for when I know the main server’s ISP’s network will go down or when maintenance is needed. Realistically though, for the project, I know there won’t be many user’s using the app currently in the near future. Downtime is allowable in this scenario as I do not receive any monetary compensation for the project, but if I were to undertake the project as a commercial product, I would make sure to at least have a backup server, invest in remotely hosted database from trusted providers and use the full utility of cloud services such as Microsoft Azure.

Figure 4 – Diagram of different network appliances, showing the bandwidth throughput they require along with the amount of latency they operate with (Xu, 2017).

On the topic of cloud computing, as useful as it is, it also introduces additional security risks as opposed to an offline application. Some of these relate to the applications themselves, for instance, SQL Injection is a risk that involves an attacker executing SQL code in an erroneous or malicious way to retrieve data (Grobauer et al., 2011) they shouldn’t possess. Others are risks outside of the application, Popovic & Hocenski (2010) address 10 different security concerns related to cloud computing security outside its application. They discuss issues with data-integrity, where you lose physical security as your information is sent to one company, but they send it to many third parties of which the user has no idea what they will do with their information, let alone know who has it. They also touch on handling payment security, insufficient auditing, violating privacy rights and more. My app will be dealing with location data from the user as well private information such as the user’s username and password to log in for the social system. This is very sensitive information I am dealing with of which can get me into serious trouble if it is able to be accessed by others other than me for the purposes of the app; it is my top priority to ensure that user data remains secure. For this reason, I plan to add validation, follow standards, guidelines, and conventions, add unit tests, thoroughly test my app, and prototype features before they are released publicly. Since the server will be locally hosted as well as the database (if I choose not to go with a trusted provider to reduce expenses), only I would know where the data is held from a physical standpoint. Following the above steps, this should dramatically increase the chances that the data stored will not be stolen or tampered with in any way. I cannot say for certain that the data will remain secure forever as cloud computing is still evolving and new security risks may occur in the future that I will need to address but for the time being, I will attempt to do what I can to ensure that data I retrieve is stored safely and securely.

While cloud computing is related to my project, it integral to the project as mobile computing which is the concept I founded the project on. Mobile computing can be defined as the use of technology to transfer data using a computer with needing to have a physical link to the recipient (Pradeep et al., 2012). It is similar to cloud computing in that regard, but cloud computing does not have a single absolute definition. Some believe it to just a system that stores data remotely and uses a client to fetch data, others believe it to be a set of distributed systems that are linked between each other that service providers utilise to provide clients with service according to their service-level agreements. (Qi & Gani, 2012). The element that makes mobile computing unique is that it deals with mobile devices such as smartphones or tablets and make use of the internet, software on these devices and communication systems that these devices can utilise to communicate with other devices connected on the cloud.

Mobile computing isn’t perfect though, it is still a relatively new concept and can still evolve. From my research, there aren’t many other concepts that tackle the issues currently presented by mobile computing, so unfortunately I must make do with the good that mobile computing provides and work around the limitations of it. It is important to note that not all issues discussed are explicitly related to technology, some are related to real-life, physical circumstances. Firstly, one of the basic features my app will provide is to prompt the user to walk from one location to another. Depending on the terrain and weather local to the user, reception may be an issue as when walking in tunnels or between buildings, walking in heavy rain or living on the countryside. These all can reduce the users mobile phone reception (Pradeep et al., 2012), causing them to possibly walk past the objective because of not receiving a signal that they had already reached the objective. Unfortunately, there isn’t much to help this issue, the best thing that can be done is for a user to be closer radio towers to receive information. Security is also a large issue and shares many issues related to cloud computing. The user’s data must be confidential, no unauthorised user should access another user’s data. An authorised user is required to have access to their data at any time. An unauthorised user should not have the ability to modify, create or destroy any data to maintain integrity (Pradeep et al., 2012). These principles have been given the acronym “CIA” standing for confidentiality, availability, and integrity. They are of grave import when designing systems that utilise the mobile computing concept and are the foundations for having a good quality of service while also putting the security of each user’s data as a top priority when designing a system. That is why I will carefully plan my database model and structure using UML when I am adding features and make use of stored procedures rather than SQL queries in code to ensure unauthorised users cannot manufacture their own queries an manipulate, destroy, or take data. I may also implement authorisation tokens which will add extra security for each call made to the server and validate that it comes from an authorised user.

Lastly with mobile computing, the largest real-life issue is the potential for my app to cause harm to the user. People may cheat with the app and get to locations by vehicle instead of doing physical activity but even those who choose to be physically active can are in danger of injuring themselves while using the app. Nasar & Troyer (2013) discovered that between the years of 2004 and 2010, there has only been an increase in injuries year-by-year related to individuals using their cell phones. Figure 6 visualises the increase of injuries over these years with Figure 5 representing the age groups of the cumulative number of injuries across the years 2004 and 2010. In the current year, I can only imagine that injuries related to cell phone uses have only increased. Considering that I am encouraging physical activities with my app, I believe that the age-groups that would most likely use my app are people under the age of 30, which in Figure 5 shows that they are the most impacted age groups when it comes to injuries related to cell phone usage from 2004 to 2010. This is one of those issues that I cannot do much about, the user themselves must be aware of their surrounding and reduce the chances of harm to themselves. From a developer perspective, the only thing I can really do is ensure that the app doesn’t have addictive features that users keep their focus on and that they the users remain engaged with the app but also stay aware of dangers around them. I will make sure that once they attempt a journey, the UI will remain static with minimal updates so that they can focus on their activity instead of focusing on the app. Only focusing on the Chart

Description automatically generatedapp once they have completed their activity.Graphical user interface, chart

Description automatically generated with medium confidence

Figure 5 - Diagram of the amount of cell phone related injuries between years 2004 to 2010. (Nasar & Troyer, 2013).

## User Experience Design

Figure 6 - Diagram of the varying age-groups that incidents related to cell phone usage had occurred to between the years 2004 and 2010 cumulatively (Nasar & Troyer, 2013).

User experience (UX) design is a process used to find what the user experience will be like when users use your app. It is the bigger picture of user interface design by incorporating the user into the design process. Specifically this process is empathises with the user base, make abstractions about their feelings, produce ideas from these feeling and following that up by prototyping and testing the product (Babich, 2020). If I only catered for the user interface rather than the user experience, I could have a beautiful looking app but a horrible user experience; where despite the app’s interface being well designed, users would not stick around on the app long or be frustrated with the amount of effort put into the interface but not into understanding what a user wants to see or do in the app.

Diagram

Description automatically generatedUX design is an iterative process, each iteration attempts to improve the product and its usability. Babich (2020) has summarised this process quite elegantly in his article about starting with the basics of UX design. It starts with finding the product definition via interviews with stakeholders and using concept sketching with as a base. From there product research is done, this involves researching the market for ideas gathered that have been implemented before on other projects with variable success and user research to find out the target audience. After that, analysis takes place which has a variety of methods to find out what users want and why they want it. An example of one of these methods involve user personas, they are fictional but are used represented the different types of users that form your user-base, these are used as a reference when designing the product to verify you are creating the correct product for your users. After analysing the users needs, the process of designing the system beings. This involves creating prototypes and wireframes, upon which the implementation follows through. Finally, the product is tested, surveys are done by potentials users of the system and analytics are formed from the product usage which shows patterns which may or may not be favourable to the business. In that case, the product can be reiterated over and improved on. At this moment, I have completed some analysis for my userbase and I have created a low-fidelity paper prototype for my app, the left-most image in Figure 7 depicts the planned main screen with a quest objective and your location on a map with a menu button, the middle image showing the menu’s options and features and the right-most image being an example of what the profile screen may look like, with every other screen following the same principle as the profile screen. There is much to improve on and consider changing with the current sketch but most the issues with the design will hopefully be resolved further on with each iteration of the design.

Figure 7 - Low-fidelity paper prototype of the app I plan to create with planned potential features and basic layout defined.

### Effectiveness of UX design methods in practice

Linking back to the analysis stage of UX design, another method I would like to focus on is known as a user story. User stories are simple to make and are extremely effective at keeping the focus of a task on a user and providing a reason the task needs to be completed in the perspective of the user. A commonly used template is the “as a [noun], I want [feature], because [reason]”. For instance, if I were developing a timetabling application, then as a student I want to see what my timetable will look like in a weeks’ time because I want to plan my work around the days I have off. These can be used to break a large project down to “epics” which are chunks of the project split into different sections of which are further broken down to simple tasks that, when combined, create the large project. Lucassen et al. (2016) surveyed 21 practitioners on the effectiveness of user stories regarding their overall work productivity and work quality. They found that an overwhelming majority of 61% of respondents agreed or strongly agreed that their productivity increased with user stories, with 31% of respondents finding no difference in productivity. Similarly, 68% of respondents agreed or strongly agreed that user stories increased the quality of their work with 23% finding no difference. From their results, it clearly shows that user stories are extremely effective at increasing both the developer’s productivity and quality of their work. Its important to note that user story effectiveness is highly dependant on the complexity and size of the project. For my project, I would find that using user stories would be extremely useful as I can rationalise my features using user stories as well as keep track of the purpose of a feature that I will create with a complex implementation using a simple user story to describe its function and its use.

In the design stage, prototyping is a broad term defined as a draft of a product to investigate ideas and the overall design concept to their target audience before investing their time and money into a full-fledged project (*Prototyping | Usability.Gov*, n.d.). Prototyping in UX is an umbrella-term that encompasses low-fidelity protypes and high-fidelity prototypes. Both have their advantages and disadvantages, but I wish to focus on low-fidelity prototypes as this is what I will focus on mainly early in my project. Low-fidelity prototypes are quick and relatively easy to make and provide a lot of insight on the design from which new ideas can be communicated and decisions can be made to move forward from the prototype (Gerber & Carroll, 2011). This also has a positive side effect by, making the workflow more efficient in the long run when working on the project. Walker et al. (2002) found that low-fidelity prototypes major advantages was in cost, ease of iteration and allow designers to target interaction design. Gerber & Carroll (2011) also found from their investigation on the psychological experience of prototyping that practitioners tend to reframe failures into opportunities to learn, reinforces everyone’s beliefs about their creative ability and conjure a sense of positive or forward progress. In summary, these studies prove that low-fidelity prototypes are not only good for the product by helping the products evolution, but it is also good for the designers of the product as it reveals faults in their work, makes them more confident in their ability to design the product and gives a sense of improvement with each prototype. That combined with the ease of their creation makes it a great tool for me to utilise in the design stages of my application. I can gather opinions from peers and reflect on features I plan to create that are implemented in my prototypes like in Figure 8 and iterate over the designs till I am happy that most to all the issues discovered are sufficiently dealt with.

## Summary

The research on the above topics during the making of this document made me aware of many considerations I should stay aware of when designing my application from all aspects such as security, implementation of the concept and designing the app around the user’s rather than just around how I, as the developer, would like to see the app used. I already had a trivial understanding on the themes discussed on this literature but the scope of potential issues I would have to approach and solve magnified upon conducting further research on the above themes. I realised that since I am no longer making a product that only I will use, but others will use, I need to ensure their safety as well as mine. As much as the features matter as well as the overall concept that encompasses them, their data’s safety is paramount to a project like mine. On the other hand, I have also learned of many processes, templates, conventions, and standards that I would like to utilise in my project that will not only make it easier to break down the project but also improve my productivity and work quality. I am on a time constraint for this project so anything that can ease the workflow or make it more efficient will provide me with additional time to ensure that the quality of my work meets a certain standard or add features that I would like to implement if time allows. Finally, at the origin of my project idea, I wasn’t aware of the existence of applications that also followed the same concept of gamifying physical activity, so I didn’t understand whether it would be a viable commercial venture or a doomed project. I now know of a variety of products that have existed and utilise the ideas I have also proposed, with statistics and research done to show both the positive and negative effects of their features followed by potential improvements in future implementations of the idea. I aim to have a unique implementation of the concept and learn from the mistakes of the preceding implementations to produce a more quality product. It won’t be perfect, but it is a learning experience of which I believe I will make many mistakes in, but I will learn much from.

# Project Planning

## Introduction

The details within this chapter will cover the chosen development methodology, tools and techniques required, legal and ethical issues, as well a set of requirements separated by the MoSCoW prioritisation technique.

## Methodology

There are two methodologies I know of when it comes to the process of software development: Waterfall SDLC (software development life cycle) and the Agile methodology. The Waterfall SDLC is very linear and rigid - having iterative steps to move the development cycle forward, whereas the Agile methodology is much more cyclic by working on a few tasks following similar steps to the Waterfall SDLC in something called “Sprints”. Each Sprint is designed to be completed within a short, time-boxed period that will allow them to work on those specific tasks (Rehkopf, n.d.). Following each Sprint is another Sprint and that continues until the project is completed. A Sprint is synonymous with iterations on a project; So, for each Sprint, there is an iteration of the project that is working towards completion.

Both methodologies have their differences and are more applicable in some cases than others, there isn’t a best methodology for all cases. When comparing the efficiency between the Waterfall SDLC and Agile methodology, it was found by Mccormick (2012) that the Agile methodology was more efficient due to its adaptability to be used within the real world. Mccormick (2012) describes the Waterfall methodology as being a lot more “rigid” and claims it is better applied on a project that is already complete and stable which requires additions and among that, the flaws within the project undertaken should be accurately predicted to make the Waterfall SDLC worth using in a situation; There is no space in the Waterfall SDLC to make any last-minute design changes. The pre-requisites alone when trying to work efficiently with the Waterfall SDLC are extremely specific to get the best out of the methodology. Given my project does not meet those pre-requisites to make effective use of the Waterfall SDLC, it is unlikely that I would consider choosing this methodology for my project.

The Agile methodology in contrast, can respond to changing requirements effectively and consequently, can handle design changes at the last minute. By using prototyping and iterations using Sprints, it is possible to focus on a specific set of tasks for a Sprint and complete features to a degree that is good enough for the end artefact. It was concluded by Kumar Bhatia & Jambheshwar (2012) that the Agile methodology increases productivity and quality innately when utilised effectively and further emphasized by Maruping et al. (2009) by stating that in their conclusive study that in a situation of a high frequency of requirement changes, Agile would be most effective in terms of promoting an increase in the project quality. Provided by the research I have done to find the most effective of the two methodologies I know of, the Agile methodology seems the most applicable to my use case as I am creating a project in which I am uncertain of the changes I would have to make to ensure features work accordingly to the specification I have created in Appendix 1 and Appendix 2, that will also work within the selected frameworks I am using for development. I plan to iterate over the development of my project as well to have functional prototypes ready enough for testing at each stage to provide quick feedback on improvements needed to enhance the quality of my product.

Therefore, the chosen methodology I will use to develop this project is the Agile methodology. I have anecdotal prior industry experience with the Agile Methodology and upon undertaking the project, it was already indicative to me that this project would do well with this methodology for the above researched reasons. Testing each iteration is also not limited to me; For this project, my intended user demographic happens to be quite broad and widely available to me. From past mistakes, an issue that can occur is spending too much time on an aspect of the project in which the time spent is not worth the result. To ensure this doesn’t occur, I plan to implement Sprints on a bi-weekly basis to shift my attention to components in my project that require more time and effort than the small shortcomings I am focusing on. Hence, it is extremely effective in optimising my time spent on each feature and together with MoSCoW prioritisation, I can ensure I am getting the best overall result over my time spent on this project.

