FITVENTURE

FINAL REPORT

Cardiff University

School of Computer Science & Informatics

CM3203 ONE SEMESTER INDIVIDUAL PROJECT

40 CREDITS

Karol Krajcir C1444273

Supervised by Dr. Wendy Ivins
Moderated by Professor Chris Jones

Abstract

There are lots of fitness applications on the market, but not a single one focuses purely on stimulating the intrinsic motivation of user to lead a long-lasting healthy and active lifestyle. Furthermore, they all assume that everybody has someone to work out or share their goals and progress with.

Even though the worldwide obesity rates are on the rise, fitness industry has seen a massive growth in the last few years and is expected to further increase in the near future. And the same applies for mobile device usage. Underpinned by research, FitVenture makes use of this ever-growing fitness market, increased popularity of group activity and mobile devices ubiquity to provide user with a set of tools which make it easier for them to achieve their fitness goals, and, most importantly, to incorporate them to their lifestyle.

FitVenture is an Android application that enables user to connect with other like-minded fitness or sport enthusiasts in their proximity and to work on their goals together. At the same time, FitVenture is also a growing community of people with the common attitude and goal in mind – to take fitness as a life-lasting adventure. Despite the fact that many people underestimate or do not realise this at all, enjoyment is one of the most critical factors which directly affects one's intrinsic motivation to stick to their (not only) fitness plans – and that is what FitVenture is all about.

Acknowledgements

First and foremost, I would like to thank my friend Lukas Schweighofer for all the support and passion for FitVenture idea. His honest feedback, creative ideas and determination have helped me a lot throughout difficult and busy times.

I would also like to thank my family and friends for all the support, encouragement and patience whilst I was working hard on this project. Special thanks goes to Monika Krajcirova, Miroslav & Pavol Hlubik, Maros Galovic, Michael Wuketich and everyone else taking part in and supporting FitVenture activities – together, we make a wonderful team.

Last but not least, thanks to my supervisor Wendy Ivins for providing me with useful and insightful advice, guidance, support and, finally, patience throughout the entire project duration. Her honest, constructive and valuable feedback has helped me to improve and grow on this exciting journey.

Table of contents

1. Introduction 1
1.1. Outline 1
1.2. Scope 1
1.3. Aims & objectives 2
1.4. Approach 4
1.5. Tools used 6
1.6. Structure of the report 7
2. Problem specification & analysis 9
2.1. The problem 9
2.2. Problem analysis 9
2.3. Existing solutions 11
2.4. Project justification 12
2.5. Constraints 13
3. Non-development activities 17
4. System design & structure 21
4.1. Logical data model 21
5. Sprint one 23
5.1. Background 23
5.2. Requirements 23
5.3. User story & tasks 25
5.4. Design documentation 25
5.5. Implementation 27
5.6. Testing 29
5.7. Sprint evaluation 30
6. Sprint two 31
6.1. Background 31
6.2. Requirements 31
6.3. User story & tasks 33
6.4. Design documentation 34
6.5. Implementation 36
6.6. Testing 43
6.7. Sprint evaluation 43

7. Sprint three 44
7.1. Background 44
7.2. Requirements 44
7.3. User story & tasks 46
7.4. Design documentation 46
7.5. Implementation 47
7.6. Testing 55
7.7. Sprint evaluation 55
8. Sprint four 56
8.1. Background 56
8.2. Requirements 56
8.3. User story & tasks 56
8.4. Design documentation 56
8.5. Implementation 58
8.6. Testing 64
8.7. Sprint evaluation 65
9. Sprint five 66
9.1. Background 66
9.2. Requirements 66
9.3. User story & tasks 68
9.4. Design documentation 69
9.5. Application UI & UX Design 74
9.6. Testing 77
9.7. Sprint evaluation 78
10. User testing 79
10.1. Background 79
10.2. Survey structure 79
10.3. Results & consideration of feedback 80
11. Evaluation 84
11.1. My own testing 84
11.2. User testing 84
11.3. Future improvements 85
11.4. Project management 86

11.5. Reflection on learning 86
12. Conclusion 89
12.1. Summary 89
12.2. Project aims & objectives 89
12.3. Future work 94
13. Bibliography 97

1. Introduction

1.1. Outline

Fitness industry has seen a huge growth over the last few years, producing countless number of different supplements, nutritional experts, gyms and other facilities, personal coaches and, lastly, applications.

However, even though there is a vast number of applications on the market, I can still see people struggling to stick to their plans in the long term. In the better case, they achieve their goal but then they get back to their old habits and have to start over. In the worst case, they do not even achieve that goal. The problem of all these applications is that only a very small number of them truly focuses on research-underpinned factors that affect our motivation in the long term.

Since I used to struggle to stick to my active lifestyle myself and my motivation levels would often fluctuate quite drastically, I decided to read more about the topic and find out why it is so hard to stay focused and motivated at all times. I then put some elements of this newly gained knowledge, such as group activity, sharing goals with one's friends and family, having the gear visible at all times, scheduling the fitness activities or tailoring the exercise and nutrition around one's lifestyle into practice, and, to my surprise, it really worked. Despite the fact that these elements, among others, significantly affect one's motivation to adhere to a long-lasting active lifestyle, they are often forgotten and instead, less relevant and less effective factors are promoted as superior.

I tried to find an application which would make use of all these things as well, but I could not find any suitable one. Therefore, to enable myself as well other users reach their goals and stick to their plans by always having steady levels of motivation, I decided to research this even further and develop such an application myself.

1.2. Scope

The main aim of the project was originally to develop a solution to the identified problem. However, as I delved deeper into several courses on Android applications development, I gained better understanding of what it nowadays really takes and means to create a successful application.

For a completely new application to have a chance of succeeding on today's extremely competitive and saturated market, one cannot underestimate the importance of having a vivid and passionate community which cares about their product as well as having effective marketing in place (Muellauer and Yu, 2018). The development of the application itself is just the tip of the iceberg.

Moreover, considering the focus of my degree course – Business Information Systems - and having learned what it actually encompasses to launch such a product in real world has helped

me view the project as a much more complex process than just development. Since I am planning to continue working on this project even after finishing the university, I decided to simulate the real world conditions as much as possible. Therefore, I stopped looking at the project in an isolation and I stopped focusing on development only. Rather, I researched and did my best to implement all other important factors and processes which are inevitable to success of such a project. These will be covered in more detail in Chapter 3: Non-development activities. Hence, the entire project would usually consist of these broad elements in real world (Muellauer and Yu, 2018):

1. Idea
2. Requirements
3. Design documentation
4. Application development
5. UI & UX design
6. Application testing
7. Idea validation
8. Marketing
9. Application launch
10. Monetisation

Whilst I tried to simulate the real world conditions as much as possible, given that there is only 11 weeks for the entire project including both initial and final reports, it was not feasible to cover all of the above.

After careful consideration, I chose to focus on the first seven elements – from an initial idea all the way through to idea validation - as I felt these would be realistic to complete as well as giving me chance to put all the gained knowledge into practice in a self-managed project. The rest is out of scope of this project.

1.3. Aims & objectives

By the end of the project, I am aiming to have the following components finished:

1. **Acquire knowledge about developing Android applications**

By undertaking several courses, I am going to get insight into Android application development and testing as well as touching on topics like application launch and marketing to be able to see the bigger picture.

2. **Gain an understanding of the problem background**

To gain a better understanding of the background of the problem I am trying to solve with my application, I am going to research more about motivation to exercise and to lead a long-lasting healthy lifestyle as well as factors that influence it.

3. **Gain an understanding of the market & the need for my application**

To gain an understanding of the market, I am going to do a research on what has already been done to solve the problem. I will also figure out how my application will complement the market and how it will add value rather than just copy or modify what has already been made. I will be looking at the most successful and widely used solutions and I will then compare them to my app and justify the need for it.

4. **Acquire an awareness of the possible approaches and tools & choose a suitable one**

First of all, I am going to choose a suitable set of tools that I will use throughout the project, such as project management tool, software development methodology, development environment, development language and database system to name a few. This needs to be done as soon as possible for me to be able to undertake any potential courses needed before I delve into implementation.

5. **Gain an understanding of user needs**

Based on the research of the problem, existing solutions and an informal research of my potential users, I am going to collect all the desired functionality and document it in what will become the functional and non-functional requirements of the app.

6. **Outline a graphical representation of the app's functionality**

All the requirements will be represented by a low fidelity design technique (flow charts) and higher fidelity technique (mock-ups). This will be done in line with the software methodology used.

7. **Develop a working solution**

I will consider this project a success if the following functionality is working. I am going to refer to this as a Minimum Viable Product (MVP).

The working solution will have the following:

- find a fitness buddy (nearby) – user will be able to find their potential fitness or sport partner in their area
- messaging system – user will be able to text their partners
- user system – user will have their own profile with some basic as well as fitness information
- activity tracker – user will be able to record their activity and see their progress graphically

If there is enough time, these features will be "nice to have":

- goal setter – user will be able to set a desired fitness-related goal
- reputation system – each user will have a reputation score influenced by several factors
- simple verification system – this system will be optional and will track user's progress against what they have set out as their goal in the beginning. It will reward or take away reputation points depending on if they stick to their goals or not.

8. **Gain an understanding of the app's performance & usability**

Once implemented, I am going to test the application against the acceptance criteria in line with the software methodology used. Once the development is completed, the application will further be tested by a small group of users to help me better understand how the app is performing and whether it lives up to their expectations. For this test, ethical approval will be needed to let users know of what this will consist of and to make them aware they can withdraw at any time. Therefore, I am going to submit my ethical approval form as soon as possible to allow plenty of time for approval.

9. **Build a community around the idea**

Since an app needs to have a vivid and large community of people built around it before it is even launched to have a chance of success on today's oversaturated market (Muellauer and Yu, 2018), my goal is to create a community of at least 100 fitness enthusiasts who share the same beliefs and ideas.

1.4. Approach

The approach is one of the most important things in the project as it directly affects the project processes, direction and subsequently deliverables and outcomes. Hence, I considered all the options carefully before making my choice.

After careful consideration of all the approaches, I decided to split the entire project into two separate sections, each having its own lifecycle:

- non-development activities (including idea refining, logo creation, market research, problem analysis, blog creation and others),
- development (including design documentation, requirements, coding, testing and UI & UX design)

and use a different approach for each. Waterfall Model for Non-development activities and Agile Approach for Development – overall making it a hybrid approach for the project.

As for non-development activities, I chose Waterfall, also called linear-sequential lifecycle model, because of several reasons. Firstly, I have had previous experience of working with this methodology during past university projects and I knew I would be comfortable using it. Secondly, it is in theory very simple and straightforward to use as it gives the project a nice clean structure in which phases do not overlap. This perfectly fits to my Non development part of the project - since the project is of rather short duration and everything has its own precise timeline, it was important to have everything planned out carefully before the start of development with as few changes as possible. And that is one of the few areas in which Waterfall excels (Leach, 2016).

Regarding the Development part of the project, I was not very confident about using Waterfall as in my own experience, barely anything ever goes according to the plan, especially in software development. More so, when I did not have any prior experience in Android or Java

programming. If anything went wrong in the development phase, I would likely need to amend or even completely rework the entire previous phases including requirements and design documentation. Apparently, that was not feasible considering there is only 11 weeks for the entire project.

Being aware of these Waterfall limitations which could potentially result in a serious disruption to the solution and even the entire project, I decided to look for a different, more flexible and changes-friendly approach which would ideally let me minimise the risk by building the product in a series of smaller iterations building on top of each other. So that when my plans change and I have to revert back and amend things, I will still have at least something and will not have to start all over again from the very beginning.

Having ruled out all the approaches that are mainly designed for bigger teams or large organisations, such as Waterfall or DevOps Deployment Methodology, and having realised the things mentioned above has helped me to make my decision easier. I further considered the following agile methodologies and frameworks.

feature	Scrum	Lean	Kanban	Extreme Programming	Rapid Application Development
iterative cycles of development	yes	yes	yes	yes	yes
duration of iterative cycles	2–4 weeks	as short as possible	not fixed, continuous flow	1-3 weeks	custom
customer/product owner involvement	high	N/A	high	high	medium
focus on documentation	low	low	low	low	medium
suitable for single persons/small teams	yes	yes	yes	yes	no

Table 1: A sample of SW development methodologies & frameworks

From the above, I completely ruled out Rapid Application Development as for it to work it relies on a skilled team of people (Powell-Morse, 2016) and therefore is not very suitable for a single person. The second one I ruled out completely was Kanban, as there is no fixed periods of cycles and the deployment is made when it is ready. Apparently this would not be a wise choice for me considering there was exactly eleven weeks dedicated for the entire project.

Out of the remaining three – Scrum, Lean and Extreme Programming - I chose Scrum as the basis of my approach, adopting it to my own needs. However, I also adopted some of the ideas and practices from both Lean and Extreme Programming.

I chose to mainly go for Scrum because it consists of fixed short time periods called Sprints (each delivering a working functionality) and does not focus on the documentation too much (Scrum.org, n.d.), thus allowing me to build a working solution, even if the intended functionality changed during the project or if some functionality was stripped due to time constraints.

Moreover, as opposed to Kanban, Scrum relies heavily on micromanagement, planning sessions and regular evaluations and since I am used to setting daily goals for myself anyway, Scrum seemed like a perfect match.

Of course, as Scrum is not designed for individuals but smaller teams, I had to slightly tailor it for my project's purposes. My Scrum events therefore included 2-week sprints, sprint planning at the beginning of each sprint, daily scrums (but of course without having to synchronise activities with anyone) and sprint evaluations in which I reviewed what was done and achieved and what went wrong or well.