## Requirements

To be clear, I have acknowledged that the project artefact will not contain the full list of features to be presented. To reiterate, the MoSCoW prioritisation is a technique to prioritise tasks in a way to differentiate the most critical of tasks from those that are not as impactful but are nice to have from the end artefact(s). I realise that those features I ideally would like in the final product may not be introduced due to time-constraints.

### Must

* Have a functioning app with a Google Maps implementation that tracks the device’s current position.
* Have at least a marker for which the user must travel towards (a quest).
* Have a way for the user to randomise their next location to walk towards via UI interaction
* Allow for customisation for distances that the user can travel.
* Check if the device location is at the quest location and create a notification on the device to notify user of the completion of their quest.
* Save data locally on device to keep track of information the app has procured while in use that would be useful to the next launch of the app.

### Should

* Create a server application to compute the most CPU intensive operations.
* Create an API in which the server can be interacted with.
* Host the server implementation to be accessible by the web.
* Establish successful client-server communications between the app and the server.
* Defer all CPU intensive tasks running on the app to the server.
* Add a radius on the maps implementation to allow user to walk within a range of the quest objective.
* Add a SQL database in which the server can communicate with and store data.

### Could

* Add a login system in which users create an account and login via the app communicating to the server.
* Save user specific details such as gamified elements including experience points and currency.
* Save details regarding the user’s current quest, experience points and additional user specific details to be implemented on the users record within the database
* Track the users step count for the day.
* Add a leader board for experience and/or points.

### Would

* Track the users step count for every day.
* Create a graphical representation for the steps covered per day, week, or month.
* Add in rewards to spend the user’s currency points.
* Add in a friends list in which the user can communicate with others who have signed up to the app.
* Add in a chained journey mode in which the user can travel to multiple points and gain more experience points for each subsequent journey attempted.
* Allow users to share quests and attempt a group walk between each other with bonus rewards.

## Development Decisions

### Android or iOS

My planned chosen IDE for this project is Android Studio. I have had prior experience with this in my second year of University and I worked mainly in Java. Kotlin is another language I could develop in as well that has in recent years, become the official language of Android Studio (Lardinois, 2019), however due to my lack of experience with the syntax, my preferred language for building the app is Java. Another option I have considered for this project is using a framework called “Flutter” in Android Studio that utilises the “Dart” language. From many previews of sample work, this is by far the ideal choice when building an app as the framework can compile the Dart languages code into an app that is suitable for both iOS and Android. The main advantage to me is being able to expand the scope of the app beyond Android as using Java to create the app comes with the cost of only developing in Android. Yet the reason why I haven’t committed to using Flutter is because I unfortunately found out too late. I have already planned too much for the app in Java and with Flutter being a newer framework, its susceptible to frequent updates which deprecates existing functionality and therefore requires more maintenance in the long run. The Dart programming language is also unknown to me and while it doesn’t look very complicated since it is an Object-Oriented programming language of which I am familiar with many languages structured around objects, I cannot guarantee that I will be able to solve issues further in development due to lack of knowledge or lack of supporting documentation around a topic. Java on the other hand has lots of community support and is one of the few languages I have enough background on to start a project of this nature.

## Tools and Techniques

### No-IP

I discovered a platform called “No-IP” recently that is a dynamic domain name system (DNS) that will take a user’s dynamic IP address when they connect to the internet and direct oncoming traffic to their static IP address on their own network and check every 5 minutes if it needs to re-map the dynamic IP address in the case that it has changed as described by Geekboots (2019).

This is incredibly useful as it makes locally hosted databases and servers a lot more viable for a solo developer. With the correct setup, I would not need to pay a penny to any service to host my database or server to be accessed globally. No-IP also provides a free host name of the users choosing, from which it can connect to hosted material from the user, so I would not need to pay for the domain name either. If used, this service can make my project available on the web for free, allowing me to use my application anywhere and at any time given my personal computer is on. The disadvantage of the service is the pre-condition that whatever is using its service is constantly on, in that way I am still paying for upkeep as I will be having to keep my computer on. But for the time being, for testing purposes and for just a university project, this solution should suffice.

### GitHub

GitHub is a version control service where any user can store and maintain their code for their software to build up in each iteration. For each commit you make, all files affected by your changes if any would be stored in a separate commit, allowing you to back track through your previous iterations in the case your most recent versions are unstable.

## Legal, Social, and Ethical Issues

### Legal Issues

As said previously in Appendix 1 and Appendix 2, a large cause for concern is for users walking towards unreachable places or private properties. This is as much as an ethical issue as it is a legal concern as I cannot control where the end user would walk and the consequences of doing so would be. First and foremost comes the user’s safety when using the app itself, I cannot foresee if the app (when it is completed enough for submission) will provide a location for the user to walk towards that will not cause harm in any way to the user. A similar app that my app can be compared to is “Pokémon Go” – it saw much success upon its release in 2016 but along with the mass number of users came several cases where users had accidents that resulted in injury or death by using the app unsafely. A documented case by Derbyshire Times (2016), reports that a young woman had been playing the game and slipped causing multiple injuries. Other cases resulted in death such as where a child had been killed by a truck driver who was at the time, playing the game while driving (Sim, 2016). Given this, the app’s demographic is directed towards late-teens or adults. My aim is to keep the app engaging but not too engaging to ensure the user pays attention to their surroundings when and where necessary to reduce accidents. While this limitation comes at the cost of adding certain elements to the app to increase engagement, I believe it is a necessary cost to make the app safer to use for the end user to ensure a user doesn’t lose their life due to being distracted by my app.

As well as that I have thought about creating:

* A terms and conditions document to be clear of the users the consequences of using the app outside the way I (as the sole developer or possible future developers) have intended for it to be used.
* A privacy policy document to disclose to our users how we will manage their data. As much as a terms and conditions is a guideline set by the developers for the users, this would show how the developers would use the end user’s data. Every user has a right to know how their data is being handled and a privacy policy would do that for them. I would at minimum follow the principles of the Data Protection Act (UK Government, 2018).

### Ethical Issues

As also described in Appendix 2, data security is not a small issue and many large corporations in the past have suffered from data breaches and have had to pay a large fee for exposing user information to an undisclosed party of which they failed to stop. One documented case is from Tidy & Molloy (2021) where an organisation called Twitch had been breached and had exposed more than 100 gigabytes of data, of which contained transactions between them and their top contributors to their platform. I plan to make a login system that as a consequence of using the app, will take in user data. While most of it is statistics from the user and does not provide much in terms of exposing their identity, I still wish to minimise the amount of data a user must provide to use my app without issue and so I will consider my options during development and weigh their importance to the app against the amount of user data I need for it.

## Kanban board for project management

Project management is key to getting work done within time constraints and while there are many concepts out there to follow and maximise the efficiency of my development time, there are also services that help you organise your tasks to also help along with the development process. The one that I used is Trello, which organised my project into a single board and tells me what needs working on, in what order tasks need to be worked on, what is done and what is currently being worked on (Trello, 2021).

Graphical user interface, application, Teams

Description automatically generated

Figure 8 – Example of using Trello to manage my project timeline.

The above figure is shown in greater detail in Appendix 3. The above shows the tasks that I plan to do, the features I have been working on and the features I consider as done. Among each card is a coloured label where they represent the Must, Could, Should and Would in MoSCoW prioritisation as red, orange, yellow and green respectively. Any issues that come along with a feature that is considered “Done” is added as a new card in the “To Do” section and placed appropriately as I work on each card sequentially from the top. There are improvements I can make to the structure and better ways of going about managing my project but personally, I found this to fulfil my needs for my project. It goes straight to the point, was quick and easy to set up and is even quicker to modify.

## Summary

I have discussed how I used the MoSCoW prioritisation to my advantage and how it is enforced with Trello and a few tools I have used to help my project in both the short and long term. Legal and ethical issues were also covered as well as my choice of integrated development environment (IDE), operating system I plan to work with and methodology I plan to use during the time I spend developing the artefacts and any related material on this project.

# Design

## Introduction

While having a functional app is great, a user not only cares about its function, but its looks and the way it conveys information pertinent to the user. This chapter will cover the details and importance of system architecture design for developers and user interface (UI) design for end users along with user experience (UX) design.

## System Design

System design is a broad term that defines the process of creating components, modules, and interfaces of a system as well as the flow of data through each to satisfy the requirements of a system. System architecture is a subsection of this concept and exists to organise and assemble the components of a system.

### Model-View-Controller design pattern

The Model-View-Controller (MVC) design pattern is used primarily by my server. It is a cycle that allows for data within the system to be interpreted by the user of the system. Each word within the term has a specific use in a system:

* A Model is a component that encapsulates data and the behaviour of that data related to the system. They can be used to communicate between a database and a server to send and receive for a specific purpose or to send back data to the user when serialised in JSON through a data-transfer object (DTO).
* A View is a user-interface that is shown as the front-end for a server application. It allows for data to be visualised to the user in a way that is interpretable by the user.
* A Controller is the component that allows for the user to input data and for it to be manipulated appropriately (depending on what the user requested). Post-manipulation this data can then be taken from a Model, from which the Controller then updates a View with the Model data for the user to interpret again.

A depiction of this cycle is shown below in Figure 9*.*

Figure 9 – Illustation of the Model-View-Controller design pattern (Rhoudunda, 2020)

Diagram

Description automatically generatedAs well as using design patterns to make my server API more understandable, I have also structured and ordered my files by feature. In my server application, each component and module that is to do with a certain feature is grouped into a library specific for their use. For instance, there might be a specific user controller and model, they would be placed in folders named “User” and placed in the class libraries “Controller” and “Model”. Figure 10 provides a visual example on what I mean by this.

Figure 10 – Example file structure within server application

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generatedSimilarly, in my client application, I have all my features separated into folders specific to each feature, I could have created packages, but I don’t believe the contents of the client application provide a more meaningful structure than separating each feature into a separate folder for the client-side of my artefact. Figure 11 provides a visual example the of file structure within the client application.

Figure 11- Example of file structure within client application

### Database Design

Graphical user interface, table

Description automatically generated with medium confidenceThe databases’ structure is more important than the file structure of a client or server. Since a database will be containing data related to all models, any expansion above the initial design must be carefully considered as this may cause many side-effects as the database scales, some issues related to the database design warrant an entire refactor of a table or multiple tables given the magnitude of the change. When a controller takes in input from a user, it gathers data related to the request from the database, so while the server computes the data, the database is the store for the data. Below in Figure 12 is an Entity-Relationship Diagram of the proposed database I have created so far.

Figure 12 - ER Diagram of database design so far

Among that, I have also considered the normalisation of my database structure. Normalisation is defined as a technique that reduces the amount of data that is unnecessary to store and decreases anomalies when using databases operations like using insertions or deletions (Peterson, 2022). There are many levels of normalisation (they go between levels 1 to 6) that effectively reduce the amount of data stored and changes the structure of data. The structure I have followed in Figure 12 is the third level of normalisation, also known as 3NF (third normal form) in which it must comply with 2NF and have no transitive functional dependencies (Peterson, 2022).

A 2NF database structure requires 1NF (which is simply each cell in a table containing a single value and each record made being unique) and must contain a primary key, indicated by the letters “PK” in Figure 15. A primary key is an identifier of a record and is typically and integer that starts from one and only increments, thereby creating a unique value associated with each record. For 3NF, transitive functional dependences are basically breaking down a table further such that columns within a table that depend on the value of another column in the same table is instead moved to a new table. Given the example of a gender, if the name “Howard” was associated with a title or honorific of “Mr” (mister) but became transgender and changed their name to “Holly”, their title or honorific would also change to “Miss” or “Mrs”. The title or honorific is a transitive functional dependency and so in 3NF, this would be associated with its own table and instead be linked with an Id to the main table on the record with the name “Holly”. Similarly, my CurrentQuests, UserSettings and UserActivities tables are all following that principle by separating themselves from the Members table and instead being linked by ID’s associated to a record within the Members table.

While as of now the database structure in Figure 12 is concise, more will be added to the database design when it is needed. This comes with the caveat of needing to refactor a table for a feature but for what I have planned, I believe this won’t take much time to do.

### Client Application Design

The plan for the client application is to have a plethora of activities in which the user can engage with but to get a good idea of what I need the system to be like, I must generalise the code down to a manner that is understandable to me and is easy to interpret for me. Visually, this is accomplished by using Activity Diagrams, which details how inter-related components of a system work dynamically to achieve a goal. It is in layman’s terms, an advanced flow chart showing the flow of an activity in a particular system. An example of this is shown in Figure 13, it details the flow of the system when the user clicks upon the login button. It includes everything I think should happen, including the handling of errors and communication with my server to get all the data required for logging a user into the app. Among that, many other Activity Diagrams I created illustrate the flow of different functions throughout my application. Some functions were too simple to be worth creating an Activity Diagram for, such as the Menu button or Back button, both of those are obvious in name and purpose. Other navigational buttons were also quite simple in name and purpose such as the Settings menu item I plan to add and so, for the more complicated features, these Activity Diagrams breakdown the code’s functionality in small pieces to follow along and base my code upon. This simplifies the complexities of each feature massively and is a great tool for understanding the flow of the system before it’s even been created.

Diagram

Description automatically generated

Figure 13 – Activity Diagram displaying the process of logging a user into the application.

### Server Application Design

As mentioned before, my server uses the MVC design which primarily focuses on the interaction with the server but not the server’s full architecture. I plan for the server to utilise many classes besides the controllers within the MVC design pattern. These would be used to further compartmentalise the manipulation of data caused by requesting data from the server or sending data to the server. They would serve to follow many programming principles such as “DRY” which is an acronym for “Don’t Repeat Yourself”, which exists to make sure repeated code is instead placed in methods to be re-used throughout the entirety of the program. While some or most of the work-in-progress architecture I have illustrated in the Class Diagram shown in Figure 14 below will have most likely changed from the end result of the implementation, it presents what I think my classes will look like, what functionality they would have and how they would interact with each other. The diagram also includes the controllers I believe I will implement into the application.

Figure 14 may look like the most basic form of a Class Diagram, but it is used as a guide to roughly map out what I think is required from my server given the requirements I have set for the project. Within the diagram, I am unsure of the fields and properties I would include in each class, but I know that the “DatabaseContext” field would be used everywhere as it represents the variable used to interact with my database. I plan to use a package called “EFCore” in my server architecture, it allows for me to easily interact with my database using C# syntax and is interacted with by using a database context - represented as a field on each class on the diagram. This context is provided an instance at runtime due to dependency injection, therefore I know that it is going to be used everywhere throughout each class that manipulates data. Alongside that, each class is associated with one instance of the other, I plan to have no multiplicities within my architecture as I fail to see the use of it. If the Class Diagram had included the queries from EFCore (which would make the diagram exponentially larger), a variety of multiplicities would be shown but given that it is only a rough mapping of my server architecture, I found that details surrounding the specifics were redundant enough to not be included into the figure.

Diagram

Description automatically generated

Figure 14 – Class Diagram showing the proposed server architecture (prone to changes depending on future implementation)

## User Interface Design

For the most part I have created prototypes and justified most of the functionalities with a user story. An early digital sketch can be seen earlier in [Section 2.4](#_User_Experience_Design) in Figure 7 that depicts a low-fidelity prototype of my app’s design in its early stages. This provided a base overview for my UI to be created upon but as I finalised my set of features I planned to implement, I decided to make a more high-fidelity prototype using a free-to-use website called “Figma” that has both prototyping and code generation capabilities for designing an app (Bracey, 2018). With it, I was able to create a fast and easy prototype of my UI accurate enough to use as a reference during my designing and use its code generation feature to incorporate some of the assets created, directly into my own application. As a disclaimer, the end artefact may not fully reflect the incorporated features within the prototype. The prototype reflects what I envision the application may look like by the end of the implementation but given that I will more than likely encounter issues during development, I cannot be sure what will and will not be included from the end artefact that is derived from the prototype.

### Maintaining consistency

One of the concepts I used to make sure the user experience was the best it could be was Nielsen’s 10 usability heuristics. They are named heuristics as the rules should be treated as generalised guidelines and not as rules that an artefact must comply with (Nielsen, 2020). One of these rules is maintaining consistency across my client application. UI can be descriptive, but the best UI doesn’t need explaining to understand what it does. Take Figure 15 for instance, across each screen linking to the main screen, there is a button that takes you back to the previous screen. This is consistent across each of my screens and is easily recognised as a button to press to leave a certain screen.

As well as that, I will provide information most pertinent to each user at the top of each screen, this descends to less pertinent information until it becomes miscelaneous information that the user may find interesting.

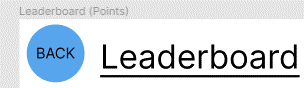
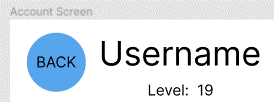
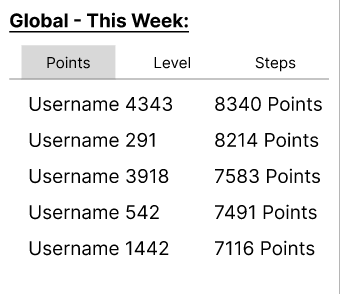
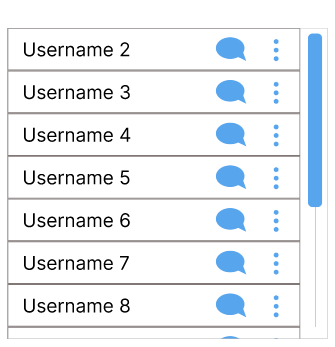
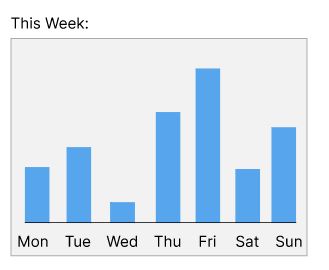


Figure 15 - Back button at the top left of each of my sub-screens

### Showing lots of information succinctly and elegantly

Features such as the leader-board, friends list and graphs need to show a lot of information in a compact space. It’s important to display as much of that information to the user without overloading the screen with too much information to digest. This is a point made clear by Nielsen (2020) when attempting to create an aesthetic and minimalist design – it is important to prioritise the main content when applying minimalism. For this reason, I have attempted to show as much information for the feature to be useful while making it concise enough to ensure the user does not get confused or find it hard to interpret. Figure 16 shows how I plan to design my screens showing large sums of data in one compact area.

Figure 16 - Depiction of graph (Accounts screen), friends list (Friends screen) and global leader-board (Leader-board screen) from high-fidelity prototype



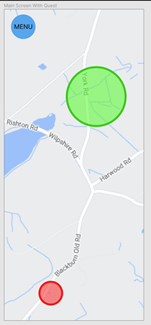
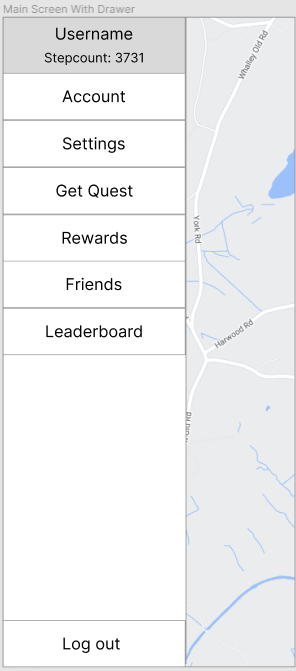
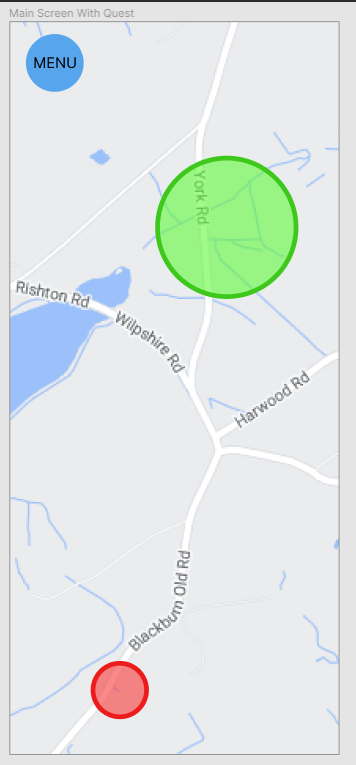
The graph contains each day of the week with bars representing the step-count, with all the bars being sized relative to the day that has the most steps that week. If it is feasible to implement, I would also add the number of steps accumulated per day on top of each bar. The friends list has a scroll bar that will appear as more friends are added to not clutter the friends screen and the leader-board screen has a selector to choose which leader-board the user would like to view and the top 5 users in each category.

### Keeping the UI simple

As for the main feature of the app (where a user starts a quest and walks towards an objective shown on the map), the logo image itself shows the main feature of the app. The main feature is represented in the same fashion on the main screen as it is depicted in the logo itself. This idea was derived from the rules Nielsen (2020) stated of having a minimalist design but also to have the user recognise features as opposed to recall a feature. Figure 17 shown below surmises the equivalence between the logo and the main screen’s actual depiction of the main feature itself.

Along with that, the focus of the game within the app is to prioritise the main feature on the main screen. All navigation elements are deferred behind a button considered as the “menu” button for my app and allows the user to access all further features via the navigation drawer that reveals itself upon the menu button being tapped. My intention behind this design is to have clear vision of the objective when using the app while all side-features can be accessed by a button in the top left, a bottom or top navigation bar is not apt for my app since the user should be able to navigate the map without issue. A navigation bar could not only cover parts of the map but can also be pressed accidentally depending on where the user is attempting to shift the map or how they are holding their device. Figure 18 further below shows the night and day difference between having a navigation drawer open and closed and shows their impact on the user experience overall when the user is focused on the main feature.

Figure 17 - Demonstration of the equivalence between the main feature on the main screen and the logo image



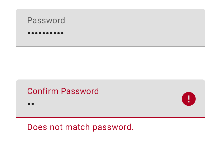
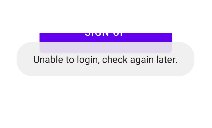
### UI Error Handling and packages in Android Studio

Another rule I have generally followed by with Nielsen (2020) on his 10 usability heuristics is including error prevention measures and making sure errors easily noticeable and interpretable by the user when encountered as well as recoverable. My client application is dependent on the components within a user’s device and their availability. I will need to send data about the user back to my server from time to time but this process may be interrupted if the user walks towards a location where they will not have access to the internet. For this reason, counter measures are put in place to make certain that the app will not crash on the user given that the resources that are required are not available or the minimum criteria for the app are not fulfilled for actions to take place. These counter measures must be interpretable by a user with no understanding about the system internals. I cannot expect every user to understand what a “404 Not Found” means in the context of the system. While the Android software development kit (SDK) itself has built in functions that can catch errors and return a UI element to describe errors in as simple detail as possible, for errors pertaining to for example – a username field in the sign up process having too few characters, the Android SDK itself doesn’t have a great way of automating the translation of those error messages to the UI in a method that does not fulfil my standard for the user experience.

Figure 18 - Separate states of the main screen exemplifying the difference between having the navigation drawer open and hidden during gameplay

Continuing from that, I plan to use a package called “Material Design” which describes itself as following the best practices of user interface design by having an adaptable system of guidelines and components (*Homepage - Material Design*, n.d.). While the components within its library look fantastic, it also subtly caters for the user experience by merging the error information below the field the user has typed in. For this reason, I have used Material Design components throughout my app to maximise the error handling feature that comes with applicable components. Figure 19 below shows error handling on the UI caused by user error using Material Design and errors caused within the system handled by the default Android SDK. Material Design will show the error directly below the field associated with the error. The server error associated with Figure 19 is produced when the server is not currently running but worded in a way to not confuse a user that does not understand what a server is or its function. It suffices enough as an error message, given the context of how the system errored and in turn, what it means for the user, rather than providing the system error itself.

Figure 19 – Android SDK showing server error and Material Design showing user error respectively.



## Summary

I have described in length the designs I plan to use and have created to support the creation of both the app and server as well as the database. For the UI, I have also considered the UX of my application and planned carefully with a high-fidelity prototype on the general placement of UI components to ensure that the UI looks good, but more importantly, to make sure it feels good for a user using my app. To get a better look at the prototype, Appendix 4 contains the entire layout for the prototype and each of its components and Appendix 5 contains a link to an interactable version of the prototype.

# Implementation

## Introduction

For this chapter, I will present both the major and minor implementation details about the server application and the client application. Within those, topics surrounding database interaction, development of features, implementation of a variety of systems and human computer interactions (HCI) will be detailed and expanded upon.

## Server Application

The server application is more important than the client application in my case for my implementation of thin-client architecture (discussed in [Section 2.3](#_Thin-Client_&_Mobile)). I have deferred all computationally time-consuming tasks to the server including the computation of the main feature. The server is the medium between the client application and the database and can manipulate data before it reaches the client app to provide filtered data that is easy to work with in the client side. This results in a more efficient usage of time developing the client-side and resources for the end user.

### Controller and data manipulation

Controllers as discussed in [Section. 4.2.1](#_Model-View-Controller_design_patter) take in data and manipulate the data to return to the user and/or write or read from the database. Below in Listing 1 is a controller specific to functions related to quests, it gets a new location for a user to walk towards and its input consists of their member ID, their current location and their user settings. Given the data is not provided, appropriate error messages are returned to ensure that the server does not crash, and the user understands why an action has not taken place. On the last line in Listing 1, I am accessing a method in another class called “GetQuest” of which will return a result that I would be able to send directly back to the caller of the endpoint.

[HttpPost]

public async Task<IActionResult> GetNewQuestLocation(int memberId, [FromQuery] Location userLocation, [FromQuery] UserSettingsDto userSettings)

{

if (!ModelState.IsValid) return BadRequest("A validation error occurred.");

if (userSettings == null) return BadRequest("User settings are empty.");

if (userSettings.MaxDistance.Equals(userSettings.MinDistance)) return BadRequest("Max Quest Distance must be less than Min Quest Distance");

return Ok(await \_questRandomiser.GetQuest(memberId, userLocation, userSettings));

}

Listing 1 - [QuestController.cs] Endpoint to get a new quest location (C#)

The method “GetQuest” will produce a quest location for the user depending on their current location and the bounds set within their user settings. Listing 2 shows the computation taking place to set a quest location in the GetQuest method.

var radiusInDegrees = userPref.MaxDistance / 111f;

while (IsNewPointWithinBoundary(questPoint, currentLocation, userPref, out selectedDistance) != true)

{

var u = \_random.NextDouble();

var v = \_random.NextDouble();

var w = radiusInDegrees \* Math.Sqrt(u);

var t = 2 \* Math.PI \* v;

var x = w \* Math.Cos(t);

var y = w \* Math.Sin(t);

questPoint = new Location(y + currentLocation.Latitude,

x + currentLocation.Longitude);

}

Listing 2 – [QuestRandomiser.cs] Segment of method showing how a quest location is set (C#)

In the method, there is a while loop which has the objective to check that the quest location is within a boundary set in the user preferences in kilometres. If it is above the maximum quest distance or below the minimum quest distance, the while loop will continue until it finds a location that satisfies the bounds within the user settings. The contents of the while loop attempt to set a quest point that the user can walk towards. It takes in two random doubles and goes through a small bit of computation to get a location that is roughly near the user’s current location, at which point it finally returns this new data to the controller to return to the user.

### Quest System

The quest system is the core of my project, it is the gameplay element all users will interact with to play the game within the app. There are also supporting systems such as the “Reward” system detailed further in [Section 5.4](#_Reward_System) that work in tandem with the Quest system to provide rewards attributed to each generated quest.

#### Getting a user’s quest location.

In Listing 2 within [Section 5.2.1](#_Controller_and_data), I described the function of the GetQuest method but not what the formula within the contents of the while loop was doing. A brief explanation of each line of maths:

* radiusInDegrees – value for a chosen radius in degrees given 1 degree’s equivalent to the radius of the earth. Calculated by taking the maximum distance the user has set in kilometres and dividing it by 111. The 111 represents the radius of the earth multiplied by π all divided by 180. The result of this calculation would be approximately 111,300 but since we are working in kilometres and not metres, this value is divided by 1000 to get an approximate value of 111.3 and further rounded to get 111.
* The variables u and v represent the random values between 0 and 1 that will determine the location of the quest.
* The variable w is a factor that allows the end values of the latitude and longitude to be within the bounds set by the user, the randomised part is derived from variable u.
* Variable t is used to determine a uniformly randomised circumference and used as an input further in the formula, the randomised part comes from variable v.
* Final variable x takes variable w and multiplies it by the cosine of variable t to get a value that is within the bounds the user set and can be added to the user’s current location to get a new latitude. As stated previously, variable w is a factor to keep the end value in the correct bounds whereas the cosine of variable t can produce a value between -1 and 1 depending on the value of t. Cosine determines the ratio of the adjacent side to the hypotenuse in a triangle and the adjacent side is measured on the x-axis or in the real-world equivalent: the longitude.
* Similarly, the final variable y determines the longitude in the same fashion as variable x but this time we are using the sine of t. Sine determines the ratio of the opposite side to the hypotenuse in a triangle and the opposite side is measured on the y-axis or its real-world equivalent: the latitude.
* Lastly, the variable x is added to the user’s current longitude while the variable y is added to the user’s current latitude, the result is a location within the users requested radius which is considered as the quest location.

Diagram

Description automatically generatedFigure 20 below is a visual aid on the above explanation of each of the components within the method, it illustrates the uses of the concepts described to achieve the goal of getting a quest location.