On top of that, I also adopted some ideas from Lean Thinking and Extreme Programming, such as minimisation of waste or focusing on simplicity (Vliet, 2008). At all times, I tried to keep in mind that there is only 11 weeks available for the entire project and thus did my best to reduce the "waste" whenever possible by cutting unnecessary tasks or activities. For example, even though I was very interested in finding out more about certain Android or motivation related topics, I only took out what was absolutely necessary to continue working on the project and left the rest to be learned afterwards.

In summary, my approach for Development part of the project was heavily based on a slightly adapted Scrum methodology whilst also adopting some elements of Lean Thinking and Extreme Programming.

1.5. Tools used

Throughout the project, I will be using a wide range of tools, including:

Project Management Tool

There is a countless number of tools available for agile project management including JIRA or Taiga. I used JIRA on my placement last year and whilst I found it really useful, it seemed a bit too complicated for a project involving one person only. Therefore, following the recommendations of my supervisor, I chose Taiga which is free for one person and very simple and easy to use. Nonetheless, it contains everything I needed including product backlog, user stories, tasks and sprint planning. Moreover, it gave me an opportunity to learn something new.

Software development editor

There are many different IDEs and code editors available for Android development. They can be divided into two categories – those enabling user to develop native applications and those enabling user to create hybrid ones. Since I chose to create an application solely for Android, I was able to rule out all the tools like Cordova or Corona.

From the native tools, the most known are Android Studio and Eclipse. Android Studio is now the official IDE for Android which usually means there is more documentation, support and updates available. Therefore, I decided to go for Android Studio.

Database system

Even though I have never used FireBase before, I decided to choose it because of multiple advantages which fitted my project perfectly. Since it is a NoSQL database and all the data is stored in JSON trees, it was up to me to decide on the structure of the data. Furthermore, as it is a cloud-hosted real time database system, all clients connected to the instance of the Realtime database automatically receive updates with the newest data (Firebase, n.d.). However, the most important factor for me was that it is cross platform which means I can easily integrate it with a web or IOS application later on.

Version control system

Version control is one of the most important things with regards to development and cannot be underestimated. I decided to use GitHub as I was already familiar with it and it works well with Android Studio. Furthermore, it provides additional useful features such as bug tracking, feature requests or wikis which I will make use of in the future, too.

Balsamiq

For creating mock-ups and prototypes, I chose to use Balsamiq as wireframing tools are usually expensive and I can use Balsamiq for free under my academic email. I used it for a couple of other university projects in the past and it does the job well.

Draw.io

For the application's data model and flowcharts, I used a tool called Draw.io. It was one of the few free modelling tools available but does the job perfectly, allowing user to customise the project in a countless number of ways.

1.6. Structure of the report

Chapter 2: Problem specification & analysis covers the outlined problem into detail as well as its analysis. In addition, it discusses already existing solutions and the reason why they are not sufficient in solving the problem. Various legal and ethical issues are considered, too.

Chapter 3: Non-development activities covers all activities other than development and their contribution to the project.

Chapter 4: System design and structure explains the structure and design of the mobile application.

Chapters 5 to 10: Sprint 1 to 5 track and explain the development process from beginning to end, consisting of 2-week long sprints.

Chapter 11: User testing explains how user testing was conducted, the feedback received and how it was interpreted and acted upon.

Chapter 12: Evaluation covers a critical assessment of my progress against the project aims & objectives as well as evaluation of my own progress.

Chapter 13: Future work discusses the potential of FitVenture with regards to the future and all the work that can be done.

Chapter 14: Conclusion contains the final summary of the project.

2. Problem specification & analysis

2.1. The problem

We live in a fast-paced society where people are full of stress and anxiety and tend to have desk-bound jobs. As technology is improving faster than ever and becoming a stable part of our lives, we understandably rely on it more and more. However, all these factors have contributed to the fact that we often struggle to find some time for ourselves and to commit ourselves to leading a healthy lifestyle. The insufficient amount of movement and exercise combined with bad eating habits and sedative jobs have recently largely contributed to the increased obesity, currently being the second highest cause of premature death in Europe, increasing the risk of various health conditions such as diabetes, high blood pressure or osteoarthritis (Dearden, 2017). What is worse, obesity rates are projected to further increase, according to experts from OECD (2017).

To efficiently solve this problem of inability to lead a long-lasting healthy lifestyle, the underlying cause needs to be understood and removed. One of the several main reasons for this lack of active lifestyle among people of all ages is lack of the right kind of motivation. Whilst it is fairly easy and straightforward to start doing a physical activity, the challenging part is to stick to it (Biddle and Fox, 1989).

Therefore, the main purpose of my mobile application is to equip user with the tools necessary to reach their desired fitness level within a certain period of time. In order to achieve that, user needs to have the right kind of motivation. FitVenture application seeks to solve this by enabling user to connect with other like-minded individuals and pursue their fitness goals together. It will also provide them with an activity tracker and verification system to ensure their motivation remains at steady levels at all times. As will be explained in the following sections, these are the most critical factors when it comes to motivation and sticking to a plan or programme.

2.2. Problem analysis

The motivation to exercise can be split into two categories - intrinsic and extrinsic. Whilst the first one stems from the inner joy of doing the activity itself and getting competent at it, the latter is highly influenced by the achievement of extrinsic outcomes, such as improved appearance or losing weight (Ryan et al., 1997). In fact, a three-year long study (Silva et al., 2008) points out that only one in five people whose primary goal is losing weight is actually able to maintain a long-lasting physically active lifestyle after reaching that goal. On the other hand, those driven by intrinsic motivation are much more likely to stick to their plans in the long term, since their energy, confidence, level of joy and mental health are all boosted after each session of exercise or activity.

Couple or group activity

According to many experts including Wankel (1993), leading a healthy lifestyle and exercising in a couple or group has many advantages and is superior to exercising alone. This is because social interactions usually tend to add to one's enjoyment of participation as well as reducing stress even more than when exercising alone (Plante, Coscarelli and Ford, 2001). Since enjoyment has proven to be a critical factor when it comes to adhering and sticking to a plan, it cannot be underestimated.

Another important factor is that group activity makes use of Köhler Effect (Kerr and Hertel, 2011) – the idea that nobody in a group wants to be the weakest link – and hence, everyone pushes harder than they would on their own.

In addition, one study (Holt-Lunstad et al., 2015) even points out that regular social interactions decrease the risk of early mortality associated with social isolation. Apparently, I am not assuming my users to be living in isolation, but this is one more good health reason to work out and exercise with a partner or group.

Doing activities in a couple or group is the essential idea behind FitVenture and the application implements it by allowing users to get to know other like-minded individuals in their area who are into the same kind of fitness activity or sport. On top of that, open group trainings (meaning that anyone can join for free) are organised regularly by FitVenture purpose of which will be explained in the following chapter.

Own decision

According to experts (Ryan et al., 1997), one has to decide on their own and voluntarily make the choice to lead a healthy lifestyle. That is, if it is to last long term.

Because of the ever-growing fitness industry, the increasing popularity of group activity and other important factors, FitVenture as a mobile application has a great potential of becoming popular among certain age groups. When people see what their friends have or are currently using, they tend to want to own or use it as well. Therefore, when people see how FitVenture helps leading a healthy lifestyle, they will likely want to try it out as well. As a result, they will be inspired and it will be their own choice to change their lifestyle.

Gear visible at all times

Having the workout gear, clothes or sports equipment such as kettle bells, tennis rackets and others on visible places prepared and ready for use is also a great way for increasing one's motivation (Adam and Galinsky, 2012). Not to mention one's busy mind, there are also many things and objects competing for their attention which in turn tends to affect their priorities and actions. Hence, many recommend not putting the gear away but rather keeping it on eyes as this visual cue is a wakeup call for the brain.

Currently, there is more than 2,5 billion of smartphone users worldwide and this number is expected to reach 2,87 billion by the end of 2020 (Statista, 2018). In 2017, almost 30% of people worldwide spent on average more than seven hours per day on a mobile device. FitVenture as a mobile application is always at hand and visible. It does not only remind users

to exercise and to remember the commitment they made, but also makes use of customised notifications, serving as a very powerful and strong incentive for users to stick to their plans.

Find the fun in it

Even though many may think this is not of much importance, according to Ryan & Deci (2000), in order for a person to stick to a training programme, they need to have an intrinsic motivation for it stemming from enjoyment. In other words, they have to find it fun and engaging. In fact, this is one of the most critical factors that influences one's ability to adhere to their plans and therefore should be a priority of any application claiming to be boosting motivation in users.

FitVenture not only promotes and facilitates working out in a group or with a partner, in contrary to their direct competition (TennisBuddy, FootyAddicts described in detail later in Project justification) it also provides users with plenty of activity options to choose from, be it dancing, calisthenics, hiking, running or others. As a result, users are much more likely to find a partner who is passionate about the same kind of activity - which they both will enjoy.

2.3. Existing solutions

Prior to start of the project, I was fully aware that there is many other solutions to the problem on the market already. However, none of the solutions viewed or focused purely on the intrinsic motivation as a deciding factor between failure and success. Here is the most well-known solutions, sorted by the number of downloads on Android's Play Store:

- MyFitnessPal (50 million)
  This an app with freemium earning model. The free section contains features like goals setting, discussion forums, diet planner and meal and exercise logger. Users can conveniently log their meals by choosing from a huge pre-made database of food and various recipes. The premium version contains bonus articles and tips and users are able to set a different goal for each day. Moreover, it analyses their nutrition in real time so they are aware of their macro nutrient intake at all times and can act upon it accordingly straight away.
- Endomondo (10 million)
  This is also a freemium application, but focuses more on "fun" element to fitness. With the features like fitness tracker, personal stats, newsfeed containing friends' activities and various competitions against them in the free version, it makes a solid "competition" to FitVenture. In the premium version, users can get a tailored fitness programme based on a fitness test.
- FatSecret (10 million)
  This free application explicitly focuses on weight loss and offers users an exercise and diet tracker, calorie counter, vast amount of healthy recipes and weight chart.
- LifeSum (10 million)

LifeSum is a freemium app providing users with workouts and meals logger, habits tracker and daily feedback. Users can also find more advanced stats and various healthy recipes in the premium version.

- Strava (10 million)
  Strava is a social network fitness app offering standard fitness applications components such as training log, activities analyser or goals setting. On top of that, there is also routes and trails database, navigation and leader boards. Premium members can also enjoy customised goals, real-time feedback and live tracking (a safety feature).
- Runtastic (10 million)
  Runtastic in its free version provides users with basic exercise and running diary. There is a vast amount of exercises to choose from already and new ones are being added all the time – these will satisfy the majority of users. If users choose to go for the premium version, they get access to various training plans, real-time voice coach, leader boards, weekly email reports and also become much more flexible in terms of tracking their workouts.
- LoseIt (5 million)
  As the name suggests, this application focuses on weight loss and as one can expect, it provides calorie and exercise tracking. What makes it stand apart from the competition is the ability of users to track their nutrition by simply scanning the barcode or QR code where possible.
- Accupedo (5 million)
  Accupedo is a smart step counter providing various stats as well as integration with MyFitnessPal (mentioned above).

2.4. Project justification

To a casual user, it may seem like the above-mentioned applications provide just about everything one can ask for. However, as explained earlier (Section 2.2: Problem analysis), one of the main challenges and key strategies to leading a long-term healthy lifestyle is to enjoy the activities instead of focusing on goals like body appearance or losing body weight. Even though it is nice to have goals and lose weight or gain muscle, the likelihood is, one is not going to stick to it for the rest of their life unless they truly enjoyed the process.

Even though some of the applications above do try to incorporate factors that affect one's intrinsic motivation by providing users with various leader boards or even Facebook-like friends newsfeed, they have one thing in common. They all assume that everyone has somebody to work out or play sports with. Or even share their goals and progress with. According to my experience, that is not the case with majority of people.

In reality, most people

- do not know of anybody who is into the same kind of fitness activity or sport,

- are shy to ask (regardless of if it is a stranger or someone they know), especially if they are beginners,
- do not realise the potential of exercising with a partner.

By giving users opportunity to find and connect with other similar fitness enthusiasts in their area as well as educating them about why it is so beneficial to exercise not alone at least from time to time, FitVenture solves all three points at once.

There is a couple of applications on the market trying to solve this as well, but they are too specific, focusing on one sport only. There is currently no application for finding a fitness partner, be it for calisthenics, cross fit, dancing or working out in a gym.

- TennisBuddy (10 thousand downloads)
  As the name suggests, this application allows users to find a tennis partner within their area.
- FootyAddicts (5 thousand downloads)
  This application is for organising friendly football games.

Despite the fact that group activity or exercising is certainly not a new concept, it has seen a massive raise in popularity over the last couple of years, climbing more than 10 places and currently ranked as number two in the Worldwide Survey of Fitness Trends for 2018 (Thompson, 2017). Moreover, it is predicted to grow even more in near future. Social media, websites and increasing awareness of the potential benefits have all contributed to this positive trend. Fitness industry is also experiencing a steady growth of 2.6% in revenue globally and according to the latest IHRSA report (2017), is projected to reach $87.5bn by the end of 2018.

Another factor that plays a role is the rising number of various fitness applications and websites, providing users more and more quality content than ever before. By quality content I mean various articles, text and video guides on nutrition, exercising and healthy lifestyle in general. As a result, the need for expensive fitness trainers and personal coaches is reduced (Thompson, 2017). This decrease makes room for the growth of solutions like FitVenture.

Overall, the ever-growing global fitness industry and mobile devices use, increasing popularity of group exercise combined with the easier-than-ever-before ability to connect with other like-minded people nearby gives FitVenture a great potential to succeed.

2.5. Constraints

Every project is affected by its environment including economy, law or various regulations and FitVenture was no exception. Therefore, all possible factors and issues that could affect the project were considered and accounted for.