Figure 20 – Illustration of concepts used and end result of getting a quest location

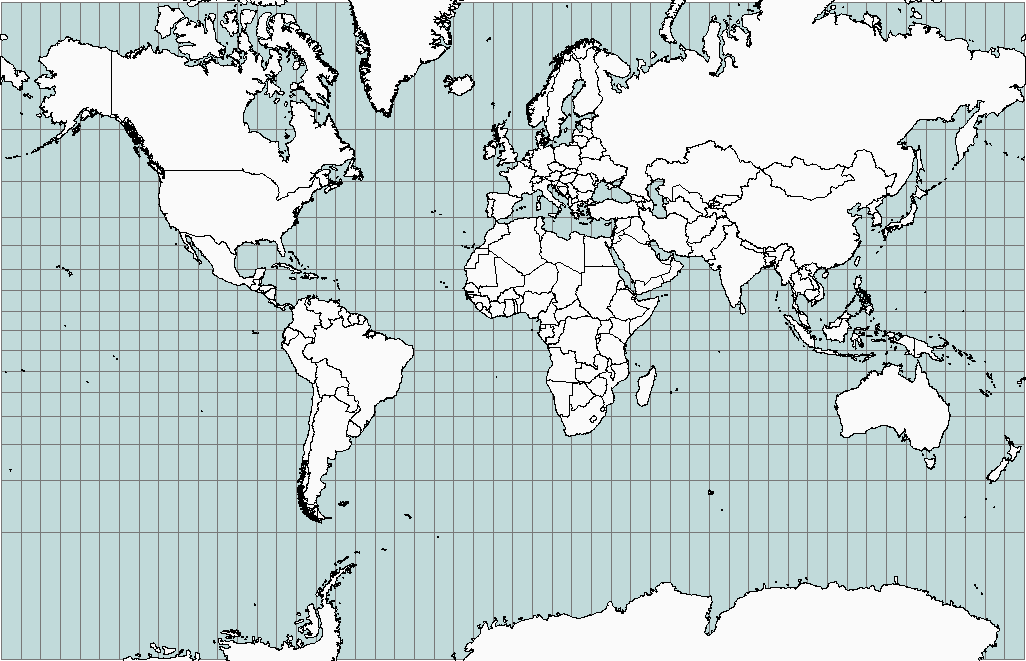
Originally, the computational element that decided the point of the quest location was governed by the Haversine formula which is a formula used to find the distance between two points across a sphere. The issue here was that since Google Maps uses the Mercator projection, quest locations that were set relative to the user’s current location were a greater distance away from the user’s location in the north and south but remained perfectly fine in the east and west. This is clearly seen in Figure 21; all columns are evenly spaced but the rows on the grids have varying length’s that expand across the length of the map the further they are from the equator. This matches my observation that quest locations set in the north and south seemed to be further away from the user than points set in the east and west.

Figure 21 – Depiction of the Mercator projection and its distortion pattern across the entire map.

As a result of this, the formula was modified to find the distance between two points on a circle and that is reflected in the current implementation shown above in the current section. Assume that in an extreme scenario, a user intends to walk up to a maximum of 100 kilometres in a quest - while 100 kilometres seems like a lot, this figure barely scratches the circumference of the earth in kilometres. Across the 100 kilometres, when highlighted as a radius on the sphere of the earth’s size, the radius would have an extremely low amount of curvature to it. Heuristically, a section of a sphere with near to no curvature can be approximated as a simple 2-dimentional circle. The current implementation of the algorithm following that logic now picks locations within a circle in the radius of the user rather than on a sphere. Therefore, it negates the distortions on any 2-dimentional projection of the earth since I am working with circles and not sections of a sphere. The inaccuracies in this formula would only be noticeable if the user intends to travel many hundreds of kilometres to their quest location, which I believe to be an extreme minority if any. The goal is to be accurate enough and I believe this formula works fine for all intents and purposes of my app for this feature.

#### Checking a user has reached the quest location.

To check if a user has completed their quest, an endpoint was created that would be called at intervals on the client side to check if the user was within the location of the quest. For this to function as intended, I need to find the distance between two geographical coordinates and check whether that distance is less than the distance between the user and the quest. A problematic scenario described in [Section 3.6.1](#_Legal_Issues) is if the user cannot reach the exact position of the quest, then they cannot complete the quest. Moreover, since I am working with exact values, the user would need to be in the same latitude and longitude of the quest for it to register as completed. For this reason, I have added artificial radii for the user and the quest location to make sure that the quest can be completed if the user is at least within a close enough area to the quest. Below in Listing 3 is the implementation of this endpoint that is mandatory to have the endpoint working as intended. .

var isWithinQuestRadius = QuestDistanceChecker.CalculateDistanceInKilometers(

questData.QuestLocation, userLocation) \* KilometersToMetersFactor <= questData.QuestRadius + userRadius;

UserAccountDetailsDto updatedAccountDetails = null;

if (isWithinQuestRadius)

{

await \_questManager.RemoveCurrentQuest(memberId);

updatedAccountDetails = await \_questManager.AwardXPAndPoints(memberId,questData);

}

return Ok(new CheckQuestCompleteDto(isWithinQuestRadius, updatedAccountDetails));

Listing 3 – [QuestController.cs] Most critical code in the endpoint for checking the completion of a quest (C#)

As seen above in Listing 3, the method CalculateDistanceInKilometers takes in the user’s location and their quest location as input, the value returned is the distance between both location in kilometres. That is converted back into metres and then it checks if the distance is less than the quest’s radius and the user’s radius combined. If this is true, the quest is removed, and the user is rewarded. The last line returns a Boolean value that confirms if the quest has been completed or not, as well as updated account information. Both returned values are used in the client-side to allow the app to take an appropriate action based on the result.

The main component within Listing 3 that makes it work is the function CalculateDistanceInKilometers which determines the distance between two coordinates. Listing 4 below goes into greater detail of its implementation and how it works.

public static class QuestDistanceChecker

{

public static double CalculateDistanceInKilometers(Location questPoint, Location currentLocation)

{

const int ky = 40000 / 360;

var kx = Math.Cos(Math.PI \* (currentLocation.Latitude / 180)) \* ky;

var dx = Math.Abs(currentLocation.Longitude - questPoint.Longitude) \* kx;

var dy = Math.Abs(currentLocation.Latitude - questPoint.Latitude) \* ky;

return Math.Sqrt(dx \* dx + dy \* dy);

}

}

Listing 4 – Implementation of method that calculates distances between two points.

To provide a better understanding of how the code in Listing 4 works, I have provided an explanation of each line of code below:

* The variables ky and kx represent kilometres per degree in latitude and longitude respectively in relation to the size of the earth. Specifically, ky is the earths circumference measured at the equator divided by 360 to provide the amount of kilometres per degree around the earth.
* The variable kx is the cosine of the resulting value that is formed after converting the currentLocation.Latitude to radians using the formula π/180 and multiplying it by the latitude. This value is then multiplied by ky to scale kx to a value that is appropriate to the circumference of the earth per degree.
* After that, I found the difference in the x coordinates (the longitudes) and multiplied them by kx
* Similarly, the difference in the y coordinates (the latitudes) is also calculated and multiplied by ky
* Finally, using the distance formula (which is the square root of the differences in the x values (dx) + the difference the in the y values (dy) both to the power of 2) I was able to find the distance between the user’s coordinates and the quest’s coordinates in kilometres.

The methodology used in Listing 4 to attain the distance is not conventionally popular as it represents the earth as a flat circle with the circumference of the earth, this naturally will return a distance with a small degree of inaccuracy over short distances but greater inaccuracies over long distances. Usually, the Haversine formula is used to calculate distances between two geographical coordinates and is considered the convention as it represents the earth as a sphere. The justification for using this algorithm is the same as what was detailed at the end of [Section 5.3.1](#_Getting_a_user’s), the accuracy of this algorithm does not need to be exact to within centimetres for the distance, it just needs to be good enough. Considering the algorithm represents the earth as a flat circle, it reduces the overall computation required to get the distance between two points when compared to the conventional method of using the Haversine formula. The radius provided to both the user and the quest covers the inaccuracy of this algorithm, but the inaccuracy only matters over larger distances that cover many kilometres which is not what my app is oriented towards.

### Reward System

So far I have discussed how a quest location is set and how it is checked for completion but not about how I incentivise the completion of a quest. To gamify the walking experience, I have incorporated some of the common video game tropes such as accumulating experience points (XP) for completing a task as well as a form of currency to reward the user upon the completion of a quest.

Experience points are used to show the amount of progress a user has made by completing quests. You can only gain experience points and unlike currency, experience is not a currency that can be spent to gain rewards. It contributes to the user’s overall level; in video games this is considered to show the most basic level of prestige as it represents the amount of time and effort a user must have put into a game to achieve such a level. Statistics about a user such as this influence the creation of personal milestones that a user may want to achieve, which assists with overall engagement with the app.

The currency within the app is used to acquire an assortment of rewards from the rewards screen accessed in the navigation menu seen in the high-fidelity prototype in Appendix 4 and Appendix 5. These rewards could include a booster, additional themes, and other incentives to keep the player engaged with the app. The rewards in the rewards screen are not all too incentivising but one of the goals of the app was to allow socialising between friends and to also create custom reward tables tailored to each individual to utilise with their friends. This way, artificial rewards are created and are redeemed using an arbitrary number of currency points, and so a user is not limited to spending their currency in the rewards screen that is default to the app. Instead, they can spend it between friends by using their own rewards table that could include a variety of bespoke rewards specific to each friendship and is more local to each user and their friends. When the currency within the game is spent in this way, each transaction does not mean that the user will get their reward. At this point, the reward given between friends relies on the “reward giver” in this transaction to follow through with their end of the transaction, meaning the receival of the rewards redeemed is purely based on the trustworthiness of the “reward giver” to follow through.

Following this, it was also important to distribute enough XP and currency for a given quest. Different games have different strategies to provide a good enough amount of XP for the user to complete certain tasks but not too much XP to make the user overachieve and progress them through levels too quickly. Depending on the game, as the player levels up more XP is generally required to attain higher levels. Per level, some games require a player to gain an exponential amount of XP to progress to the next level while others require a general linear increase in XP to achieve the next level. For my project, I thought a linear increase was a necessity as most other video games don’t require the need to move towards an objective in real life, this by itself is a lot to ask for a user. Below in Listing 5, using simple arithmetic, I have calculated the amount of XP a user will gain for the distance they have walked and based on that, calculated the amount of currency points they will gain, and the amount of XP needed to achieve each level.

internal class EarnableCalculator

{

private const int KilometersToMetersFactor = 1000;

private const double XpFactor = 1.2;

public static int GetRequiredXpFromLevel(int level) =>

(int)(Math.Round(100 \* Math.Pow(XpFactor, level - 1)));

public static int GetXpForDistance(double distance, Random random)

{

var xpToDistanceFactor = Math.Round(distance \* KilometersToMetersFactor / 200.0);

return (int)(xpToDistanceFactor != 0 ? xpToDistanceFactor \* random.Next(25, 50) : 1);

}

public static int GetPointsForXp(int xp, Random random) =>

(int)Math.Round((decimal)(xp / random.Next(3, 6)));

}

Listing 5 – [EarnableCalculator.cs] A view of the properties and methods associated with earnable types within the game.

As seen in Listing 5, to get the required XP from a level, I simply raise the variable level to an exponent of the XpFactor which is 1.2, then I multiply that by 100 to get the required XP for a level. I calculate the XP for a distance that the user must walk by multiplying the distance (which is in kilometres) and convert it to metres then divide that value by 200. If this value is not 0, then the value is further multiplied by a value between 25 and 50, otherwise the user will receive 1 XP for their quest as it is deemed too small of a distance to walk to get any more XP worth the distance. Lastly, the points are derived from the amount of XP the quest is receiving, it is between a third and a sixth of the XP the user will receive for the quest. Randomisation is included in the formulae to reduce the consistency of quests receiving the same linear amount of XP and currency when the distances are similar and to therefore reduce how predictable the rewards within a quest are. By artificially adding in advantages and disadvantages in a quest, it invokes thought from the user to consider taking the quest or instead attempt to get another with better incentives.

### Login System

The login system was created to provide an identity to each user using the app. Since the app was designed to have social features, each user must have a unique identity to distinguish one user from another. The login system is quite rudimentary, requiring a username, email address and password to sign-up. The set of data provided by the user would be saved in the database and compared to when attempting to login to their account.

#### Password Hashing

Passwords upon sign-up are not immediately stored in the database as plain text. Instead, I have used a hashing algorithm called “Bcrypt” to hash a plain text password and then store it in the database. Bcrypt is a hashing algorithm designed to specifically hash passwords and is typically a good recommendation for a password hasher. It also generates a salt for a password upon hashing which further secures a password from being brute forced using a dictionary attack. Bcrypt’s main quality is that it is a slow hashing algorithm, while this seems like a disadvantage, it also means it takes longer for attackers to crack passwords but not too unbearably slow to allow for users with the correct credentials to log in, in a timely manner.

Other password hashing algorithms such as PBKDF2 and Scrypt were also considered as they are also hashing algorithms designed specifically for passwords. All three mentioned algorithms have their advantages and disadvantages but the overall best one to use according to Preziuso (2015) is Bcrypt. From a variety of sources online, there is a list of reasons as to why Bcrypt is better than other hashing algorithms:

* SHA & MD5 – These are fast cryptographic algorithms, as such they can equally as fast be brute forced with a dictionary attack, they are not specifically designed to be a password hashing algorithm. Bcrypt is slower and so by default, takes longer to crack.
* PBKDF2 – Is also a good password hashing algorithm but can easily be cracked by using a GPU or multiple GPU’s running in parallel. Bcrypt still takes a long time to be cracked on both CPU’s and GPU’s, effectively making it better than PBKDF2,
* Scrypt – It is an evolution from Bcrypt that solves its shortcomings but at the time of picking an algorithm to use, most sources discussing Scrypt were from years ago. Scrypt was introduced in 2009 and articles surrounding the hashing algorithm mentioned that it was too early to determine whether it was better than Bcrypt or not. Given Bcrypt was introduced in 1999 and remains unbroken to this day, I chose Bcrypt over Scrypt as it has withstood the test of time longer than Scrypt.

Bcrypt has also had enough time on the internet now that the algorithm is used globally on different programming languages and has had more efficient implementations since its release. Unfortunately, near the completion of my implementation of my client and server, I also discovered another password hashing algorithm called Argon2 that turned out to be better than Bcrypt. Argon2 had won the Password Hashing Competition back in 2015 (*Password Hashing Competition*, 2019) and was announced as the go-to algorithm by Preziuso (2019), who also recommended to discontinue the use of Bcrypt in newer systems. Since my implementation was nearly finished and test accounts were already created to test my systems, I have not changed my implementation to utilise Argon2. Bcrypt still remains unbroken and was until a few years ago, the recommended method; but it still is a good hashing algorithm, just that there is a better alternative. However, if I were to continue development on the project after submission, I would migrate to use an Argon2 implementation instead as it is currently the best in the industry when it comes to password hashing.

#### User Account Usage

User accounts are used to encompass a user’s journey with the app. They keep track of details such as their level, current XP, and currency points that they have, but also statistics and specific user settings they would like to have loaded back-up when they return to the app.

The statistics of the app for the current implementation solely include step counting. This statistic is recorded on a day-by-day basis and is sent to the server to store in the database. Each row is linked to a single day for each user and contains the member ID its related to, the date at the time of the recording and the number of steps for that date in a table called UserActivities. Table 2 below shows a snippet of each row for two users along with the data contained within each row.

Table 2 – Snippet of the list of data within the UserActivities table.

|  |  |  |
| --- | --- | --- |
| MemberId | Date | Steps |
| 17 | 2022-03-23 00:00:00.0000000 | 5435 |
| 16 | 2022-03-23 00:00:00.0000000 | 8295 |
| 16 | 2022-03-24 00:00:00.0000000 | 0 |
| 16 | 2022-03-25 00:00:00.0000000 | 10622 |
| 16 | 2022-03-22 00:00:00.0000000 | 13256 |
| 16 | 2022-03-26 00:00:00.0000000 | 6841 |
| 16 | 2022-03-27 00:00:00.0000000 | 12577 |

From the above data, I can extrapolate the average steps done in weeks and months for a user and present it in on a graph through the user interface on the client application as seen in the design of my prototype shown in [Section 4.3.2](#_Showing_lots_of) in Figure 16 and described thereafter.

Along with that, each user has a row dedicated to their entry within the Members table to store their user settings in a table called UserSettings. It currently holds only the maximum and minimum quest distance in kilometres the user would like to have to tailor their walks to a time and amount of effort they are comfortable with. Table 3 shows a snippet of the settings table, with each row tailored to each user. The default values are 1 kilometre for the maximum and 0.5 kilometres for the minimum unless the user has modified those values themselves.

Table 3 – Snippet of rows dedicated to each setting within the UserSettings table.

|  |  |  |
| --- | --- | --- |
| ID | QuestMaxDistance | QuestMinDistance |
| 2 | 0.7 | 0.3 |
| 3 | 0.4 | 0.05 |
| 4 | 2 | 0.6 |
| 5 | 1 | 0.5 |
| 6 | 1 | 0.5 |

Both the UserActivities table and the UserSettings table relation to the Members table can be seen in [Section 4.2.2](#_Database_Design) where their columns are represented the same as what is in the actual implementation of the diagram shown in Figure 12.

All the precedingly discussed systems in the server are interacted with through endpoints on the server. A full list of endpoints and their URI’s (uniform resource identifiers) are shown and listed in their individual controllers along with their HTTP methods on the far left in the snippet within Appendix 6.

## Client Application

While most of the core computation resides within the server, the client application still needs to put in some effort in processing the data from the server to present data that is interpretable and useful to the user and cater for the user experience which includes the user interface design. It isn’t the prettiest app, but it is a functional one that I have fine-tuned to give the most optimal user experience I could. The user experience plays a large part in what keeps the user engaged with the app, and the app is the main contributor to the success of the project both as a university project and in the commercial world.

### Using Retrofit to communicate with my web API

Retrofit is a package available from GitHub that provides an easy-to-use HTTP client to communicate with outside systems hosted on the internet that is also compatible with Android Studio. This is the only package I use to communicate to my web API hosted on the web that has many in-built methods to support the sending of serialised data and receival of packets of data to be serialised into objects within Java to use throughout my app. It is used because the Android Studio SDK and the Java development kit (JDK) from my findings, does not have built-in support for sending or receiving data from the web. Listing 6 and Listing 7 both show the ease of using the Retrofit library to setup a request to the endpoint on a server and sending a request across the web respectively using a function to attempt to login the user called attemptLogin.

@POST("api/login")  
Call<JsonElement> attemptLogin(@Body LoginDetails requestBody);

Listing 6 – [ILoginApi.java] Setup for a request to the server attempting to log in a user (Java).

Call<JsonElement> call = RetrofitClient.*getInstance*()  
 .getLoginApi()  
 .attemptLogin(loginDetails);

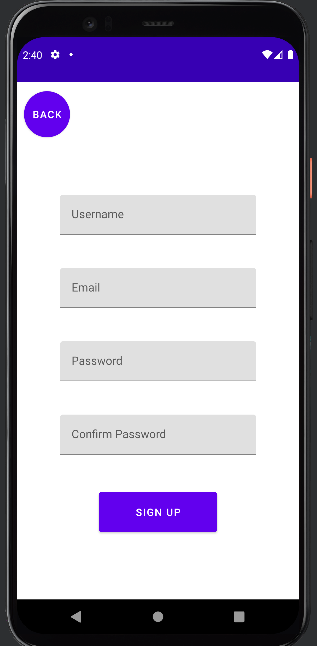
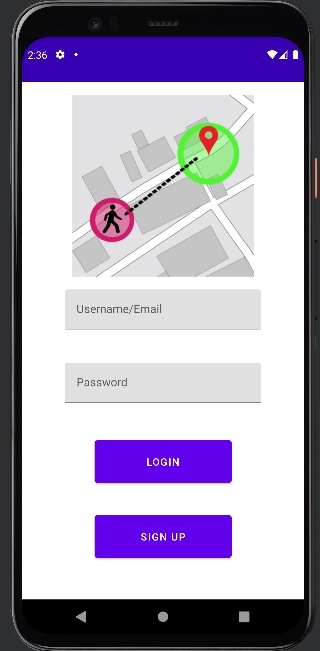
Listing 7 – [LoginExecutor.java] Sending the request to the server with the login details as the parameter (Java).

### Login and Sign-up Screen

The login screen is quite basic, providing a field to enter your username or email as an alternative and a password field with the logo of the app occupying the top half of the screen. The sign-up screen is also very rudimentary, providing fields to input data from the user and a back button to return the user back to the login screen. The design of both these screens can be seen in Figure 22.

While the designs are basic, the code behind the sign-up and login has a deeper level of complexity, validating each field appropriately to make sure that the input from the user is to the standard the system requires. For instance, the username cannot be too small or too big, the email field must be in the appropriate format by following an example structure such as “xxxxx@yyyy.com” and the password must have at least 8 characters, a digit, and a symbol. All these requirements are shown upon a user exiting the field after typing invalid text on the sign-up screen as shown in Figure 23.

Figure 22 – Design of the login screen shown on the left and sign -up screen on the right.



Similarly, for the login screen, if the user has typed in credentials that do not exist within the database, it would mean they do not have an account and an appropriate system error message would be shown. However, if they typed in a username or email that is found within the database but the wrong password, it will specify that the password is wrong and not that the account doesn’t exist. If the user has typed in completely incorrect credentials that fail validation, there is also a specific error message for that. I did not find it apt to place the errors under the fields as seen in Figure 23 for the lack of consistency between the different errors that could occur. Instead, I used the in-built Android SDK feature called a “Toast” which renders a small message that appears at the bottom of the screen for a period. Figure 24 shows the result of all three scenarios described, with the top message associated with failed validation, the middle message associated with an incorrect password, and the bottom message associated with credentials passing the validation but not linked with any account within the database.

Figure 23 – Validation on each field appropriate to the meaning of each field in the sign-up screen.

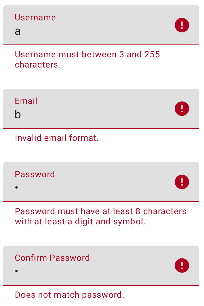
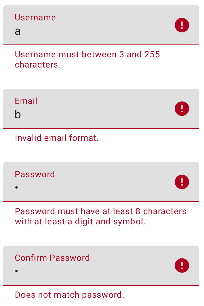


Figure 24 – Variety of login error messages that exist for specific situations.



My aim with each of my screens is to show mostly required information without bloating the screen with too much data, handle errors as well as conveying their meaning to the user if the need arises and striving for a level of consistency. Following guidelines for Human Computer Interactions (HCI) such as Nielsen’s 10 heuristic principles (discussed in [Section 4.3.1](#_Maintaining_consistency) and onwards) aids the user experience and reduces confusion by providing a simple layout that covers human error with easy-to-understand system messages. The login and sign-up screen involve the most user input required from the entire app, which made the screens proceeding the login section a lot easier to implement as there was less that could go wrong from user input thereafter.

### Quest/Main Screen

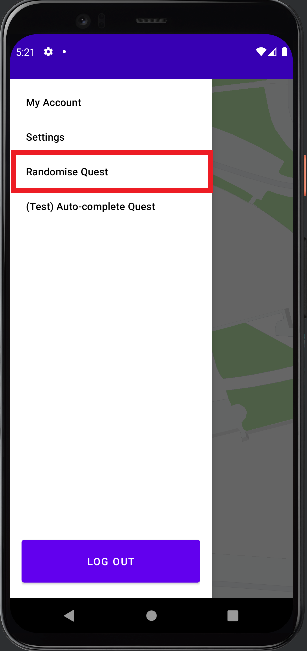
A screenshot of a cell phone

Description automatically generated with medium confidenceUpon logging in the user will be shown a screen like that seen in Figure 25, it shows a single button in the top left that expands the menu and their current location covered by a semi-transparent red circle with a 10-meter radius on the map, just like the logo.

Figure 25 – Screen shown upon logging into the app.

This screens functionality is simple, it displays their current location as well as the quest location in a semi-transparent green circle when they have an ongoing quest. When the user clicks on the navigation menu to get a quest, the screen will zoom the map in or out to keep both their current location and the quest location in frame. This subtle feature implies to the user that they should head a certain direction in their walk to get to the quest location. A demonstration of this is seen in Figure 26, with the image on the left showing the menu item that starts the process of getting a quest and the image on the right placing the user’s current location and their quest location in frame automatically.

Figure 26 – Process of starting a quest and showing the UI change when a quest has been started.



Graphical user interface, text

Description automatically generatedTo make clear that the user has completed the quest, the user will receive a notification informing them that they have reached their quest location. They would then receive updated account details that add the reward from the quest, with the account details shown within the Account screen. Figure 27 shows the contents of the notification and how its presented to the user.

Figure 27 – Notification upon completion of a quest

### Account Screen

Chart, bar chart

Description automatically generatedThe account screen contains all details remotely relating to the user. It shows their username, level, experience points and currency points procured since the time the account was created at the top of the screen. A graph showing the step count per day in the week by default is shown further down the screen. Figure 28 shows the user interface in detail.

Figure 28 – Full view of the user interface within the Account screen

As seen above, the layout is centred and depicts the most pertinent information to the user at the top, becoming less relevant as the user looks further down the screen. The experience points are represented in two ways, one is in a fractional format and the other is a progress bar that shows the percentage of the way towards the next level. Following the principle of keeping the UI simple, for users that want to know how close they are to the next level, a percentage-based approach is much more understandable than a fraction format, but for the subsection of users that are curious about their exact experience points, the fractional format exists to fulfil those needs.

#### Step-counter

The step counter relies on the user’s device having a step-counter component built-in. I considered also including an accelerometer implementation in case a user’s device does not have a step-counter built into their device, but I was nearing the end of my sprint and did not find it worth the time to implement as it would need too much fine tuning. I did not trust that the end implementation of this feature to be as accurate as I would have liked by the time of submission, so as a result it was left out entirely.

The step-counter works by using a foreground service that will be running in the background if the device while the app is on or off. It is designed to start when booting the device or when the user logs into the app if the app was freshly installed. It would record the steps of the user and refresh the Account screen’s UI every 5 seconds with the updated step-count. The service persists while their device is in idle mode or doze mode, which is a state any Android phone following Android version 6.0 (otherwise known as Android Marshmallow) will transition to if their device’s screen is off and is left alone for at least an hour. This mode exists to preserve battery but requires the service to be running as it takes the steps done by the end of the day and allows for it to be sent to the server when the user next views the Account screen to record their steps and show it on the graph.

#### Graph representation of the step-count

The graph is the most sophisticated feature within the Account screen, it shows the steps done within the current week in the form of a bar chart by default. Yet this is not what the graph is limited to, all forms of data that relate to a specific set of time periods that can be viewed are:

* The current week
* The current month
* All weeks starting from the first day in the first week to the last day in the 52nd week, showing the average steps for each week.
* All months starting from the start of January to the end of December, showing the average steps per month within the year.

To add to this, these sets of data can be represented in three different formats being a:

* Bar chart
* Pie chart
* Line chart

This results in a total of 12 different permutations that the data can be arranged in to appease the users need to view the data in whatever form they choose. Further, if the user wishes to see the steps per day within a week or month that is not the current week, they simply need to tap on the segment of the graph that represents that week or month which will then load a subgraph showing the step-count per day of a particular week or month of their choosing. This sums up all the permutations of the step-count data that can be viewed to an estimated 76 permutations.

However, adjustments were needed to make the data interpretable to the user depending on the chart. For instance, it would be incredibly difficult to show all 52 weeks within one frame on all the charts and for each of the 52 segments of data to be easily interpretable by the user, but for the bar and line chart this was easily fixed by enabling the graphs to side scroll and limiting the screen to show up to 7 bars (or data points for a line chart) at a time. This would allow the user to scroll horizontally and view more data along the x-axis of the graph.

For the pie chart though, 52 different entries occupying the small area reserved to render the pie chart in a single frame was not an easy task to accomplish but a partial solution was achieved with a small amount of mathematics. I summed up all the step-counts within the days that the set of data contained for a specific time-period that the user had loaded and made sure to only include entries into the pie chart that were at least 4% of the cumulative total steps recorded from the entirety of that dataset. This would come at the cost of reducing the amount of entries on the pie chart but would present a readable chart that would show the top days within the week or weeks and months within the year for the user. After much testing, I found the value of 4% provided a good balance between the reduction of the data and readability. An example of the result for the dataset containing current week’s data for all graphs is shown in Figure 29 while Figure 30 shows an example of handling the extremes of multiple entries (with the line and bar chart partially scrolled for evidence of horizontal scrolling).

Figure 29 – Current week data represented on all graphical options within the Account screen.