2.5.1. Legal & ethical issues

General Data Protection Regulation (GDPR)

Considering that I want to keep working on my application even after university and the advent of a regulation intending to unify and strengthen data protection of all individuals – GDPR - in the upcoming weeks, it was important for me to build the application in such a way that complies with this new legislation. Therefore, I created Privacy Policy as well as Terms & Conditions documents, ensuring they are both in line with this new legislation, by for instance:

- Uploading them on a website so that they can be read from any device (and not exclusively just mobile) to improve the readability and accessibility. They can be found online at http://fitventure.sk/privacy-policy/ and http://fitventure.sk/terms-conditions/
- Making sure it is not overly excessive
- Making sure it is easily understandable by most readers by reducing the jargon as much as possible
- Letting user know who controls the data and who processes them
- Lettings user know they can ask to see what data are hold on them at any time

Apart from creating GDPR-compliant Privacy Policy and Terms & Conditions documents, I also ensured to receive an explicit consent from user before registering by putting an empty checkbox next to the consent statement (see Image 1 below). As a result, user cannot process to the next step until the checkbox is ticked. This was very important as according to GDPR, one must be able to pinpoint how exactly was the consent given and what was consented to (General Data Protection Regulation (GDPR), n.d.).

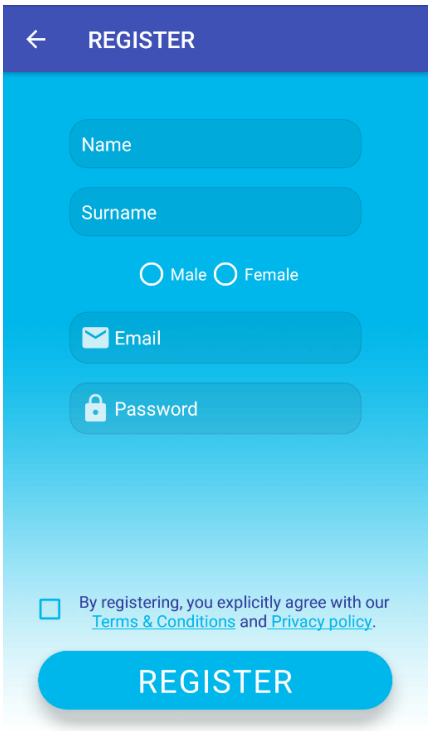
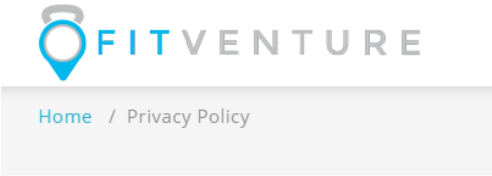


Image 1: Explicit user consent upon registration

Privacy Policy, Terms & Conditions

Since it was not in scope of this project to release the application to public, Privacy Policy and Terms & Conditions were not essentially required. However, since my aim was to simulate the real world conditions as much as possible, I decided to create both documents anyway (See Image 2 and Image 3 below. For full documents, please see Appendix D and E). Moreover, as I will be working on this project even after university, these two documents will be useful in the future, too.



PRIVACY POLICY

Introduction
COLLECTION OF YOUR INFORMATION
1.1 Personal & Sensitive Data
1.2 Derivative Data
1.3 Geolocation Information
1.4 Mobile Device Access
USE OF YOUR INFORMATION
DISCLOSURE OF YOUR INFORMATION
3.1 By Law or to Protect Rights
3.2 Interactions with Other Users

Image 2: Online privacy policy document

TERMS & CONDITIONS

Introduction
1 Content
1.1 Links to other websites
2 Your responsibilities
2.1 Cooperation
2.2 Failure to cooperate
3 Personal information and Registration
3.1 Registration
3.3 Contacting you
4 Licence Agreement (EULA)
4.1 Acknowledgements

Image 3: Online terms & conditions document

Safety

Since this application promotes connecting with and meeting new people, there are naturally safety risks associated with this. To reduce the risks, the search system was implemented in such a way that user cannot be tracked by another user (this will be further explained in Chapter 7: Sprint three). Moreover, the application will encourage using common sense and provide small notification hints for users to stay careful.

Pre-made fitness programmes & routines

Even though Activity Tracker was not implemented due to time constraints in the end, I have still considered the potential issues associated with it.

Since activity tracker provides user with an option to also choose a pre-made fitness programme or routine which were made by FitVenture, I will need to make sure that these are created with the help of a certified coach or in line with a verifiable study. Under each pre-made programme or routine, there will be a note informing user to discuss with their

doctor before enrolling. Lastly, I will also specify in Terms & Conditions that FitVenture is not liable for any injury caused by use of these pre-made programmes.

User testing

Originally, I aimed to conduct a user research on a medium-sized group of 10-15 people with different demographics, collecting all kinds of sensitive data supported by the Privacy Policy and Terms & Conditions.

Since it consisted of a questionnaire involving sensitive data, I had to undertake Research Integrity Online Training Programme and apply for an approval from Cardiff University's Research Ethics Committee to ensure the research was conducted in line with the University's regulations and best practices.

However, due to time constraints and the fact that the application was not going to be released to the public within the scope of the project anyway, I changed the aim of my research - as the application would still need to be thoroughly tested before launch again.

Hence, I decided to only conduct a small scale user testing consisting of 3-5 people which would give me a useful insight into the how the application performs and how usable it is. And for this, I did not need to collect any sensitive which immensely simplified the approval process.

Therefore, I firstly created the consent form and briefing and debriefing sheets and, secondly, amended both Privacy Policy and Terms & Conditions documents to match only what was going to be collected.

3. Non-development activities

I had no previous experience with Android prior to start of the project. To compensate for this, I undertook a comprehensive "Android & Java – Mobile App development, beginning to end" course on Udemy, starting in December - well before the project start. I managed to successfully complete the course by the end of Week 3. Whilst this course could not possibly have covered everything there is from the coding perspective, it has certainly given me solid grounds towards developing FitVenture. Most importantly, it has also broadened my horizons and showed me that gone are the days when one could just code an application, put it on Play Store and become famous and rich within hours. It has showed me that the development, or coding part, is really just the tip of the iceberg.

As mentioned in Section 1.2: Scope, the project could logically be split into the following processes, which did not always happen in the order they appear here.

3.1.1. Idea

The idea formed in my head gradually. I have been into fitness and sports since my teenage years. Firstly, it was table tennis, then various martial arts, running, gym, calisthenics, tennis and then jumping from one to another.

Apparently, one needs a partner for table tennis and tennis. From my own experience, I know how hard, at times, it may be to find one at a similar level. Whilst there usually are opportunities to play, I was quite surprised there is no website or application for finding a sport partner in one's location and at a similar level.

In terms of running and working out, it is true one does not necessarily need a partner. However, as I was gradually finding out on my own, it is much more engaging, inspiring and motivating to work out or run with someone else. What is more important, it is easier to stick to the goals and routine when one has somebody to support them.

Over the years, I have realised this and also seen on a daily basis that it can do wonders, hence I came up with the idea to create a simple mobile application which will do just that – find a workout or sport partner nearby. The final year project seemed like a perfect opportunity to realise this idea as it is supervised and I can receive useful feedback and guidance.

Name

The first important milestone and starting point was to create a name. One may argue that name is not important in the beginnings, but I believe the name can give one a strong sense of tangibility. In addition, they can then start talking about it and spread the "buzz" around it which is very important.

Therefore, I decided to come up with the name as soon as possible. Out of more than 150 proposals I created over the first three weeks, I managed to narrow it down to five. Then it was an easy choice as the other four were, to my surprise, already being used by companies

operating in a similar segment. A good name should be memorable, rather shorter and clever and most importantly, it should be congruent with the story behind the idea or company. “FitVenture” not only satisfies all of the above, it is also clever and inspires users to think out of the box. Essentially, it summarizes the entire idea of having fun and going on a long-term fitness adventure in one word.

Logo

Another important milestone was logo. Similarly to the name, it also ideally triggers all four cognitive events in humans – attention, response, meaning and memory (Lidwell, 2014). Since the today’s fitness market is oversaturated and it is very difficult for a new application to stand out, I could not stress this point enough. Furthermore, it should represent the story behind the idea, too. My first thoughts were, understandably, a dumbbell or a barbell symbol combined with the symbol for location. However, every other fitness app uses them, so I came up with kettle bell which is an amazing fitness tool used by many worldwide. Especially now that fitness is not all about just going to gym anymore, kettle bell was a perfect match for my idea. It symbolizes fitness as a whole as well as the idea that one can keep themselves fit by doing any kind of fitness or sport, not necessarily working out in a gym. It inspires novelty and experimentation. The blue part – a GPS sign – efficiently tells the story of FitVenture. That is, users can find and connect with other fitness enthusiasts in their whereabouts within a few clicks. Together, these two symbols form a simplistic, yet attention-grabbing, memorable, friendly and meaningful logo with a propositional density higher than one. Since the logo will often be the first thing potential users notice, it was very important for me to get it as good as possible.

3.1.2. Requirements

Based on my idea and countless number of informal talks with my friends who are into fitness as well, I was gradually forming requirements for the application. Below is an overview of the basic functional requirements which will further be specified in detail in the subsequent chapters.

User system

Every user should have their own profile which will retain all their data and into which they can log in from any device with Internet access.

Fitness partner finder

Users should be able to search through a list of other registered fitness enthusiasts nearby and connect with their potential fitness or sport partner.

Messaging system

Users should be able to chat with their fitness partners.

Activity tracker

Users should be able to record their activity and see their progress graphically.

3.1.3. Design documentation

Documenting the design of the application was another important step and for each sprint, I created a low fidelity sketch (on paper) and high fidelity mock ups and prototype (in Balsamiq). Moreover, I also created low fidelity flowcharts to help user visualise the flow of each sprint. All of these were done in line with the agile approach used. This proved to be very useful and ensured my piece of mind, as if something changed, I would only need to recreate the current sprint's design as opposed to recreating it entirely.

3.1.4. UI & UX design

Once the application was implemented, I followed the mock ups and prototypes to create a simplistic design that adheres to Android's material design and Android's best design practices.

3.1.5. Application testing

The application testing was performed in two independent steps – firstly by myself at the end of every sprint against the acceptance criteria and then by a small group of users once the application was completed.

3.1.6. Idea validation

Every idea should be validated first before investing excessive amount of resources and time into it and FitVenture was no exception.

Since this was a completely new idea and application, it was of paramount importance to me to see how much potential it had. Moreover, as I would love to continue working on this project even after university, I decided that I will start the process of validating the idea now.

One can validate an idea in many ways. I decided that I would validate mine by:

Community

To see if there is interest in this idea, I decided to create a vivid and passionate community of people around it through both social media and in person. Therefore, I firstly created a Facebook and Instagram page and tried to attract as many fitness enthusiasts passionate about leading a long-term healthy lifestyle as possible. The plan was to provide them with quality informative content with added value that would inspire them on a regular basis.

Secondly, I decided to organise regular open trainings during which people would gather and work out together. These would be open to public so that anyone could join.

Thirdly, I set out to create a webpage with blog and to regularly add articles.

Because all Android applications get released into what is called Play Store, I studied how Play Store algorithms work. It turned out that the new applications are ranked based on their number of downloads in the first day, first week and first month in comparison to other similar applications. Therefore, the main goal of any new application is to get as many downloads as possible from the very beginning. Apparently, this is impossible in today's oversaturated market when more than 6000 new applications are released every single day (Statista, 2018). And this is why the creation of such a passionate community of people is inevitable for the success of a new application. It does not guarantee it, but it definitely does increase the chances.

Hence, creating the community was not only good for me to see if there is an interest in my idea, but it will also be of paramount importance when the application will be launched. Moreover, I decided to also create a landing page about my application and collect emails of fitness enthusiasts who would be interested in using it, but this quickly turned out to be out of scope of this project due to time constraints.