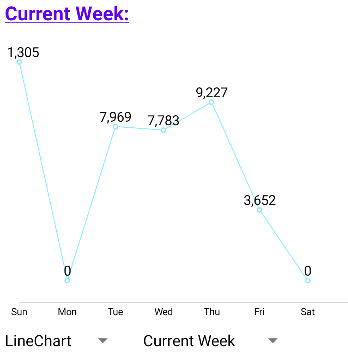
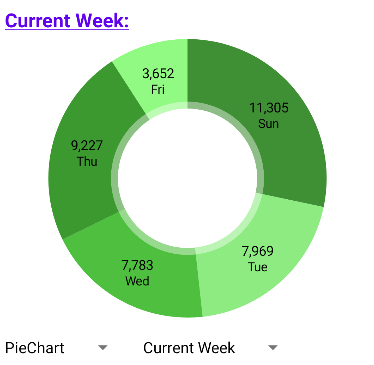
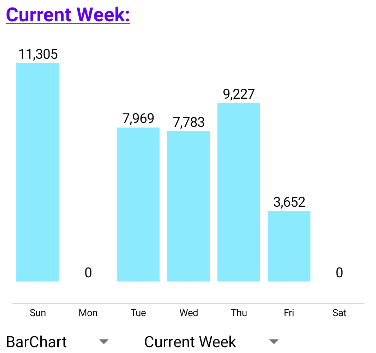
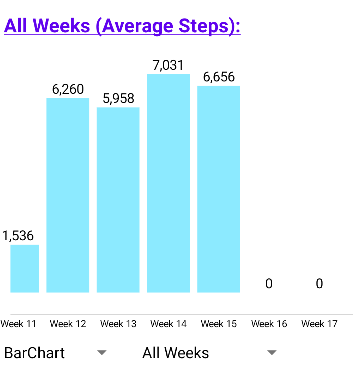
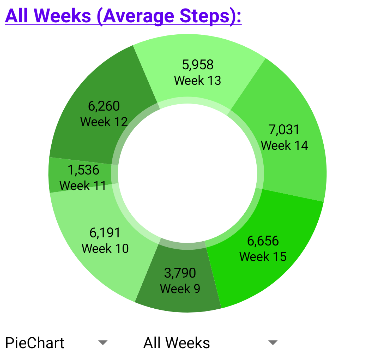
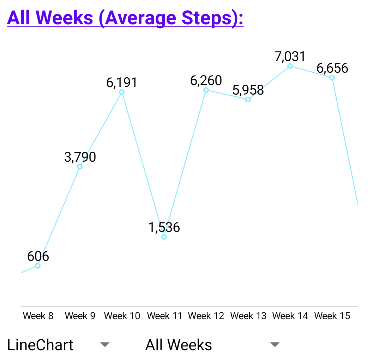


Figure 30 – Average steps from all 52 weeks on all graphical options within the Account screen to show handling of larger sets of data.



## Summary

Most of the implementation details that were most impactful to my project have been discussed with explanations behind the design decisions made in the final implementation. Nearly all the constituents that form the project’s implementation have at least been mentioned and at most have had a section dedicated to them. The most critical and well-developed features have been covered at large with sparse use of technical jargon and without providing too much code to not dwindle on the specifics, rather to generalise the many hundreds of lines of code they encompass. Lastly, clarification on the usage of HCI guidelines were mentioned on specific parts of the UI with reasoning as to how I have achieved them and how it is useful to the end user of my app. As a sidenote, the entire client application’s architecture is shown in a Class Diagram in Appendix 7, with intricate detailing of all the fields, properties and methods used in each class.

# Investigation

## Introduction

Software testing is the method of ensuring that a product meets the expected requirements with minimal to no defects. It is used to identify and enumerate any gaps or errors within the product that contrast with what is expected from the product (Hamilton, 2022). I will cover how I have personally tested the app as well as details of how the app was tested by the target demographic of my app to better understand how the app is used by others without insider knowledge of how the app functions internally. Additionally, I will detail how software was used to test my server program during its development.

## Functional Testing

During development, I utilised two sets of tests after each iteration of prototypes for my app were complete. The first test was Smoke testing, and it was used to make sure that the app would build successfully with its critical functionality working correctly after each iteration. The second test was Sanity testing which was used to validate if new features were functioning as intended or if a patch for a bug has fixed a bug. Both types of testing assured that the app was working as intended in its entirety, but it is completely dependent on the time spent testing the app itself.

My code is very modular throughout both my server and client application and as a result, I can isolate and identify issues with absolute certainty on which module was causing such issues. This quickened the process of testing and finding results as well as fixing the results. Regardless of this advantage though, ample time is still needed to properly test both my server and client; just because the process is faster does not mean I should test my artefacts less, but it means I can dedicate that saved time to conduct more rigorous testing that include more testing strategies.

The tests carried out were manually done by reproducing steps taken to validate if a feature was working as expected. I would try different combinations of testing with coupled modules such as the effect of changing the settings in the settings screen before and after I save to see if it affects the acquisition of a quest appropriately. This would be opposed to logging in and getting a quest immediately, branches of execution that affects the output of a feature are individually tested at the time of their creation using Sanity testing, or if it is part of a critical feature it will be covered in each Smoke test. Table 4 shows a small sample of tests conducted on the Sign-up screen a few iterations ago. The tests that have failed have since passed.

Table 4 – Sample conducted Sign-up screen tests from an earlier iteration of the project

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected Result | Actual Result | Pass/Fail |
| Type a less than 4-character username. | Error message showing criteria for username is shown | Error message showing criteria for username is shown | Pass |
| Type more than 255 characters for the username | Error message showing criteria for username is shown | Error message showing criteria for username is shown | Pass |
| Tap username text field then exit field | Error message is not shown | Error message is not shown | Pass |
| Type a username that meets criteria | Error message is not shown | Error message is not shown | Pass |
| Type less than 8 characters for password. | Error message showing criteria for password is shown | Error message showing criteria for password is shown | Pass |
| Type minimum number of characters for password without a number | Error message showing criteria for password is shown | Error message showing criteria for password is shown | Pass |
| Type minimum number of characters for password with a number without a symbol | Error message showing criteria for password is shown | Error message is not shown | Fail |
| Type minimum number of characters for password without a number but with a symbol | Error message showing criteria for password is shown | Error message is not shown | Fail |
| Tap password text field then exit field | Error message is not shown | Error message is not shown | Pass |
| Type a password that meets criteria | Error message is not shown | Error message is not shown | Pass |
| Tap confirm password text field then exit field | Error message is not shown | Error message showing criteria for confirm password is shown | Fail |

The failed tests in Table 4 for the password field that contained a number or a symbol character had reached those outcomes as a result of a faulty regular expression I constructed which has since been amended. The other failed confirm password test had reached that outcome because of checking for errors upon exiting a field. Since its own error was dependent on another field being empty, namely the password field, it would assume the user had typed a password prior. This functionality has since changed to make sure there is text in the password field before it attempts any form of validation.

Following from the above example, each finalised iteration post-testing phase that builds and passes validation is then added to my artefact specific GitHub repositories if I consider it a stable build. Any refactored code that is unstable or any local copies that are unstable for any reason from that point forward can be compared or replaced by the previous stable version of my artefacts stored on GitHub.

More examples of functional testing I have conducted can be seen in Appendix 8 where I have used a software called “Postman” to test a few endpoints within my server to check if their actual return values are the same as the expected return values.

## Non-Functional Testing

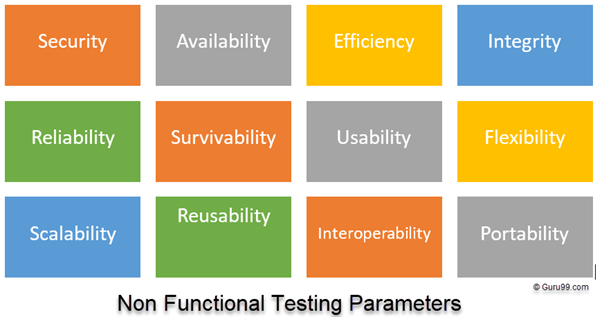
Non-functional testing are tests that care more for the structure of code in terms of performance, reliability, reusability, and many other parameters that measure the readiness of a system in ways that functional testing cannot (Hamilton, 2022). Within my artefacts, every refactorization done between iterations of my stable builds is due to at least one of the parameters shown in Figure 31.

Figure 31 – A list of parameters associated with non-functional testing presented by Hamilton (2022)

An attribute of my code that I focused quite a lot on was reusability and I have refactored many parts of my code since the initial programming of each feature to be reusable for other features that may be included in the system. An example of which is the way I handled asynchronous tasks being handled in the background of my app while the user was busy using the app itself; this specific refactorization also considered the future errors that may occur with reliability testing as the object used was considered deprecated at the time of development. Listing 8 contains code from an early prototype of my app used to configure the asynchronous tasks that I am referring to while Listing 9 shows the refactorization of this code to suit both the reusability and reliability of this code.

private class AttemptLogin extends AsyncTask<LoginDetails, Void, UserSettingsResponse>

Listing 8 – [LoginExecutor.java] Class inheriting from a deprecated object before refactoring (Java)

private class AttemptLogin implements Callable<UserSettingsResponse>

Listing 9 – [LoginExecutor.java] Class inheriting from refactored object to satisfy reusability and reliability tests (Java)

Like the above, numerous other refactors have taken place due to failing Survivability tests which has resulted in what is seen and discussed in [Section 4.3.4](#_UI_Error_Handling) as well as failing Usability tests which culminated into the final product seen in the UI that is depicted in Figure 23 within [Section 5.3.2](#_Login_and_Sign-up), just to name a few.

## User Testing

One of the most important ways to test my project is to gain feedback from a user that is not associated with the development of the project. The reason as to why this is extremely useful is because they might consider branches of execution that I previously haven’t thought of while manually testing the project in the real world myself. Untested branches of execution may yield bugs which could be minor or major depending on the feature affected. I asked a friend to try the app on their device and to give it a try at some point and to let me know how it feels and any problems they have encountered while using it. I have also noted that not much may come to fruition giving the app to a friend as they already had some amount of association to me, this may impose a bias on the feedback they give back, but any feedback is better than none.

Almost immediately I got results back, I provided them a release version rather than the debug version which wasn’t tested. For example, I didn’t realise that the release version of the app needed its own Google API key to communicate with and could not use the debug version’s API key, this was easy to solve but I would not have known if they didn’t use the release version. To keep track of the bugs that others would find, I updated my Trello page to include details about the bugs, the steps taken to get the bug and the result of following those steps. Each card has a comment that compactly describes the progress made so far and has labels on the card to show how critical an issue is. The label colours associated with a bug card shows the severity of a bug and an additional label is given if the bug is solved. These updates to the Trello page are shown in Appendix 9 and a snippet of a card with this update is shown in Figure 32 below.

Graphical user interface, text, application

Description automatically generated

Figure 32 – Snippet of a bug card with user comments on steps taken surmised with progress shown in the comments of the card.

## Summary

I have shown how functional and non-functional tests were carried out as well as the importance of user testing when developing software. Following the explained testing strategies have led to stable releases of each version of my application. However, I cannot say that it is completely stable given that each individual testing strategy tests a specific part of my pieces of software. It may fail some other metric I have not tested thus far using my testing strategies. Given my routine programmatic and manual tests on each component within both pieces of software I have created for this project has yielded expected results within the final iteration, I believe the testing strategies I have chosen has sufficed for the duration of the project.

# Evaluation, Conclusions and Future Work

## Project Objectives

I have made a function app and server with all the critically needed features in place and extra features in place that are fully developed and tested. Notably, I have completed the “Must” and the “Should” features with almost all of the “Could” features and one or more “Would” features within the list of MoSCoW objective prioritisation shown in [Section 3.3](#_Requirements). All features that were partially implemented were left out of the project entirely and some of the features I wanted to create - I didn’t have a chance to because of issues with the development during the length of the project. That is not to say I am not proud of what has been achieved in the time-span working on this project, I am extremely happy that the project functions as I wished from the start and adequately satisfied on my progress with it to this point.

## Self-Evaluation

There were a few things when working on the project I found momentous and monolithic that I was in fear of under-performing due to the lack of knowledge surrounding concepts I have never covered in university, but I have a knack for solving problems that I get myself into. When I mentioned the specific objectives within the project proposal shown in Appendix 1, I remember the fear of making such a project as what is said in a few words as a requirement can translate to a month in development, it is easier said than done put simply. I thought I didn’t have what it took to make a project of what I was asking for in the window of time I was provided but what I have done that and more which shocks me. I didn’t just make an app that worked with an open-source ready-to-go API, but I created my own API to fulfil my needs with my own server. With both the server and database saved locally and hosted on my PC, being shown online on the internet, accessible from anywhere – that was a huge success and one of the many hurdles in my project I managed to get past.

When talking about development specifically, there were many concepts I had to learn and in so doing: I learned the limitations of external packages, the limitations of the development kit for Android Studio, Java and C# and within those, the intricacies of their usage when looking at documentation to create features I previously thought would be too difficult to create with my level of expertise at the start. I am by no means an expert in any of the concepts learned but none-the-less it is knowledge that I doubt I would have ever learned at university. Before attempting the project, I had no idea there were many testing strategies to help ensure my project works as expected, or that there were easier ways to get data from a database using a certain package called “EF Core” to query my database using C# and not SQL.

On Android Studio specifically, I had to learn many new concepts as I had the least amount of experience with it from all other aspects in my project. I learned how to handle asynchronous code, making screens using fragments to dynamically update components, using external UI packages with specific features and far more. I replaced a lot of my bad practices and followed guidelines to help with the structure of my code and future additions to my code.

However, I would be lying if I said I didn’t have issues with the project. For one, the text on the UI seen in the Android app is hardcoded in the UI rather than in a separate file for strings. This would mean extra effort would need to be made to localise my app to other regions for translative purposes if I intend to release this app to non-natively English-speaking countries. Another is when I spent far too long on the intricate details of my step-counter, the steps sometimes wouldn’t reset at the end of the day, or the service wouldn’t call a method at a specific time because of doze mode. I had many issues using my service that would be non-existent one day but would be apparent the next. Since it was based on time it was hard to debug and even harder to debug since it runs in a separate process to the main app and the bug only occurs when the device is in doze mode, meaning I need to wait an hour or force my phone into doze mode using external software to test. Even then I wasn’t getting good enough data to determine what the issue was, so eventually I had to refactor the entire service to work another way and it resets the step counter without issue now, but it did take at least 2 sprints for it to work as intended.

While I perceive myself as hardworking, it is better to be a smart worker than a hard worker if I want to be pragmatic. I should have prioritised my time more effectively, Trello and Sprints with Sprint Retrospectives helped the project management immensely and provided structure, but I didn’t handle the time as best as I could have. Setting myself soft-deadlines on features should have probably in place so I don’t get carried away with the small details and hard deadlines as a limit to how much I should work on features. Rather than giving a single feature 100% of my time each time is probably worse for my type of project, it was probably better to follow a minimum viable project (MVP) architecture and working on each feature for a set amount of time each sprint. For now, though, that is something I will have to consider when working on project such as this in the future.

## Project Evaluation

To present all the requirements I have accomplished within the set of requirements shown in [Section 3.3](#_Requirements), Table 5 was created detail all the requirements and their completion. The top of the table starts with the “Must” requirements and the bottom of the table ends with the “Would” requirements from the MoSCoW prioritisation.

Table 5 – List of requirements for the project and their completion

|  |  |
| --- | --- |
| **Requirement** | **Completed?** |
| Have a functioning app with a Google Maps implementation  that tracks the device’s current position. | Yes |
| Have at least a marker for which the user must travel towards (a quest). | Yes |
| Have a way for the user to randomise their next location to walk  towards via UI interaction | Yes |
| Allow for customisation for distances that the user can travel. | Yes |
| Check if the device location is at the quest location and create a  notification on the device to notify user of the completion of their quest. | Yes |
| Save data locally on device to keep track of information the app has  procured while in use that would be useful to the next launch of the app. | Yes |
| Create a server application to compute the most CPU intensive operations. | Yes |
| Create an API in which the server can be interacted with. | Yes |
| Host the server implementation to be accessible by the web. | Yes |
| Establish successful client-server communications between the  app and the server. | Yes |
| Defer all CPU intensive tasks running on the app to the server. | Yes |
| Add a radius on the maps implementation to allow user to walk  within a range of the quest objective. | Yes |
| Add a SQL database in which the server can communicate with and  store data. | Yes |
| Add a login system in which users create an account and login  via the app communicating to the server. | Yes |
| Save user specific details such as gamified elements including  experience points and currency. | Yes |
| Save details regarding the user’s current quest, experience points  and additional user specific details to be implemented on the users  record within the database | Yes |
| Track the users step count for the day. | Yes |
| Add a leader board for experience and/or points. | No |
| Track the users step count for every day. | Yes |
| Create a graphical representation for the steps covered per day,  week, or month. | Yes |
| Add in rewards to spend the user’s currency points. | No |
| Add in a friends list in which the user can communicate with  others who have signed up to the app. | No |
| Add in a chained journey mode in which the user can travel to  multiple points and gain more experience points for each subsequent journey attempted. | No |
| Allow users to share quests and attempt a group walk between  each other with bonus rewards. | No |

From the 24 listed requirements, 5 of the requirements have not been fulfilled. From the 5, 1 is from the “Could” list of requirements and the other 4 are from the “Would” list of requirements. From these, the one I wish I could have implemented the most is adding a friends list and being able to chat with others. This is because it was unique from the rest of the implementation which is to do with the questing and experience points. I believe I would have learned a lot from creating a social system within the project and the social aspect alone would make the app far more inviting for people wanting to walk in groups and complete a walk together but again, I did not have enough time to create these features despite the project management practices I have used.

## Applicability of Findings to the Commercial World

In a commercial environment, I believe my app would do extremely well. I am personally against microtransactions due to how they are applied to apps within the current decade but if further incentives are brought into the game that make it more immersive, a way to convert real world money into currency points could be made as one form of income. Another is placing advertisements when trying to access the Account screen but when play-testing the app itself, I could see how this could quickly become annoying and cause a player to quit my app entirely. Alternatively, a membership could be purchased to remove those advertisements or perhaps a one-time fee that is priced at a suitable average investment from the average paying user or less. My last suggestion would be to instead have a monthly paid membership to see statistics on your walk in the form of the graph represented and perhaps highlights of your year, month, week, or day also provided as text in the UI as illustrated in Appendix . These statistics could also show calories burned or other statistics related to questing by themselves or with friends.

There are many ways to monetise my app but if I were to ask any user what they would monetise, the response is likely to be “nothing” as no one likes to lose something to gain features that are already installed but inaccessible as a result of not paying.

## Conclusions

This project has allowed me to express my newly acquired knowledge from both my placement and all the knowledge I have acquired over the course of the project unrelated to the material taught at university. Given that many assignments deadlines were all around the project artefact submission date and not getting a single extension for any of them, I managed to complete all the work within such a small timeframe and still do extremely well with my project (in my opinion). I plan to at least revisit and refactor portions of my project I was unable to get around to along my journey of discovery late in the project and hopefully share it between likeminded friends with many more features.

## Future Work

First and foremost, while the user experience is catered for, user interface isn’t as appealing as I would like it to be. I was relying most on my knowledge of an assignment I completed in my second year for university. While I did exceptionally well on that assignment, it wasn’t the most visually appealing and better UI tools exist to aid me with the user interface. While there isn’t too much I would like to change, I believe the finished product could be dramatically improved. Many popular apps are not only easier to use but their UI mixes well with their great user experience so both can certainly be achieved.

Along with that, any hardcoded data in the UI such as themes and text to be placed in their own file for easier customisation of the colour schemes and more importantly, increasing the feasibility to implement localisation features for users that prefer to read the UI in other languages that are not English. These improvements are more in line with guidelines but in general, I am trying to follow the recommended guidelines for building an app that I have discovered across the development of the app. It increases accessibility and the userbase overall as more people that understand the app also means more potential users willing to sign-up and use the app.

For the server, I believe it needs to be cleaned up a little more with more structuralised responses on error codes and more modularisation. More adequate testing needs to be in place as well as Postman does not guarantee that all branches of execution within my code have been tested. There may come a time where the server runs into an error as a result of a bad internal state, so this is probably best handled by placing in unit tests to cover all angles of execution.

# References

Babich, N. (2020). *UX Design Process: Everything You Need to Know | Adobe XD Ideas*. https://xd.adobe.com/ideas/guides/ux-design-process-steps/

Bracey, K. (2018). *What Is Figma?* https://webdesign.tutsplus.com/articles/what-is-figma--cms-32272

Chaudron, M. R. V., Heijstek, W., & Nugroho, A. (2012). How effective is UML modeling?: An empirical perspective on costs and benefits. *Software and Systems Modeling*, *11*(4), 571–580. https://doi.org/10.1007/S10270-012-0278-4

*Client–server model - Wikipedia*. (2011). https://en.wikipedia.org/wiki/Client%E2%80%93server\_model

*Data protection: The Data Protection Act - GOV.UK*. (n.d.). Retrieved April 3, 2022, from https://www.gov.uk/data-protection

Denny, P. (2013). The Effect of Virtual Achievements on Student Engagement. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/2470654

Eickhoff, C., Harris, C. G., Srinivasan, P., & de Vries, A. P. (2012). *Quality through Flow and Immersion: Gamifying Crowdsourced Relevance Assessments*. http://www.geann.org

Epstein, J. A., & Harackiewicz, J. M. (2016). Winning is Not Enough: The Effects of Competition and Achievement Orientation on Intrinsic Interest: *Http://Dx.Doi.Org/10.1177/0146167292182003*, *18*(2), 128–138. https://doi.org/10.1177/0146167292182003

Fogg, B. J. (2009). *A Behavior Model for Persuasive Design*. www.bjfogg.com

Fujiki, Y., Kazakos, K., & Puri, C. (2008). NEAT-o-Games: Blending Physical Activity and Fun in the Daily Routine. *ACM Computers in Entertainment*, *6*(2). https://doi.org/10.1145/1371216.1371224

Gerber, E., & Carroll, M. (2011). The psychological experience of prototyping. *Design Studies*. https://doi.org/10.1016/j.destud.2011.06.005

Grobauer, B., Walloschek, T., & Stöcker, E. (2011). Understanding cloud computing vulnerabilities. *IEEE Security and Privacy*, *9*(2), 50–57. https://doi.org/10.1109/MSP.2010.115

Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does gamification work? - A literature review of empirical studies on gamification. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 3025–3034. https://doi.org/10.1109/HICSS.2014.377

Hamilton, T. (2022a). *What is Non Functional Testing? Types with Example*. Guru99. https://www.guru99.com/non-functional-testing.html

Hamilton, T. (2022b). *What is Software Testing? Definition, Basics & Types in Software Engineering*. Guru99. https://www.guru99.com/software-testing-introduction-importance.html

Haroon-Sulyman, S., & Shakirat Oluwatosin, H. (2014). Client-Server Model. *Journal of Computer Engineering*, *1*, 67–71. https://doi.org/10.9790/0661-16195771

*Homepage - Material Design*. (n.d.). Retrieved April 5, 2022, from https://material.io/

*How does No-IP works? | Geekboots*. (2019). Geekboots. https://www.geekboots.com/story/how-does-no-ip-works

Ivankov. A., (2020). *Thick Client vs. Thin Client: Advantages and Disadvantages - Profolus*. Retrieved October 20, 2021, from <https://www.profolus.com/topics/thick-client-vs-thin-client-advantages-and-disadvantages/>

Kanter, J. (1997). *Understanding Thin-Client/Server Computing Chapter 1: The Thin-Client/Server Computing Model*.

Khan, Z., Haider, Z., Ahmad Mumtaz, N., College, P. G., & Khan, S. (2011). *Sports Achievement Motivation and Sports Competition Anxiety: A Relationship Study*. *2*(4). www.iiste.org

Kumar Bhatia, P., & Jambheshwar, G. (2012). Impact of Agile Methodology on Software Development Process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE*, *2*(4). https://www.researchgate.net/publication/255707851

Lardinois, F. (2019). *Kotlin is now Google’s preferred language for Android app development | TechCrunch*. TechCrunch. https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/?guccounter=1&guce\_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce\_referrer\_sig=AQAAAHBx7TUyqRkxkn5Fskr9XqxlLC1Mubcb925KTk4FiEU8H6FbV4qe-002YbfN5xajWFakzzmTvKqlOwj200bjLdt8oXGdZQ-p24O9gzPGcjbPRr85JCNKOL1STtilWsbkdxr94Xmw4frn7tR4ZUJklzXl6KgFgjNAbedntKqAbkmJ

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M., & Brinkkemper, S. (2016). The Use and Effectiveness of User Stories in Practice. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9619*, 205–222. https://doi.org/10.1007/978-3-319-30282-9\_14

Maruping, L. M., Venkatesh, V., & Agarwal, R. (2009). A control theory perspective on agile methodology use and changing user requirements. *Information Systems Research*, *20*(3), 377–399. https://doi.org/10.1287/ISRE.1090.0238

Mccormick, M. (2012). *Waterfall vs. Agile Methodology*.

Nasar, J. L., & Troyer, D. (2013). Pedestrian injuries due to mobile phone use in public places. *Accident Analysis and Prevention*, *57*, 91–95. https://doi.org/10.1016/J.AAP.2013.03.021

Nielsen, J. (2020). *10 Usability Heuristics for User Interface Design*. Nielsen Norman Group. https://www.nngroup.com/articles/ten-usability-heuristics/

*Number of gamers worldwide 2021 | Statista*. (2021). https://www.statista.com/statistics/293304/number-video-gamers/

O’donovan, S., Gain, J., & Marais, P. (2013). *A Case Study in the Gamification of a University-level Games Development Course*. https://doi.org/10.1145/2513456.2513469

*Password Hashing Competition*. (2019). <https://www.password-hashing.net/>

Pellegrini, M., Ponzo, V. et al. (2020). Changes in Weight and Nutritional Habits in Adults with Obesity during the “Lockdown” Period Caused by the COVID-19 Virus Emergency. *Nutrients*, *12*(7), 2016. https://doi.org/10.3390/nu12072016

Peterson, R. (2022). *What is Normalization in DBMS (SQL)? 1NF, 2NF, 3NF, BCNF Database with Example*. Guru99. https://www.guru99.com/database-normalization.html#4

Popovic, K., & Hocenski, Ž. (2010). Cloud computing security issues and challenges. *Undefined*.

Pradeep, Dr., Sagar, D., & Deepak, G. (2012). *Challenging Issues and Limitations of Mobile Computing*.

Preziuso, M. (2015). *Password Hashing: PBKDF2, Scrypt, Bcrypt | by Michele Preziuso | Medium*. Medium. https://medium.com/@mpreziuso/password-hashing-pbkdf2-scrypt-bcrypt-1ef4bb9c19b3

Preziuso, M. (2019). *Password Hashing: Scrypt, Bcrypt and ARGON2 | by Michele Preziuso | Analytics Vidhya | Medium*. Medium. https://medium.com/analytics-vidhya/password-hashing-pbkdf2-scrypt-bcrypt-and-argon2-e25aaf41598e

*Prototyping | Usability.gov*. (n.d.). Retrieved November 28, 2021, from https://www.usability.gov/how-to-and-tools/methods/prototyping.html

Qi, H., & Gani, A. (2012). Research on mobile cloud computing: Review, trend and perspectives. *2012 2nd International Conference on Digital Information and Communication Technology and Its Applications, DICTAP 2012*, 195–202. https://doi.org/10.1109/DICTAP.2012.6215350

Rehkopf, M. (n.d.). *Sprints | Atlassian*. Atlassian. Retrieved April 14, 2022, from https://www.atlassian.com/agile/scrum/sprints

Rhoudunda, B. (2020). *MVC design pattern. why do developer use model view… | by Bryan Rhodunda | Medium*. Medium. https://medium.com/@rhodunda/mvc-design-pattern-fe76175a01de

Sim, W. (2016). *9-year-old boy killed by truck driver playing Pokemon Go in central Japan | The Straits Times*. The Straits Times. https://www.straitstimes.com/asia/east-asia/9-year-old-boy-killed-by-truck-driver-playing-pokemon-go-in-central-japan

Tang, W., Lee, J. H., Song, B., Islam, M. M., Na, S., & Huh, E. N. (2011). Multi-Platform Mobile Thin Client Architecture in Cloud Environment. *Procedia Environmental Sciences*, *11*(PART A), 499–504. https://doi.org/10.1016/J.PROENV.2011.12.079

*Thick Client vs. Thin Client: Advantages and Disadvantages - Profolus*. (n.d.). Retrieved October 20, 2021, from https://www.profolus.com/topics/thick-client-vs-thin-client-advantages-and-disadvantages/

Tidy, J., & Molloy, D. (2021). *Twitch confirms massive data breach - BBC News*. BBC. https://www.bbc.co.uk/news/technology-58817658

Turner, R., Ingold, D., Lane, J. A., Madachy, R., & Anderson, D. (2012). Effectiveness of kanban approaches in systems engineering within rapid response environments. *Procedia Computer Science*, *8*, 309–314. https://doi.org/10.1016/J.PROCS.2012.01.065

Walker, M., Takayama, L., & Landay, J. A. (2002). High-Fidelity or Low-Fidelity, Paper or Computer? Choosing Attributes when Testing Web Prototypes: *Http://Dx.Doi.Org/10.1177/154193120204600513*, *46*(5), 661–665. https://doi.org/10.1177/154193120204600513

*What is Trello? - Trello Help*. (n.d.). Retrieved April 3, 2022, from https://help.trello.com/article/708-what-is-trello

Xu, L. (2017). Context aware traffic identification kit (TriCK) for network selection in future HetNets/5G networks. *2017 International Symposium on Networks, Computers and Communications, ISNCC 2017*. https://doi.org/10.1109/ISNCC.2017.8072034

*Young woman suffers multiple injuries during Pokemon hunt in Chesterfield | Derbyshire Times*. (2016). Derbyshire Times. <https://www.derbyshiretimes.co.uk/news/young-woman-suffers-multiple-injuries-during-pokemon-hunt-chesterfield-57350>

Zeedyk, M., Wallace, L et al. (2001). Children and road safety: Increasing knowledge does not improve behaviour. *British Journal Of Educational Psychology*, *71*(4), 573-594. doi: 10.1348/000709901158686

# Appendix 1

## Project Proposal

**Student Name: Muhammad Katib Hussain**

**Course: Software Engineering**

**Project Title: Walking Adventure App**

## Project Context

Due to the pandemic, I as well as many people have stayed home for a very long time in isolation. I used to go on walks with a friend quite often but that slowly came to a stop. Pellegrini et al. (2020) suggested from their results that either most individuals had reduced physical activity or had more unhealthy eating habits during the Coronavirus pandemic (p. 4). Both played a part for me when I was in isolation but when the restrictions were eased off, we started to walk again but lost the passion for it. We do it mostly now to lose a bit of weight, but I wanted to bring a bit of passion back. So, I thought of making a walking game app to make walks a bit more exciting. It is evident from the research Fujiki (2008) did that games such as what this project is intended for had increased the time and engagement of mild aerobic activities for players (p. 19). So rather than making a generic analytical app that just showed statistics of your walk, I wish to combine that with a game to make the walking experience a bit more interesting.