4. System design & structure

4.1. Logical data model

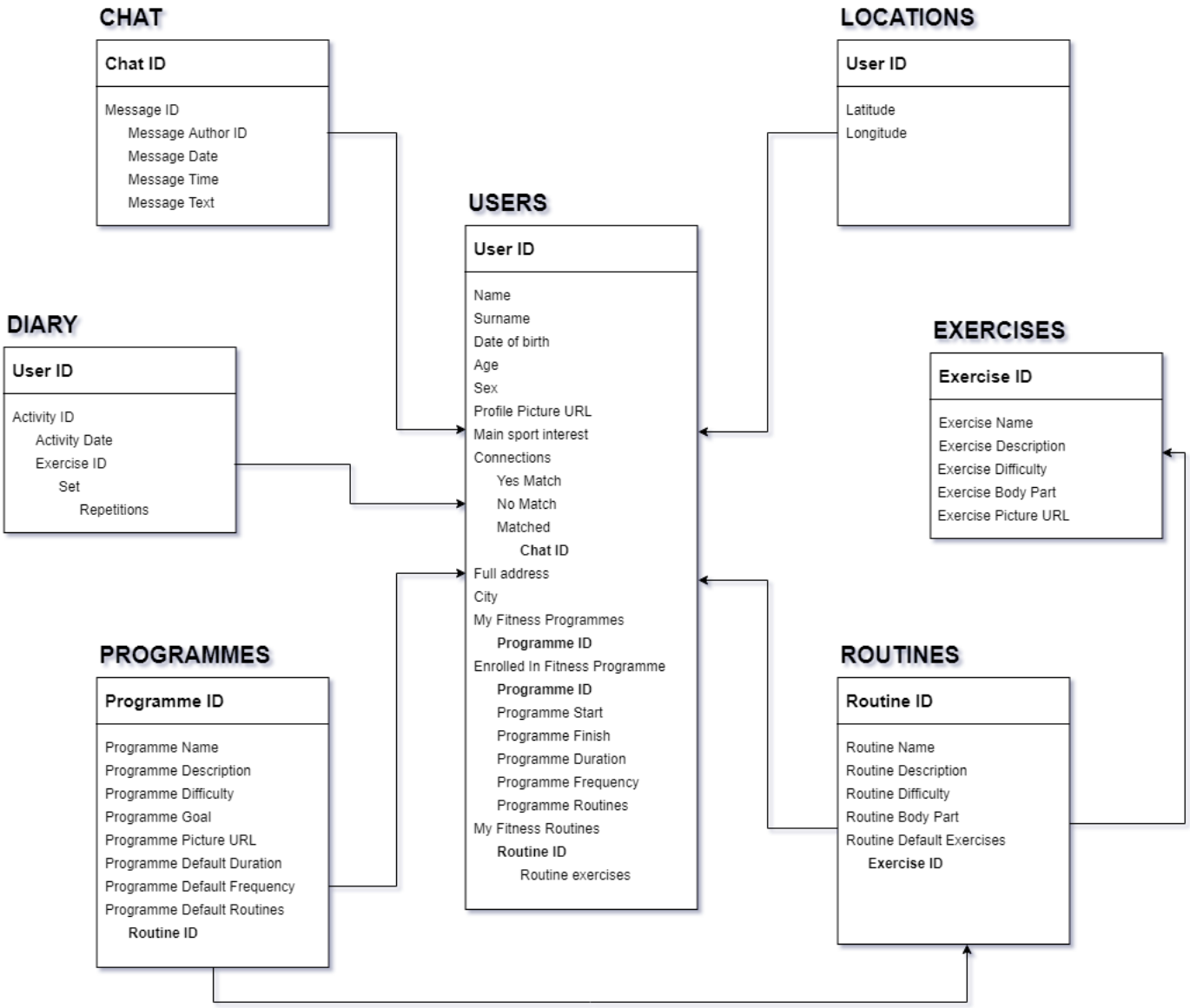


Image 4: Logical data model

Since all the data is stored in FireBase which is a NoSQL cloud database using unstructured JSON data, it was up to me to organise and structure the data model. To ensure scalability and maximum speed and efficiency potential of FireBase, I followed the best practices.

Even though it is common practice to have a deeply nested data structure when using JSON, FireBase performs best with shallow data structures (Esplin, 2016) as it can only query on one

child node at a time (direct child of the list's top level). Therefore, I normalised the data where possible to increase the reading speed potential of FireBase. Of course, I had to find the right balance between normalisation and denormalisation based on how the data would be used (Firebase, n.d.).

For example, since the application will need to have a quick access to the name of each fitness programme for it to be able to load and display them to user, they are kept in a separate list. This required me to have Programme ID in the two lists (Programmes and Users), which was a trade-off I was willing to accept for the sake of speed. In addition, I could have put *Enrolled in Fitness Programme* node as a child of My Fitness Programme -> Programme ID node. However, since this user collection of fitness programmes can potentially grow very large and the application will need to quickly determine which programme they are enrolled in, it would need to go down the branch for every single programme node just to find out if *Is Enrolled in Programme* is true. This would not be very efficient.

5. Sprint one

5.1. Background

The first sprint was to be completed over week three and four following the successful completion of the mobile development courses.

5.2. Requirements

5.2.1. Functional

1. Application should create a new account for each newly registered user

Explanation:

When user registers a new account, the application should create that account on the database as well as saving all the related data. This will be accessed and used by the application later on.

Justification:

In order for user to be able to access their data from any device, the account and all its associated data needs to be saved on a secure database. This may be needed if their device is stolen or if they simply want to use a different one.

Priority:

Essential

Acceptance criteria:

A new account is created on FireBase Authentication, containing valid email, date of creation, date of last sign in and user ID.
All the relevant data including name, surname and sex is uploaded to FireBase Database.

2. Application should let the user log in

Explanation:

Now that the data has been stored on a database, the application should let user access and see that data by logging in.

Justification:

In order to access all the account data saved by the application upon registration, user can log in to the application.

Priority:

Essential

Acceptance criteria:

User gains access to the application and the main screen opens.

5.2.2. Non-functional

1. Registration and log in process should be quick

Explanation:

Provided that user has a standard speed of Internet connection, the time it takes to register and log them in should be reasonable.

Justification:

Unnecessary long periods of time waiting for the application's response will severely worsen the user experience and likely deter them from using it.

Priority:

Essential

Acceptance criteria:

User is registered without having to wait unreasonable amount of time.
User is logged in without having to wait unreasonable amount of time.

2. Registration should be simple so that it does not deter user from creating an account

Explanation:

When the application takes user to the registration screen, they will be need to fill in a few details, such as name, surname, email and password. This needs to be simple and should not take too much time.

Justification:

Too many fields or thinking about what to fill in could deter a potential user from creating an account.

Priority:

Essential

Acceptance criteria:

The application only requires the most necessary fields to complete at the time of registration.

5.3. User story & tasks

User story

As a person concerned with my overall fitness, I want to be able to create an account in FitVenture so that I can meet new like-minded fitness enthusiasts and track my progress from any device. For this, I need to be able to register and log in.

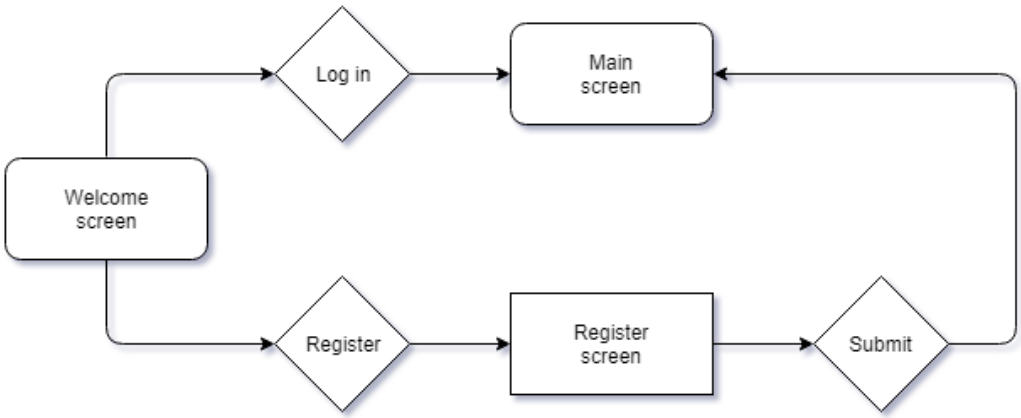
Tasks

Register – develop functionality which enables users to sign up and create a new account if they do not have one yet.

Login - implement functionality to validate the email and password combination and log the user in if everything checks out.

5.4. Design documentation

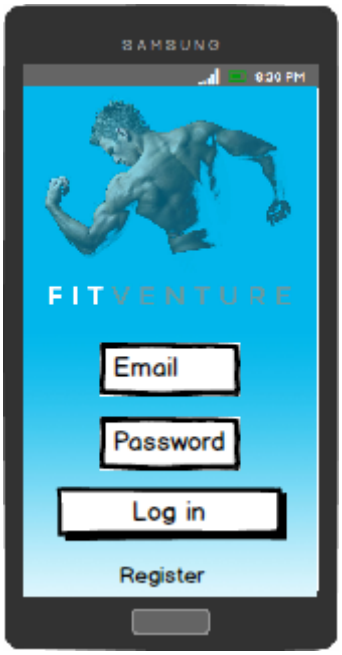
5.4.1. Flow chart



Flow chart 1: Registration and login

5.4.2. Mock-ups

Welcome screen:



Mock-up 1: Welcome screen

When user opens the application, they are presented with a simple Welcome screen (Image 1) consisting of an image, the text part of the logo, two text inputs and two buttons. They can log in using the email and password combination and clicking on the relevant button. Once their credentials have been verified, they will be redirected to the Main Screen. Alternatively, they can create a new account by clicking on the Register button if they do not have one yet.

Registration screen:

Mock-up 2: Registration screen

By clicking the Register button on the previous screen, user is taken to the Register screen. In order to create an account, they need to fill in a very simple form consisting of only the necessary fields such as name, surname, sex, email and password. This is to avoid deterring user from registering by having to fill too much information. Moreover, they need to read and agree to the Terms & Conditions and Privacy Policy. These are both clickable links and take users to their respective websites. If the mobile device screen is too small and reading of these documents too difficult, they can open the same web address on a device with bigger screen. Once the Register button has been tapped, the application does a data verification and if everything checks out, it creates the account on the database and logs user in automatically.

Regarding the method by which user can create an account, I decided to only implement the email & password combination for now due to time constraints as it still is a dominant way of registering and logging in to applications.

5.5. Implementation

Registration

Since the entire application assumes and is built around each user having their own account, it seemed natural for me to implement the registration functionality first.

Once the Register Activity is created, i.e. opened by the user, an instance of FireBase Authentication object needs to be created first in order to be able to register a new user. In addition, FireBase Authentication also provides a listener, which constantly checks if the user is logged in. If it detects that user is or was previously logged in, it creates a new Intent and redirects them to the Main Activity which will be explained in the following sprints.

```
// instantiate Fire Base authentication object:
mAuth = FirebaseAuth.getInstance();

// set up a listener to check if the user is logged in:
mAuthStateListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        final FirebaseUser currentUser = mAuth.getCurrentUser();
        // redirect to main activity:
        if (currentUser != null) {
            Intent goMain = new Intent( packageContext: RegisterActivity.this, MainActivity.class);
            finish();
            startActivity(goMain);
            return;
        }
    }
}; // end of mAuthStateListener
```

Code excerpt 1: User authentication

Now that the FireBase Authentication object and listener are all set up, user's input needs to be validated. As for sanitization of the input, since Android is programmed in Java and Java is a compiled programming language, even if user tried to pass any potentially harmful code through the input, it would not get executed.

For the name, surname and sex fields, the application simply checks if they are not blank. Regarding the password, it needs to be longer than six characters to pass the checks. I did not try to implement any fancy checks such as that password would need to contain at least one upper case, one digit and one special character. In my opinion, it may severely damage the user experience and even deter a potential user from registering. After all, this is a fitness application.

As for the email, I chose to use an open source Java project called Apache Commons which provides a handy pre-built email validation feature. This feature can be accessed through *EmailValidator* class and its relevant method called *isValid()* and can recognise a valid email address in most of the cases.

If there is an error in any of the fields, the relevant error message is shown and the field in question becomes focused.

```
private boolean isEmailValid(String email) {
    return EmailValidator.getInstance(true).isValid(email);
}
```

Code excerpt 2: Email validation

Otherwise, a FireBase method *createUserWithEmailAndPassword* is called, passing it both the email and password user just entered. This method provides an on-completion listener, which I utilized to give user a feedback as to whether the registration was successful, or not. Apart from the feedback, I also used the onSuccess listener to upload all the data to the database.

```
// create a new Fire Base user:
private void createFirebaseUser(final String name, final String surname, final RadioButton sex) {
    String email = mEmailView.getText().toString();
    String password = mPasswordView.getText().toString();
    mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                // registration success - show message:
                String msg = "Registration successful.";
                mAppMessages.showToast(msg);
                // save data to the database:
                String currentUserID = mAuth.getCurrentUser().getUid();
                DatabaseReference currentUserDB = FirebaseDatabase.getInstance().getReference().child("Users").child(currentUserID);
                Map userInformation = new HashMap<>();
                userInformation.put("Name", name);
                userInformation.put("Surname", surname);
                userInformation.put("Sex", sex.getText().toString());
                userInformation.put("ProfilePictureURL", "default");
                userInformation.put("Age", "default");
                userInformation.put("DateOfBirth", "default");
                userInformation.put("LocationAddress", "default");
                userInformation.put("LocationCity", "default");
                userInformation.put("InterestsSportsMain", "default");
                currentUserDB.updateChildren(userInformation);
                // user will automatically be redirected to the main activity thanks to the firebase listener
```

Code excerpt 3: User registration

At this point, user's name, surname, sex and email get written to the database. All the other fields are populated with a default value.

Log in

The Log in functionality resides in the Welcome Activity. Same as for the previous functionality, both an instance of the FireBase Authentication object is created for later use and listener is set up to check if the user has been logged in.

When user clicks on the Log in button, the application checks if neither email nor password are blank. If they are not, a FireBase Authentication method *signInWithEmailAndPassword* is triggered, passing both email and password from the log in form as its arguments.

Again, using the listener associated with the above method to determine whether login was successful or not, the application provides user with the respective feedback. If the login was successful, a new intent is created and user is redirected to the Main Activity.

```
// sign in using
// 1. email & password:
mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        String msg;
        if(!task.isSuccessful()) {
            msg = "Couldn't log you in this time. Give it another try!";
            showDialog(msg);
        } else {
            Intent goMain = new Intent( packageContext: WelcomeActivity.this, MainActivity.class);
            finish();
            startActivity(goMain);
        }
    }
});
```

Code excerpt 4: User login

5.6. Testing

requirement	acceptance criteria	passed/failed
FR 1	A new account is created on FireBase Authentication, containing valid email, date of creation, date of last sign in and user ID. All the relevant data including name, surname and sex is uploaded to FireBase Database.	passed
FR 2	User gains access to the application and the main screen opens.	passed
NFR 1	User is registered without having to wait unreasonable amount of time. User is logged in without having to wait unreasonable amount of time.	passed

NFR 2	The application only requires the most necessary fields to complete at the time of registration.	passed

Table 2: Sprint one testing

5.7. Sprint evaluation

Even though I undertook the Android programming course before I went for the first sprint, I still found myself having to study enormous amount of time and material, especially to do with FireBase. Fortunately, FireBase makes the process of registration and logging in fairly straightforward. Once I grasped the understanding of how it is structured and what components and dependencies it requires, I successfully managed to implement both functionalities after only a couple of failed attempts and finish everything in time for the next sprint.

6. Sprint two

6.1. Background

Since I managed to finish the previous sprint within the dedicated time frame, I did not have any outstanding tasks to bring forward to the second sprint hence I started exactly at the beginning of Week five as planned.

For this sprint, I decided to code Search system and Match system, leaving the implementation of location for the following sprint in order to spread the complexity out a little. As a result, all users matching the search criteria would be returned regardless of their location.

6.2. Requirements

6.2.1. Functional

1. Application should display all other users on their respective card layouts

Explanation:

When user logs in and is taken to the Main screen, he or she should be presented with a list of other registered users, each one being displayed on their own card. This assumes that someone else had already registered and used the application before.

Justification:

In order for user to be able to discover, interact and connect with other users, the application needs to load all the users' data from the database and put each one on a separate card, ready to be shown to the logged in user.

Priority:

Essential

Acceptance criteria:

Provided that there is another user, the logged in user is shortly after login able to see a stack of cards, each card representing a different user.

The logged in user is able to interact with each card – either swipe it left to hide and never show the user again or swipe right to express their interest.

2. Application should allow user to "match" with another one

Explanation:

As explained earlier, to express interest in another user, he or she simply needs to swipe the card right. If both users express mutual interest in each other, they “match”.

Justification:

In order for two users to be able to send each other a message, they need to match first. (Please note Chat system will be developed in the following sprint.)

Priority:

Essential

Acceptance criteria:

When two users express interest in each other, they match instantly and a confirmation toast message is shown.

When user swipes left with another user, that user will not be displayed for interaction again even after reopening the application.

When user swipes right with another user, that user will not be displayed for interaction again, unless they swipe right too, in which case they match.

3. Application should provide user with a list of all their matches

Explanation:

When user matches with one or more other users, he or she needs to be able to see them all in one place so that he or she can contact them. Therefore, the Matches screen displays all the matches.

Justification:

In order for user to be able to get in touch with their matches, they first need to be listed on the Matches screen.

Priority:

Essential

Acceptance criteria:

User can see their match(es) on the Matches screen instantly after the match has occurred.

4. Application should allow user to log out

Explanation:

Naturally, as well as providing log in functionality, the application should also allow user to log out of their account.

Justification:

User may need to log out from their account in a number of reasons, particularly after finishing a session on their friend's mobile device or if they want to let other person log in.

Priority:

Desirable

Acceptance criteria:

User is successfully signed out of their account and the Welcome screen appears.

6.2.2. Non-functional

1. Search system should be quick

Explanation:

Provided that user has a standard speed of Internet connection, it should not take for the application unreasonable amount of time to load and display other available users.

Justification:

As of now, Search system is the main functionality of the application and the more user has to wait, the bigger chances of him or her giving up, closing and never using it again.

Priority:

Essential

Acceptance criteria:

User is able to see other users without having to wait unreasonable amount of time.

6.3. User story & tasks

User story

As a person keen to improve my fitness, I want to be able to connect with other like-minded individuals so that we can pursue our fitness goals together and keep each other motivated.

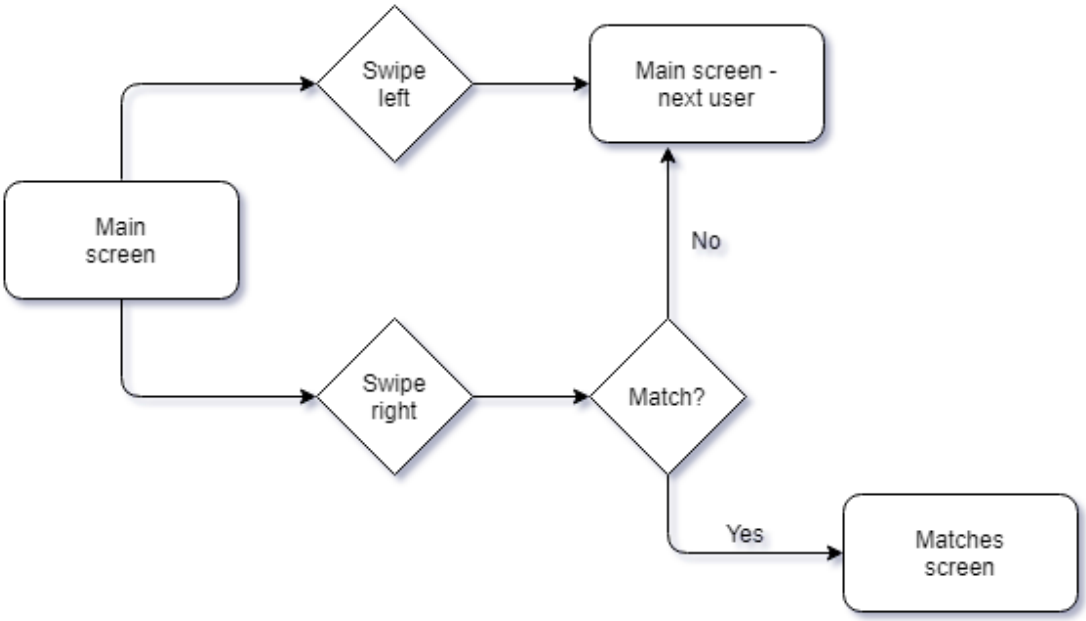
Tasks

Search system – develop functionality which displays a list of other users in a card layout stacked on top of each other.

Match system - develop functionality to determine a match and to handle "interested" as well as "not interested" interactions.

6.4. Design documentation

6.4.1. Flow chart



Flow chart 2: Search and match system

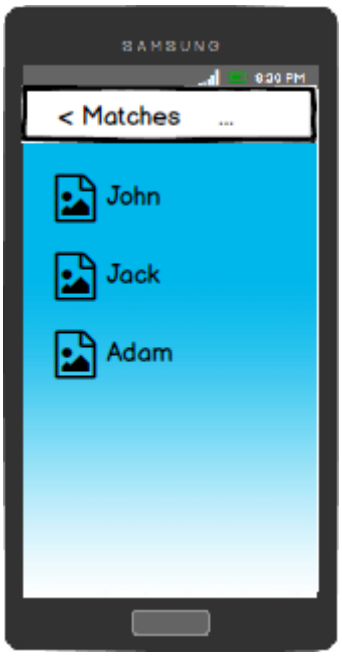
6.4.2. Mock-ups

Main screen:

Mock-up 3: Main activity screen

Once user logs in or is already logged in and opens the application, the Main screen is displayed. In the top, there is a menu with three icons – Matches, Activity Tracker and Settings (from left to right). A big layout element symbolising the "swipable" user card dominates the rest of the screen. It consists of a user profile picture and their information including name, gender, age, location and main interest.

Matches screen:



Mock-up 4: Matches screen

By clicking on the Matches button, user is taken to the Matches screen containing a list of all their matches so far. Each match has their own profile picture as well as name.

6.5. Implementation

I am going to refer to the currently logged in user as “user” and users displayed on the card layout as “potential matches” or “users”.

High-level structure of the code for the first functional requirement:

1. check current user’s gender
2. display the potential matches - populate cards with users’ data
3. listen for interactions with the card
4. record those interactions in the database
5. after rights swipe, check if the interest is mutual

There was several options with regards to displaying users. I could either display them as a simple list, a scrollable grid of images, a set of points on the map or on separate card layouts stacked on each other. I ruled out the simple list straight away as it would not be visual and engaging enough. A set of points on the map representing the users’ whereabouts seemed like a good idea, but in order for the point to not take up half of the screen, it would need to be a very precise location. This would quite likely be an ethical issue – if I were the user, I would not like the idea that strangers can precisely see where I am. Additionally, it would give any foul people a number of options on how to use it in their favour, such as for thefts, stalking people, etc.

On the other hand, both the scrollable grid of users’ profile pictures as well as card layout seemed like the best way to go. Since the cards stacked on each other (each one representing one user) are currently heavily used among the most successful apps, I decided to implement that as well. I believe they are so popular thanks to its simplicity and almost zero effort needed from user – all they need to do is swipe left to indicate that they do not like the content inside the card or right to express their interest. Since this is so widely used, the likelihood is that even a completely new user will know how to interact with it with ease. Moreover, it allowed me to use city-level location rather than having to use very high granularity.

Now I could develop this swiping functionality myself, but I believe a good programmer should be “lazy” and should never try to reinvent the wheel unless they have a good reason for it. Considering I did not have any good reason and the time constraints of this project, I decided to use a third party library. After some research, I found several libraries which did just what I needed. I chose the Swipecards library from Lorentzos because of its simplicity and customisation options.

First of all, a dependency and import were needed so that I could use the library in my project:

```
dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.google.android.gms:play-services-location:11.8.0'
    testImplementation 'junit:junit:4.12'
    // swipe cards:
    implementation 'com.lorentzos.swipecards:library:1.0.9'
    implementation 'com.android.support:cardview-v7:26.1.0'
```

Code excerpt 5: Location services dependencies

```
import com.lorentzos.flingswipe.SwipeFlingAdapterView;
```

Code excerpt 6: Swipe cards dependency

Then, the application checks gender of the current user to determine which users to display. At this time, it shows users of opposite gender, but since many people actually prefer to work out or do activities with someone of the same gender, this will need to be addressed later on.

Once it filters which users to display (based on gender and interactions – those who user has already swiped with regardless if left or right should not be displayed as potential matches again), it loads their data from the database and displays each user on a separate card layout. In case there is more than one of these potential matches, they are stacked on each other. When user swipes with one card, that card goes away and the one below is shown.

Each card has its own listener which checks for any interaction, namely a left swipe, right swipe or click.

```java
@Override
public void onLeftCardExit(Object dataObject) {
    makeToast( ctx: MainActivity.this, s: "Nope!");
    // register the negative swipe into db:
    Card potentialMatchCard = (Card) dataObject;
    String potentialMatchUserID = potentialMatchCard.getUserID();
    mUsersDB.child(mCurrentUserOppositeSex)
            .child(potentialMatchUserID)
            .child("Connections")
            .child("NoMatch")
            .child(mCurrentUser.getUid())
            .setValue(true);
}

@Override
public void onRightCardExit(Object dataObject) {
    makeToast( ctx: MainActivity.this, s: "Yep!");
    // register the positive swipe into db:
    Card potentialMatchCard = (Card) dataObject;
    String potentialMatchUserID = potentialMatchCard.getUserID();
    mUsersDB.child(mCurrentUserOppositeSex)
            .child(potentialMatchUserID)
            .child("Connections")
            .child("YesMatch")
            .child(mCurrentUser.getUid())
            .setValue(true);
    // check for a match between the current user and potential match displayed on the card:
    checkConnectionMatch(potentialMatchUserID);
}
```

Code excerpt 7: Card interaction listeners

Each swipe is then registered to the database – the currently-logged-in-user's ID is written to the potential match's node under "NoMatch" or "YesMatch" node for the left or right swipe, respectively. On Click listener is not implemented at the moment, but will act as a "more information" button in the future which is not in scope of this university project.

Moreover, when user swipes right, the application also needs to check whether the other user did not swipe right too. If yes, there is a match and a short toast message is shown informing user that they are now able to text each other. Also, the ID of the person current user has just matched with is registered to the "Matched" node of the current user and vice versa so that Matches screen can be constructed easily later on in the program.

```
private void checkConnectionMatch(String potentialMatchUserID) {
    DatabaseReference currentUserConnectionsDB =
            mUsersDB
                    .child(mCurrentUserSex)
                    .child(mCurrentUser.getUid()).child("Connections")
                    .child("YesMatch")
                    .child(potentialMatchUserID);
    currentUserConnectionsDB.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()) {
                makeToast( cbc MainActivity.this,  s "You two matched! Don't wait and text your potential buddy now!");
                // register the match in the card user's db node:
                mUsersDB.child(mCurrentUserOppositeSex)
                        .child(dataSnapshot.getKey())
                        .child("Connections")
                        .child("Matched")
                        .child(mCurrentUser.getUid())
                        .setValue(true);
                // register the match in the current user's db node:
                mUsersDB.child(mCurrentUserSex)
                        .child(mCurrentUser.getUid())
                        .child("Connections")
                        .child("Matched")
                        .child(dataSnapshot.getKey())
                        .setValue(true);
            }
        }
```

Code excerpt 8: Match system

Allow matching – display matches

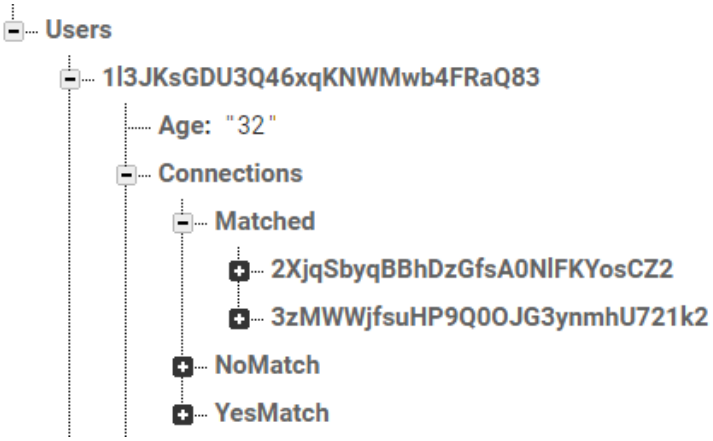
For the purposes of modularity and reusability, I created a blueprint for each match – the Match class:

```
public class Match {

    private String mMatchID, mMatchName, mMatchProfilePictureURL;

    public Match(String matchID, String matchName, String matchProfilePictureURL) {
        this.mMatchID = matchID;
        this.mMatchName = matchName;
        this.mMatchProfilePictureURL = matchProfilePictureURL;
    }

    // getters & setters:
    public String getMatchID() { return mMatchID; }
    public void setMatchID(String matchID) { this.mMatchID = matchID; }

    public String getMatchName() { return mMatchName; }
    public void setMatchName(String matchName) { this.mMatchName = matchName; }

    public String getMatchProfilePictureURL() { return mMatchProfilePictureURL; }
    public void setMatchProfilePictureURL(String MatchProfilePictureURL) {
        this.mMatchProfilePictureURL = MatchProfilePictureURL;
    }

}
```

Code excerpt 9: Match class

The Match class has three properties – ID, Name and Profile Picture. In addition, it also contains standard methods such as getters and setters for each of the property. Since each user can have unlimited number of matches and more properties will very likely be added in the future, it was an easy decision for me to go for the class structure.

In order for the application to be able to display matches, it needs to access the database and fetch all the relevant information for each match. The database is structured in the following way:



Code excerpt 10: Users database node

The "Matched" node contains the IDs of everyone user matched with. Hence, the application accesses those IDs and for each match ID, it calls the following function, passing it its ID:

```
private void fetchMatchInformation(String matchID) {
    mMatchDB = FirebaseDatabase.getInstance().getReference().child("Users").child(matchID);
    mMatchDB.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()) {
                String matchID = dataSnapshot.getKey();
                String matchName = "";
                String matchProfilePictureURL = "";
                if(dataSnapshot.child("Name").getValue() != null) {
                    matchName = dataSnapshot.child("Name").getValue().toString();
                }
                if(dataSnapshot.child("ProfilePictureURL").getValue() != null) {
                    matchProfilePictureURL = dataSnapshot.child("ProfilePictureURL").getValue().toString();
                }
                // create a new Match object & add it to the list:
                Match matchObject = new Match(matchID, matchName, matchProfilePictureURL);
                matchesList.add(matchObject);
                // notify the adapter of data change (onBindViewHolder is triggered):
                mMatchItemAdapter.notifyDataSetChanged();
            }
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });
}
```

Code excerpt 11: Fetching match information

This function retains the passed ID and on top of that, it also retrieves the name and profile picture of the match. As soon as it has this data, it creates a Match object and adds it to the list of all matches. Finally, it notifies the adapter that there has been a change in the data. This will be explained in more detail in the next few paragraphs.

I quickly realised how fast the program grew and was getting more and more complex. I started to struggle, so I chose to split the code into several functions. This made it much easier even for me to be able to search through the code quickly and be able to see the flow of the program. Moreover, it is also better for reusability and for other people to be able to follow the program flow without having to understand all the details. From this point onward, I tried to make use of functions whenever possible.

Now that the application gained the list of all matches with their relevant data, all that was left was to display them in a scrollable list. I expected this to be a simple task, but Android soon proved me wrong.

Firstly, I simply tried adding matches to the Scroll View dynamically using “for” cycle. However, it would not work even after countless number of times, hence I decided to research it. I did lots of reading and found out that there is a standard way of doing this using Viewholder Design Pattern. This would enable me to display matches in an efficient manner, allowing for smooth scrolling.

As a result, I structured my classes in such a way which I could use with the Viewholder pattern. I already had a Match class, so I only needed to add a Match viewholder and Match adapter.

```
public class MatchViewholder extends RecyclerView.ViewHolder implements View.OnClickListener {

    public TextView mMatchID, mMatchName;
    public ImageView mMatchProfilePicture;

    public MatchViewholder(View itemView) {
        super(itemView);
        // set an on click listener to every match:
        itemView.setOnClickListener(this);

        mMatchProfilePicture = itemView.findViewById(R.id.matches_iv_matchProfilePicture);
        mMatchID = itemView.findViewById(R.id.matches_tv_matchID);
        mMatchName = itemView.findViewById(R.id.matches_tv_matchName);
    }

    @Override
    public void onClick(View v) {
        Intent goChat = new Intent(v.getContext(), ChatActivity.class);
        Bundle chatBundle1 = new Bundle();
        chatBundle1.putString("matchID", mMatchID.getText().toString());
        goChat.putExtras(chatBundle1);
        v.getContext().startActivity(goChat);
    }
}
```

Code excerpt 12: Match viewholder

The viewholder puts every single match into a separate layout row. Also, when user clicks on any of the matches, it will redirect them to the Chat activity, thanks to the listener. Before that, however, the unique ID of the match is passed to the Chat activity as well in order for the application to be able to recognise whose chat should open.

Finally, Viewholder design pattern requires the application to provide the Item Adapter with all the matches in the form of a list. For each match in the list, it takes their name and profile picture and displays it in the relevant row.

Regarding the profile picture, I had problems loading it in the beginning, because it took too much space in the cache. I researched the issue and found out that there is a handy tool for effective loading of images called Glide and to my surprise, it was fairly easy to set up and use and it did solve my problem.

```
@Override
public void onBindViewHolder(MatchViewholder holder, int position) {
    holder.mMatchID.setText(mMatchesList.get(position).getMatchID());
    holder.mMatchName.setText(mMatchesList.get(position).getMatchName());
    if(!mMatchesList.get(position).getMatchProfilePictureURL().equals("default")) {
        Glide.with(mViewContext).load(mMatchesList.get(position).getMatchProfilePictureURL()).into(holder.mMatchProfilePicture);
    }
}

@Override
public int getItemCount() { return mMatchesList.size(); }
```

Code excerpt 13: onBindViewHolder method

User can now scroll through all their matches with ease.

Finally, I also implemented the functionality for logging user out. As can be seen from the below image, FireBase makes it very straightforward. Any time user hits the Logout button, the following function is triggered.

```
public void logOutUser(View view) {
    mAuth.signOut();
    String msg = getResources().getString(R.string.success_logout);
    makeToast( ctx: this, msg);
    Intent goWelcome = new Intent( packageContext: MainActivity.this, WelcomeActivity.class);
    finish();
    startActivity(goWelcome);
    return;
}
```

Code excerpt 14: User logout

It signs them out and redirects to the Welcome screen. Additionally, a toast message is shown to the user informing them that they have been successfully signed out.

6.6. Testing

requirement	acceptance criteria	passed/failed
FR 1	Provided that there is another user, the logged in user is shortly after login able to see a stack of cards, each card representing a different user.	passed
	The logged in user is able to interact with each card – either swipe it left to hide and never show the user again or swipe right to express their interest.	passed
FR 2	When two users express interest in each other, they match instantly and a confirmation toast message is shown.	passed
	When user swipes left with another user, that user will not be displayed for interaction again even after reopening the application.	passed
	When user swipes right with another user, that user will not be displayed for interaction again, unless they swipe right too, in which case they match.	passed
FR 3	User can see their match(es) on the Matches screen instantly after the match has occurred.	passed
FR 4	User is successfully signed out of their account and the Welcome screen appears.	passed
NFR 1	User is able to see other users without having to wait unreasonable amount of time.	passed

Table 3: Sprint two testing

6.7. Sprint evaluation

The second sprint was much more complex than the previous one and I also encountered a few more difficulties. What I struggled with most was trying to get my head around the Viewholder pattern design as I have never heard about or used it before. As I was researching this, I realised how much there is I do not know. This was quite frustrating as there was not enough time for me to learn about everything I would have loved to. However, this has helped me to learn prioritising things and focusing on what is inevitable only. Thanks to this, I managed to finish all the tasks from this sprint within the required time frame and could move on to the next one as planned.

7. Sprint three

7.1. Background

In the following sprint, I aimed to develop Chat and Location system. Since I did not bring forward any outstanding tasks from the previous sprint, I was able to delve right into it.

7.2. Requirements

7.2.1. Functional

1. Application should allow users who matched to message each other

Explanation:

When two users match, it indicates that they are interested in doing an activity together, be it swimming, dancing, working out or running.

Justification:

In order to be able to act upon that mutual interest, they need to be able to contact each other, get to know and potentially arrange a workout. Therefore, FitVenture provides them with chat functionality.

Priority:

Essential

Acceptance criteria:

When user goes to the Matches screen and clicks on any of the matches, a chat window opens.

User is able to send their match a message. Additionally, he or she is able to receive a message from their match.

2. Application should only display nearby users

Explanation:

Until now, the application displayed all users regardless of their location. This needs to be amended to only show the ones that are within proximity of the logged in user.

Justification:

In order for this application to be as effective and get as many users, couples or groups working out together as possible, it needs to enable them to find and connect with other people in their area. Nobody will travel half of the country just to work out with someone. This is also vital from the long term perspective – if they are to stick to

working out, they must be living close to each other otherwise it would not make too much sense and would not be as effective.

Priority:

Essential

Acceptance criteria:

On the Main screen, only people from the same area/region as the logged in user are displayed.

7.2.2. Non-functional

1. Chat system should be quick

Explanation:

Provided that user has a standard speed of Internet connection, it should not take unreasonable amount of time for the application to send, load and display messages between users. Ideally, this should be happening in real time with no delays.

Justification:

Nowadays, all the chat systems embedded in the most popular applications are real time. When a new product is launched, everybody expects it to be of at least the same quality, ideally better. Nobody would use this chat system if there were any delays in the delivery or reception of the messages.

Priority:

Essential

Acceptance criteria:

Once user sends a message, it is delivered and loaded to the other account in real time, with no delays.

2. Location system should be quick

Explanation:

When user logs in and is taken to the Main screen, their location is recorded and based on that, nearby users will be shown. This needs to be quick.

Justification:

As it stands now, this is the main functionality of the application and if user has to wait unreasonable amount of time for it to load, they are likely to close it and never use again. Also, they may think that there is no users nearby.

Priority:

Essential

Acceptance criteria:

The cards with potential matches are loaded and displayed within two seconds of opening the Main screen.

7.3. User story & tasks

User story

As a person keen to work out with my future fitness partner on a regular basis, I want to be able to only connect with other like-minded individuals in my close proximity. Moreover, I also want to be able to chat with them to get to know them a bit better and to arrange our first workout session.

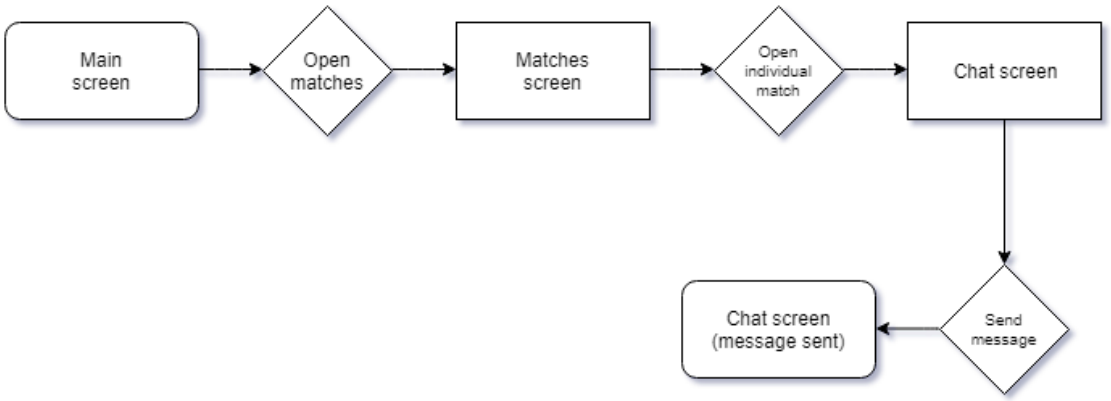
Tasks

Chat system – develop functionality which allows two users who expressed mutual interest to text each other.

Location system – amend the current search system in a way that the out of range users are filtered out.

7.4. Design documentation

7.4.1. Flow chart



Flow chart 3: Chat system

7.4.2. Mock-ups

Chat screen:

Mock-up 5: Chat screen

7.5. Implementation

Chat system

High-level structure of the code:

1. create a common Chat ID for both user and his or her match
2. take that Chat ID and create a separate node in the Chat list
3. every time a message is sent, create a child node in the Chat list under Chat ID and save all data in there
4. when the Chat screen is opened, load the chat content (if there is any)

```
o6ZNLkJ2WgequVdEjUD5RQmrCmF3
    Age: "22"
    Connections
        Matched
            BEnC4f1LToZHICRVmUwgOCPm1PY2
            yZRZlbA8NeRjBWI2X1b7f7ZtJy82
                ChatID: "-LADuRB1H6q0UZUOZ5y
        NoMatch
        YesMatch
```

Code excerpt 15: Unique chat ID

Once the unique Chat ID is created for both user and their match, it is then also stored as a separate node in the Chat list.

```
fitventure-f6e7d
    Chat
        -LADuRB1H6q0UZUOZ5yI
            -LADuSrfQMFIvA67ICo_
                MsgAuthorID: "o6ZNLkJ2WgequVdEjUD5RQmrCmF
                MsgDate: "16.4.2018
                MsgText: "ahoj'
                MsgTime: "16:44:12
            -LADumxar1XlSpmCNwNm
            -LADunduScHCtvHx8A-j
```

Code excerpt 16: Chat database node

Every single message has its own unique ID and apart from the content, it also stores information about when it was sent and by whom. All this information is very important. In sprint five, I used the information about message author to determine the styling of each message so that user can immediately see which were sent by him and which were sent by their match.

```
public void sendMessage(View view) {
    String sendMessageText = mMessageET.getText().toString();
    if(!sendMessageText.isEmpty()) {
        DatabaseReference newMessageDB = mChatDB.push();
        Map newMessage = new HashMap();
        newMessage.put("MsgAuthorID", mCurrentUserID);
        newMessage.put("MsgDate", mCurrentDate);
        newMessage.put("MsgTime", mCurrentTime);
        newMessage.put("MsgText", sendMessageText);

        newMessageDB.setValue(newMessage);
    }
    mMessageET.setText(null);
} // end of sendMessage()
```

Code excerpt 17: Sending a message

Currently, all the chats in the Matches screen are sorted by the time when two users matched. In the future, I am planning to use the date of the latest message instead to determine the order.

Once the message is sent and the above information written to the database, the recipient receives it in real time thanks to the function below.

```
private void getChatContent() {
    mChatDB.addChildEventListener(new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot dataSnapshot, String s) {
            if(dataSnapshot.exists()) {
                // fetch the data & assign to variables:
                String msgAuthorID = null;
                String msgText = null;

                if(dataSnapshot.child("MsgText").getValue() != null) {
                    msgText = dataSnapshot.child("MsgText").getValue().toString();
                }
                if(dataSnapshot.child("MsgAuthorID").getValue() != null) {
                    msgAuthorID = dataSnapshot.child("MsgAuthorID").getValue().toString();
                }
                // distinguish the current user & their match:
                if(msgText != null && msgAuthorID != null) {
                    Boolean isCurrentUser = false;
                    if(msgAuthorID.equals(mCurrentUserID)) {
                        isCurrentUser = true;
                    }
                    // create a new Chat object & pass it to the adapter:
                    Chat newMessage = new Chat(msgText, isCurrentUser);
                    chatList.add(newMessage);
                    mChatItemAdapter.notifyDataSetChanged();
                }

            }
        }
```

Code excerpt 18: Retrieving the chat content

Using the same approach as with the matches, the application loads the values from the database and creates an instance of Chat object. This instance is then added to the chat list and each element of the list is added to the Viewholder. Finally, the Adapter stitches it all together and displays a scrollable list of chat messages. Since I used the Viewholder Design Pattern in combination with the object list before, I was now confident it would work well here as well - and it truly did.

Location system

I expected this to take me no more than a couple of days. After all, I already had solid grounds with the existing Search system. The idea was to just build additional location functionality around it. However, I could not be further from truth.

My initial plan was to:

1. gain user permissions for tracking location
2. track user's location upon login (and then keep continuously tracking it every few seconds)
3. record each location change to the database
4. decode the location into user friendly format

5. create a list of users whose last recorded position was within a certain radius from the logged in user
6. before displaying users on cards, remove those who are not on the list created above
7. display users

This seemed fairly straightforward. Until I delved deeper into the very first task – tracking user's current location which proved to be very tricky. There is many challenges associated with regards to getting user's location and I had to consider three main ones: location source, user movement and accuracy. Since there is multiple sources of location available including for example Wi-Fi, GPS or cell towers, each providing different levels of accuracy and different levels of battery consumption, I had to decide what is most important for my application. As mentioned in Section: 2.5.1 Legal & ethical issues, I did not want my application to focus on the highest location granularity possible due to ethical issues. Moreover, users of my application would not be relying on precise location, as is often the case with, for example, taxi or ride provider applications such as Uber or Taxify. For the purposes of getting to know and working out with other nearby users, all they need is a location at the city level and possibly region level precision. This made my decision process easier and led me to give battery consumption priority over accuracy. Moreover, the safety risks are minimised this way as user cannot track another user with this level of accuracy. Once I considered all these things and had a clear idea of what I want to get from this, I started coding it.

Getting location permissions

Since each application runs in its own sandbox with limited access to resources outside of that sandbox, my application needed to ask for user's permission first to track the location. First of all, the access was requested in the Manifest file.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Code excerpt 19: Location permissions

Secondly, since location is considered as a "dangerous" permission in Android and the system does not automatically grant it as it does with others, less sensitive permissions, the application needs to ask for that permission once more at run time. Moreover, it needs to check if it still has that permission every time a task using location is performed.

However, this only came into effect with the release of the sixth major version of Android – Android Marshmallow. On any older version, user grants all the permissions upon download. Even though this approach was easier for the developers, countless number of applications abused this as user could not control each permission separately. That is why Android decided to rework the entire permissions architecture. For instance, a calculator application would require permissions for tracking location, reading contacts, accessing and modifying storage and others. This data would then be sold on black market.

Hence, every time user logs in and the Main screen is opened, my application first checks if the Android version is Marshmallow or higher and then tests if user has already granted it the permission to track location. If not, an explanation for why the application needs to track the

location is displayed. Once user closes that explanation dialog, another window pops out prompting user to actually grant it. Once granted, the application is able to actually request the location.

```
// if OS is Marshmallow or higher (and requires run time permissions):
if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    // check for permission:
    if (ContextCompat.checkSelfPermission( context: MainActivity.this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        if (ActivityCompat.shouldShowRequestPermissionRationale((Activity) MainActivity.this, Manifest.permission.ACCESS_FINE_LOCATION)) {
            // show an explanation dialog:
            new AlertDialog.Builder( context: this)
                    .setTitle("Location Permission")
                    .setMessage("To be able to find users in your area, please give us permission to get your location.")
                    .setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {
                            // request permission once the explanation has been shown:
                            ActivityCompat.requestPermissions((Activity) MainActivity.this,
                                    new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, MY_PERMISSIONS_REQUEST_FINE_LOCATION);
                        }
                    }).create().show();
        } else {
            // request permission:
            ActivityCompat.requestPermissions((Activity) MainActivity.this,
                    new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, MY_PERMISSIONS_REQUEST_FINE_LOCATION);
        }
        return;
    } else {
        // permission has been granted before, request the location:
```

Code excerpt 20: Check & request location permissions

Once the application was granted all the permission it needed, I started to learn how to get user's current location.

There were basically two options I could choose from. I could use Fused Location API or Fused Location Provider Client along with Google Services. The first one seemed much simpler and more straightforward to use, therefore I decided to use that one.

Getting user location – Fused Location API

With Fused Location API, the application needs to connect to it first.

```
mGoogleApiClient = new GoogleApiClient.Builder( context: this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
mGoogleApiClient.connect();
```

Code excerpt 21: Fused Location API

Once it is connected, the application makes a location request and determines how often the location should be updated and what the priority should be (i.e. accuracy or battery saving). I chose to prioritise battery saving over accuracy.

```
@Override
public void onConnected(@Nullable Bundle bundle) {
    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(120 * 1000);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_LOW_POWER);
    mLocationRequest.setFastestInterval(100 * 1000);
```

Code excerpt 22: onConnected callback

After this, my application asks for permissions to track the location. Once they have been granted, location updates are requested.

```
} else {
    LocationServices.
            FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest, locationListener: this);
    return
}
'FusedLocationApi' is deprecated more... (Ctrl+F1)
```

Code excerpt 23: Fused Location API is deprecated

However, as can be seen from the above picture, Fused Location API is now deprecated and will soon be unsupported by Android. I could potentially ignore this for the purposes of my university project (even though it would not be regarded as good practice), but since I am planning to further develop the application even after university, it was important for me to keep coding in a way that considers the future as much as possible. Hence, I decided that I would go for the more complicated recommended approach.

Getting user location – Fused Location Provider Client

Fused Location Provider Client (FLPC) is much more powerful than Fused API but since it still is relatively new, there is naturally less documentation available. Apart from the official Google documentation, there is no other decent source. And as is often the case with official documentation, it is difficult to understand and implement without any prior experience. I struggled a lot and went through many unsuccessful attempts before I finally managed to get it working.

First of all, FLPC needs to be created using Location Services which are part of Google Services. As with Fused API, intervals and priority is set up in the beginning as well.

```
mFusedLocationProviderClient = LocationServices.getFusedLocationProviderClient( activity: this);
mLocationRequest = new LocationRequest();
mLocationRequest.setInterval(5 * 1000);
mLocationRequest.setPriority(LocationRequest.PRIORITY_LOW_POWER);
mLocationRequest.setFastestInterval(2 * 1000);
```

Code excerpt 24: Setting FLPC

Again, once the permissions have been granted, the location updates are requested as follows:

```
mFusedLocationProviderClient.requestLocationUpdates(mLocationRequest, mLocationCallback, looper: null);
```

Code excerpt 25: Requesting location updates

The application is now able to receive those location updates in the Location Callback which gets triggered every five seconds, as set per the interval.

Writing location to the database

To be able to record the location to the database, I used an open source library called GeoFire. It is designed to store the location object simply as a latitude and longitude string pair.

```
LocationCallback mLocationCallback = new LocationCallback() {
    @Override
    public void onLocationResult(LocationResult locationResult) {
        for(Location location : locationResult.getLocations()) {
            mCurrentLocation = location;
            // write current location to geofire:
            mGeofireDB = FirebaseDatabase.getInstance().getReference( s: "Locations");
            GeoFire geofire = new GeoFire(mGeofireDB);
            geofire.setLocation(mCurrentUserID, new GeoLocation(mCurrentLocation.getLatitude(), mCurrentLocation.getLongitude()),
                    new GeoFire.CompletionListener() {
                @Override
                public void onComplete(String key, DatabaseError error) {
                    if (error != null) {
                        Toast.makeText(getApplicationContext(),  text: "There was an error saving the location to GeoFire: " + error,
                                Toast.LENGTH_SHORT).show();
                    } else {
                        Toast.makeText(getApplicationContext(),  text: "Location saved on server successfully!", Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
```

Code excerpt 26: Saving location to the database

Hence, once the application receives the location, it can be accessed from within the location callback. From there, it writes user location to the database. Each user has their own node within *Locations* which is equal to their ID and therefore acts as a reference between *Users* and *Locations* list.

```
Locations
    QbBQjwDNhPh2wCRySPhfly6M6Zw2
        g: "gcjszukr7c
        l
            0: 51.484558
            1: -3.169042
```

Code excerpt 27: Locations database node

For the current purposes and scope, the application rewrites the location every time a new location has been found. This way, only the last known position is stored which is perfectly sufficient for now. However, as far as the future and tracking location-reliant activities (such as running, hiking or cycling) are concerned, I will need to store every single location change and it will grow huge over time. That is also why I decided to separate users from their locations in the database.

Creating a list of nearby users

GeoFire also allows to query the location data and to find out which users are nearby (i.e. their last known location is within a certain radius from the current location of the logged in user).

First of all, the application tells GeoFire to query around the current location of user with the radius set to a fixed value of 30 kilometres. Several listeners is then attached, but my application only uses three. If GeoFire detects a new user in the radius, onKeyEntered() listener is triggered and that user is added to the list.

```
private void getNearbyUsers() {
    GeoQuery geoQuery = geoFire.queryAtLocation(new GeoLocation(mCurrentLocation.getLatitude(), mCurrentLocation.getLongitude()),
            mCurrentUserProximityRadius);
    geoQuery.addGeoQueryEventListener(new GeoQueryEventListener() {
        // user has been found within the radius:
        @Override
        public void onKeyEntered(String key, GeoLocation location) {
            nearbyUsersList.add(key);
        }
```

Code excerpt 28: Detecting a new nearby user

If GeoFire detects that any user comes out of the radius, onKeyExited() is triggered and that user is removed from the list.

```
@Override
public void onKeyExited(String key) {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        mCardList.removeIf(obj -> obj.getUserID().equals(key));
        mCardArrayAdapter.notifyDataSetChanged();
    } else {

    }
}
```

Code excerpt 29: Removing user from the nearby list

Once GeoFire cannot detect any new users in the radius, onGeoQueryReady() listener triggers the following function:

```
// all users within the radius have been identified:
@Override
public void onGeoQueryReady() { displayPotentialMatches(); }
```

Code excerpt 30: All nearby users are detected

This function now also filters out any user whose ID is not on the nearby users list.

However, to my surprise, as soon as I started using GeoFire, the application started to behave unexpectedly and I started getting weird results. For example, the application would display duplicate users – when user swiped another user, the same user would appear on the next card again. When user registered and was redirected to the main screen automatically, the duplicate users appeared twice. However, when user logged out and logged in, the duplicate users appeared for many more times.

Since I did not manage to solve this issue by the end of Sprint Four, everything is explained in the subsequent sprint.

7.6. Testing

requirement	acceptance criteria	passed/failed
FR 1	When user goes to the Matches screen and clicks on any of the matches, a chat window opens.	passed
	User is able to send their match a message. Additionally, he or she is able to receive a message from their match.	passed
FR 2	On the Main screen, only people from the same area/region as the logged in user are displayed.	passed with duplicates and unexpected results
NFR 1	Once user sends a message, it is delivered and loaded to the other account in real time, with no delays.	passed
NFR 2	The cards with potential matches are loaded and displayed within two seconds of opening the Main screen.	passed

Table 4: Sprint three testing

7.7. Sprint evaluation

This was so far the most complex and challenging sprint. I managed to complete the Chat system successfully without any major issues. However, it was very different with user location. Not only was all this new to me, but I also could not find any decent and complete sources other than the official documentation. As a result, it took me quite a long time to get the current location of user working. Lastly, I managed to only display nearby users, but with lots of unexpected results and duplicates. Therefore, I had to transfer the last task to the following sprint.

8. Sprint four

8.1. Background

For this sprint, I brought forward the unfinished location system. In addition, Settings functionality was planned.

8.2. Requirements

8.2.1. Functional

1. Application should allow user to change their profile settings

Explanation:

During registration, user only fills in the very basic information such as name, surname and gender. On top of these, the application settings should allow user to change their profile picture, main fitness interest and date of birth.

Justification:

To attract other users, get noticed and reveal a little bit more about themselves, users should be able to customise their profile. This will naturally increase the number of responses that are relevant and have a higher chance of getting to work out together. For instance, a tennis player may want to interact with other tennis players rather than with gym goers.

Priority:

Desirable

Acceptance criteria:

User is able to change their profile picture, name, surname, main area of fitness interest and date of birth in the Settings screen.

8.3. User story & tasks

User story

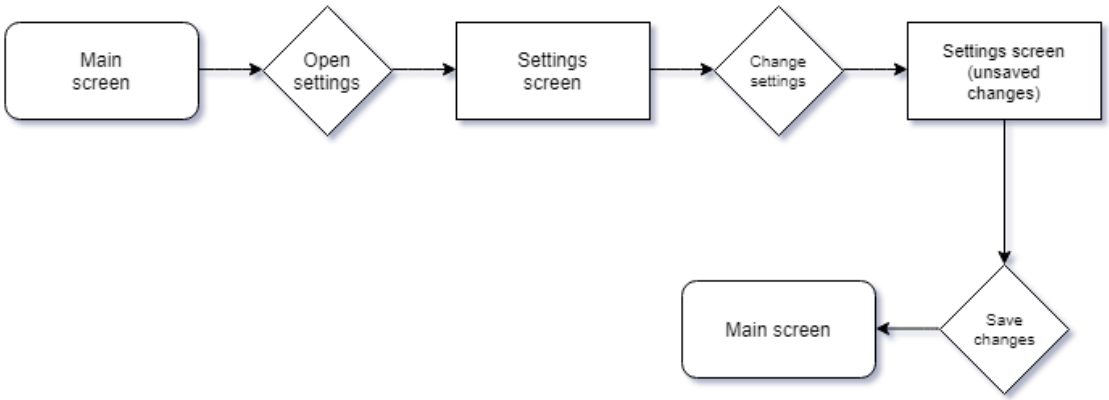
As a user who just started using FitVenture, I would like to be able to customise my profile so that I can attract more relevant users and responses.

Tasks

Settings – develop functionality to allow users change their profile picture, name, surname, main area of fitness interest and date of birth.

8.4. Design documentation

8.4.1. Flow chart



Flow chart 4: Settings system

8.4.2. Mock-ups

Settings

Mock-up 6: Settings screen

8.5. Implementation

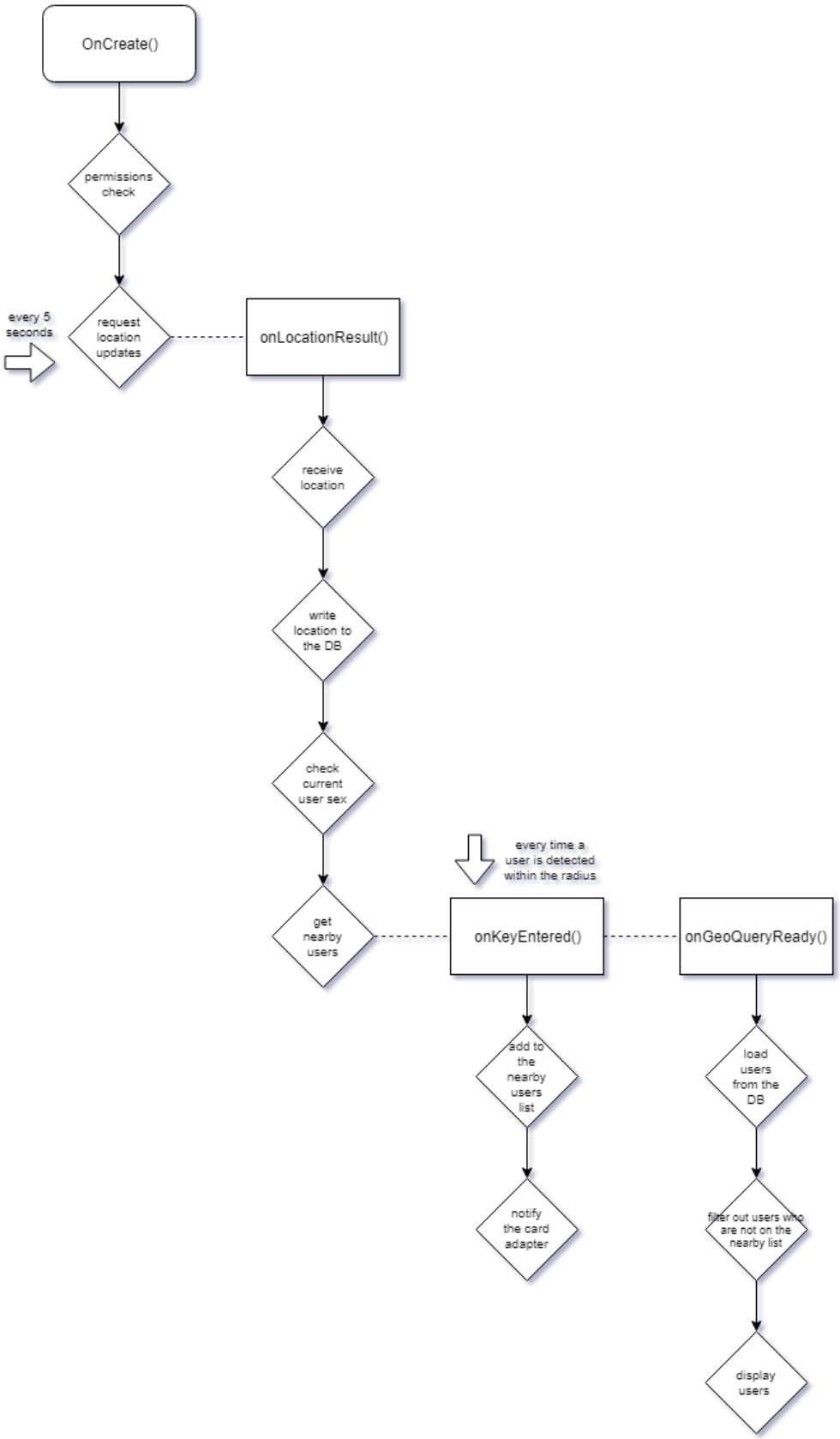
Location – continued from the previous sprint

After debugging the application for long hours, I found out that the number of duplicate users was not random. For example, when there was in total twenty users who were found in the radius and were to be displayed on cards as potential matches for the current user,

- the first found user was displayed on twenty cards,
- the second found user on nineteen cards,
- the third found user on eighteen cards,
- ….
- the twentieth user on one card.

If we suppose **X** is the final number of users found within the radius and **Y** is the order in which they are found (starting with 0), then each user was displayed on exactly **X minus Y** cards.

Hence, I started to examine the entire process and order in which the actions were executed. I created a flow chart (see on the next page) to help me visualise the code which was getting very complex, especially with all those listeners. I suspected that some of those listeners were clashing.



Flow chart 5: Original location system

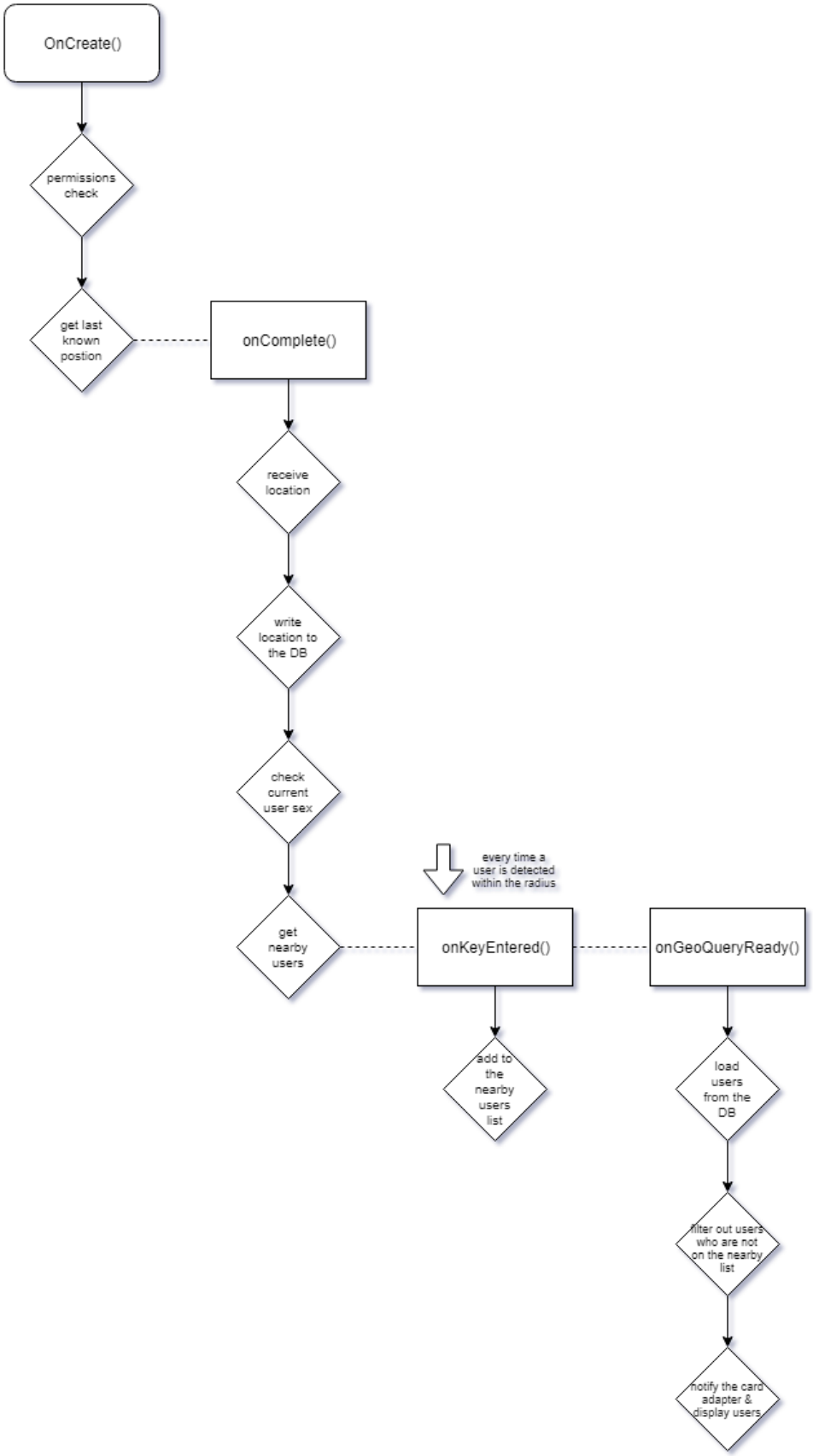
As can be seen from the above diagram, once the permissions are granted, the whole process flow gets executed every five seconds as that is the interval I had set for requesting location updates.

The second listener - onKeyEntered() – gets triggered every time a new user is detected within the radius. And every time this happens, the application not only adds that person to the list of all nearby users, but it also updates the card adapter.

The third listener – onGeoQueryReady() – is triggered once all the users within the radius have been found. Afterwards, all users are loaded from the database and then those who are not on the list of nearby users are filtered out. Finally, the card adapter passes the data to cards which are then displayed on screen.

Looking at the diagram has helped me realise that there is one apparent flaw in the process – the adapter being notified of change and updated every time a new user was detected in the radius. Once I moved the update of adapter to the onGeoQuery() listener, the users stopped being displayed *X-Y* times.

However, even after that, users would sometimes be displayed twice. After further examining the process flow, I figured this was happening because the onKeyEntered() was sometimes being fired without it first being finished, depending on how many users there were and how fast it detected them (i.e. depending on if it managed to detect them within the interval of 5 seconds). In other words, onLocationResult() and its branch as well as onKeyEntered() and its branch would from time to time overlap by running at the same time – creating duplicates as a result. Therefore, I decided to rework the way in which location is tracked. See my revised diagram on the next page.



Flow chart 6: Revised location system

Instead of getting location at regular intervals, I decided the location would only be requested once. It was not until this diagram that I realised that for the purposes of my application, there is actually no need to track location at regular intervals. Since the location was only going to be displayed to other users at the city or region granularity at most anyway, there was no point in requesting it every few seconds or even minutes. In case user has moved a considerable distance in which case it matters, the location will update any time a user re-opens the application or the Main screen itself. Therefore, for example, if user goes to settings and then back to Main screen, the location will be requested and updated if needed.

```
} else {
    // permission has been granted before, request the location:
    mFusedLocationProviderClient.getLastLocation().
            addOnSuccessListener( activity: this, new OnSuccessListener<Location>() {
```

Code excerpt 31: Requesting last known location

Settings

For the Settings functionality, my rough plan looked like this:

1. display all the layout elements
2. link the layout elements to Java code
3. load the value of each element from the database
4. upload the new values to the database when the Save button is clicked

I managed to display and link all the layout elements without any problem. For the main area of fitness interest, I used a spinner and saved all the values into a separate string resource file. I did this on purpose so that it can be easily translated into multiple languages later on.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="ISM_array">
        <item>Running</item>
        <item>Crossfit</item>
        <item>Gym</item>
        <item>Calisthenics</item>
        <item>Dance</item>
        <item>Biking</item>
        <item>Hiking</item>
        <item>Walks</item>
    </string-array>
</resources>
```

Code excerpt 32: Array list of user interests

For the date of birth, I used something completely new to me – a date picker dialog. Firstly, the application shows the day, month and year in a standard Android date dialog and then a listener is set which takes the value user entered and transforms it into a string. This string is then displayed on screen and recorded to the database when user hits on the Save button.

```
// USER DOB & AGE:
mCurrentUserDOBTV.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Calendar dob = Calendar.getInstance();
        int dob_day = dob.get(Calendar.DAY_OF_MONTH);
        int dob_month = dob.get(Calendar.MONTH);
        int dob_year = dob.get(Calendar.YEAR);

        // show date dialog:
        DatePickerDialog dialog = new DatePickerDialog(
                context: SettingsActivity.this,
                mOnDateSetListener,
                dob_year,
                dob_month,
                dob_day
        );
        dialog.show();
    }
});

mOnDateSetListener = new DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker view, int dob_year, int dob_month, int dob_dayOfMonth) {
        dob_month += 1;
        // write out date of birth:
        mCurrentUserDOB = dob_dayOfMonth + "/