The basic premise is that the app will choose a location for you at random within a set distance that the user defines. This app can be used for fun between friends but will also hopefully incorporate statistics for the user, such as step count, averages of the step count per week/month and comparisons between friends. This could hopefully evolve into geocaching and other additions to make the walk that much more adventurous.

## Specific Objectives

* To set a location on an aerial view of Google Maps and have the app automatically detect if you are within the proximity of the location
* Utilising the pedometer within appropriate devices to determine the step-count of users
* To help the consumer become more physically active.
* To give the user lots of customisation option to tailor the app to them or to their friends.
* To make the app a thin client with all processing done on a central server

## Potential Ethical or Legal Issues

The intention of the app is for the user to take a journey to a location in hopes of getting in more physical activity. However, depending on where the user is located, the app may ask for them to walk across private property to get to a location, possibly walking places unknown to the user or impossible to reach the terrain becoming unsuitable for walking. It has the potential for someone to get in harms way due to the app telling them to get to a location for an arbitrary reward. This has already happened with other apps that involve walking such as Pokémon GO as DT (2016) found that a young woman had been playing the game and slipped causing multiple injuries but thankfully lived. Some were less fortunate and became fatal; An example being from

Sim (2016) who reported in an article that a child had been killed by a truck driver who was playing Pokémon GO on his phone while driving at the time of the incident.

Children are also not the target audience for this type of project. If children were to use the app without guidance, issues could become fatal when the location is across a road. An argument could be made that if a child follows road safety, then they should be relatively safe. However, Zeedyk (2001) suggests from his results that even children that were given appropriate training on road safety had performed no better than the children who didn’t in a realistic traffic environment (p. 19). For that reason alone, children should not be permitted to use the app without adult guidance.

Data security is also a large issue, this application will handle sensitive information about a user such as their name, username and password which could mean trouble if there is a data breach. That was the case recently as Tidy (2021) reported on the data breach that occurred to and organization called Twitch in which 100 GB worth of data was leaked. The best the customers could do is to change their passwords but the company takes full blame for the incident and could be fined.

## Resources

* Mobile phone with a pedometer running Android
* Android Studio
* Google Cloud Platform
* GearHost (SQL Database Hosting Service)

## Potential Commercial Considerations - Estimated costs and benefits

This product could be commercially successful as:

* This project is tailored to those who wish to compete with their friends and have fun while being physically active.
* This project would provide lots of customisation options to the user to create rewards between friends rather than a streamlined reward system as other games do.
* Geocaching and other features such as finding 3 mystery locations by showing only images of the location and no indicator on the map for where the location is and getting a reward for it would make the app far more interesting and engaging

The minimum a user needs is a mobile phone with a pedometer and mobile data. Ideally, most the processing of requests (such as requesting to start a journey to a new random location) would be done on the server rather than using the processing power on their phones. This server would ideally be hosted at home. Storage on the other hand would be handled by Google Cloud Platform which would come at a monthly fee which is around £2.50 for 100GB worth of storage (mostly for images). Storing user data would be free as GearHost provides a free SQL Database with good enough performance but could be £5 a month for better SQL Database performance and additional perks.

## Proposed Approach

* Getting an idea for the UI (1 – 3 days)
* Adding a marker within a radius determined by the user on Google Maps in the app (1 - 2 weeks)
* Adding experience and reward features (2 – 3 weeks)
* Adding profile features (1 week)
* Adding online features such as friends list and co-operative play (utilising SQL and cloud storage) (3 – 4 weeks)

# Appendix 2

## Technical Plan

**Name: Muhammad Katib Hussain**

**Course:** **Software Engineering**

**Supervisory Team: Nick Mitchell**

## Title

Adventure Walk

## Summary

I will be creating an app that is focused on getting a user to get some exercise done while making the activity fun by getting to a location shown on the integrated map provided by the app within the user’s local area. Research has shown that game elements combined with aerobic activities increases time and engagement of aerobic activities (Fujiki et al., 2008) which is why I decided to make a small game out of this app rather than solely showing analytics. Ideally this app will be a thin client, with a server processing requests and sending data back to the user to reduce the number of resources the app uses on their device. As opposed to a thick client, a thin client is easier to maintain hardware and software wise, so they have a longer lifespan while also being more secure and is overall more cost effective and efficient (Ivankov. A., 2020). This project has a lot of room for features to be added to make the experience more immersive but due to time constraints, I will prioritise some features over others, so I believe the Agile Methodology is most suited for this type of project.

## Deliverables

* An Android Package file (APK) holding the contents of the complete app that would have full features accessible to devices running Android version 6.0 and above with a possible reduced number of functioning features on devices running any Android versions below version 6.0 at an absolute minimum of Android version 4.0.
* An executable file containing the code that gives the app its functionality and processes requests from clients. This would primarily be run on a Windows OS but could potentially runon Linux.

## Constraints

* Fixed project deadlines would mean I would have to carefully prioritise what features I would add. While I have some knowledge behind the topics I will be facing in this project I will have to do extensive research for the implementation of these features as there is a time limit for the project.
* Using a cloud-based database means there will be limitations depending on what plan I utilise. The most substantial issues when it comes to my project is possibly having:
  + Reduced length of time the server will remain active during the day
  + A low number of connections to the database.
  + A low amount of storage size on the database.

## Key Problems

* My requirements to make the project function as I would like would be having a:
* Server-side executable with as much up-time as possible to process requests
* Thin client as an app to send requests and receive responses from the main server
* API designed specifically to make these transactions between the server and the app
* SQL Database which will store information and give information when needed.

While I do have relatively basic knowledge on these topics, the list above is just the backbone of the project. Implementation of the project’s features are entirely separate; the backbone of the project needs to be rigid before I can add features that encapsulates the project idea. I would have to be extremely careful with the design of the backbone such as planning the database model, having a stable and easy to use API, allowing easy expansion for both the client app and the server application and more. This is to ensure that the user-experience is as smooth as possible by factoring in the performance and efficiency of the backbone, which will not only make it better for the user but also allow me to introduce features with less of a worry about performance of the features overall.

Data Security in the present day has never been more important and even more so in the future, I plan to hold data relating to the user’s profile which is considered sensitive and therefore I need to make sure that the user’s data is not compromised.

## System and Work Outline

I will have a Kanban board which will manage all my tasks, categorising a list of my features according to MoSCoW prioritisation using an online service called Trello. Kanban boards are an extremely effective way of maximising value from each task (Turner et al., 2012) which is why for a solo project, I need to make sure I make the best decisions within the time allotted for this project. Planning is crucial in any project and MoSCoW will be especially useful as it allows me to plan my actions while factoring in the time it would take to complete the said task, as well as the complexity of the task. I do not have the luxury of spending as much time as I would want on a feature, so this method strikes a balance between planning, time, and complexity.

Design is likely to be the most time-consuming step of the project. When it comes to the database, database modelling is not easy to get right, if not correctly designed in accordance with the features that I plan to add and features currently unthought of, missing a step could me re-designing the entire schema and their relation to each part of the database. However, I plan to use UML to make a design for the database before I start constructing the database to make sure I make the correct decisions when creating the database model thus ensuring that the right system is being built (Chaudron et al., 2012). For the server, I plan to use C# as the programming language and creating a very modular system to make it easy to expand the project. UI-planning for the client app is also crucial for the end-user, I plan on using Nielsen’s usability heuristics to create at least a decent user experience, but this will also take a lot of on paper-prototyping and iterations on the interface design.

Implementation will also be a challenge but with design as a guideline, it shouldn’t be too bad. I plan to use ASP.Net to create my API on the server application and while it won’t be easy to begin with, once the API works for one endpoint I can replicate and reuse that formula to make numerous other endpoints. When it comes to the main features of the app, I have already thought of how I would calculate the distance from the user’s current location and the objective marker. The Haversine formula will be used for that and in conjunction with Google Maps API, features should be relatively simple to implement but the complexity will come from how they are built upon. Analytical features such as using the pedometer in the user’s device to find their step count and average over a period of time are innately complex to implement due to my unfamiliarity with using components of a device as data in an app so lots of research would be needed for different features I plan to add and not just the backbone of the project.

## Project Activities

The below Gantt chart is a rough outline on how I will proceed with the project with time estimates on how long I plan to take on each step along the development of the project.

**Timeline

Description automatically generated**

## Risk Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Severity | Likelihood | Action |
| Losing Code | High | Low | Work will also be stored on cloud using GitHub as a backup. |
| Code maintenance | Medium | Medium | To reduce maintenance time, I will be using software principles like D.R.Y and S.O.L.I.D and design patterns to make code as manageable and easy to deal with as possible. |
| Database uptime | High | Low | For working purposes, I will keep a backup database that will allow me to test the app locally but also look further for services with the best uptime but otherwise I would need to find a better provider. |
| Feature changes | Medium | High | Depending on how large the change is, this could be an easy change or something that lasts a few days. If I do not have time I will make sure the feature is at minimum functional or taken out completely. |
| Not enough time for adding features | Low | High | MoSCoW prioritisation will make sure I will develop taking into account the time and complexity of the feature, by this point I should have a few features already developed so not too big of an issue if this comes to be. |

## Options

When it comes to the Target Environment, I initially thought of just having the client app only and having the only artefact as the APK that comes from it. Everything packed into one application reduces expenses for me and adds simplicity too since I can just build all I need into one environment. However, this has a cost on the user’s device as well, it will drain battery rapidly, performance would be unstable as depending on the device, the processing power needed may take different lengths of time on each device depending on their specs and overall, this would make the user experience worse.

When it comes to designing the API, there are lots of options out there, but I chose ASP.Net as I have some familiarity with it. To add to that, API’s can be made in most languages and so it is really down to preference but the disadvantage of using ASP.Net is that it isn’t the fastest option I could go with. React is also a good language to create API’s quickly, easily and has a lot of options for expansion to the API. However, I have never used React and I’d rather be safe than sorry if I run into problems with it that cause me to rethink my approach later into development.

## Potential Ethical or Legal Issues

The intention of the app is for the user to take a journey to a location in hopes of getting in more physical activity. However, depending on where the user is located, the app may ask for them to walk across private property to get to a location, possibly walking places unknown to the user or impossible to reach the terrain becoming unsuitable for walking. It has the potential for someone to get in harm’s way due to the app telling them to get to a location for an arbitrary reward. This has already happened with other apps that involve walking such as Pokémon GO as DT (2016) found that a young woman had been playing the game and slipped causing multiple injuries but thankfully lived. Some were less fortunate and became fatal; An example being from Sim (2016) who reported in an article that a child had been killed by a truck driver who was playing Pokémon GO on his phone while driving at the time of the incident.

Children are also not the target audience for this type of project. If children were to use the app without guidance, issues could become fatal when the location is across a road. An argument could be made that if a child follows road safety, then they should be relatively safe. However, Zeedyk (2001) suggests from his results that even children that were given appropriate training on road safety had performed no better than the children who didn’t in a realistic traffic environment (p. 19). For that reason alone, children should not be permitted to use the app without adult guidance.

Data security is also a large issue, this application will handle sensitive information about a user such as their name, username and password which could mean trouble if there is a data breach. That was the case recently as Tidy (2021) reported on the data breach that occurred to an organization called Twitch in which 100 GB worth of data was leaked. The best the customers could do is to change their passwords, but the company takes full blame for the incident and could be fined.

## Commercial Analysis

In terms of how I would make money from this, I would set up ads to play on a timer on specific screens that would not hinder the walking experience. For example, on the analytics screen for details about the walk, it would play an advert each time a user decides to go on that screen. This may not be the exact approach in the end but essentially the goal is to play an advert on occasions where the user is on the app during the downtime of their walk which would be a 2-minute break on the walk.

The minimum a user needs is a mobile phone with a pedometer and mobile data. Ideally, most the processing of requests (such as requesting to start a journey to a new random location) would be done on the server rather than using the processing power on their phones. I am trying my best to cut expenses but at the same time having a good/decent user experience. This server would ideally be hosted at home on my PC, but if needs be I will use an online service to host my web API to be used from the cloud such as Heroku which would be enough for project on the Hobbyist plan or Azure . The SQL database should come at a low cost to none, there are many free SQL databases hosted online that would be perfect for the purposes of the project and if needs be I could get a more performant and remote solution from the many services on the web like AWS, Google Platform or Azure.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Factor name | Description | Is this a costor a benefit | EstimatedAmount | Estimateof whenpaid |
| IDE Open-Source | The IDE’s I will be using are Android Studio and Visual Studio, both of these are open-source software | Benefit | £0 |  |
| Heroku Hosting Service | If I deploy my server application to Heroku, this will mean 100% uptime due to the hobbyist plan as well as a service-supplied domain to access my project. Again though, since I know a free alternative that I am hosting | Cost | £7 | Month |
| Azure Hosting Service | If I deploy my server application to Azure, I can have multiple instances, unlimited apps and custom domains with dedicated computing rather than a shared computing pool. But this would be a cost as the project itself can be more lightweight as there will be near to no traffic on my application during the time I am working on the project. | Cost | £9.71 | Month |
| SQL Hosting Service | I may go for a paid SQL hosting service but as stated above, this will likely be free for the purposes of the project as many SQL hosts have a free plan for hobbyists which suffice for the project. | Benefit | £0 |  |

## Employability Contribution

For my final year project, I have developed a mobile application acting as a thin client for the purposes of aerobic activity and with competitive game features that interacts with a locally hosted server. The thin client is programmed in Java on Android Studio that contains features that makes creating app’s easy while still having a variety of options for more complex features by using Activities, Fragments, and various components to make each app unique. The server application handles the resource heavy processing by receiving requests via its API created with ASP.NET Core which communicated to the processing application designed in C# to create a result and send it back as a response to the user using JSON. I have had a lot of experience saving and using user data from the app using locally and service hosted SQL Database while also working with domain name service providers to make sure my Web API is accessible from the cloud. All the code is maintained using a version control service called GitHub to make sure all the work is backed up using repositories specific to the project.

# Appendix 3

## Project Trello Board

Graphical user interface, application, Teams

Description automatically generated

# Appendix 4

## Overview of the prototype

Graphical user interface, application

Description automatically generated

# Appendix 5

## Interactable prototype

<https://www.figma.com/proto/lHAAy3y3RTwvYzBm46SYqg/AdventureWalk-WireFrame?node-id=1%3A3&scaling=scale-down&page-id=0%3A1&starting-point-node-id=1%3A3&show-proto-sidebar=1>

# Appendix 6

## All endpoints on the server listed in their entirety

Graphical user interface, application, email

Description automatically generated

# Appendix 7

## Client application’s full architecture illustrated in a Class Diagram

<https://i.imgur.com/S8bDTGZ.png>

# Appendix 8

## Functional tests using Postman on my server

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

# Appendix 9

## Trello page updates with a dedicated bug testing section

Graphical user interface, application

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated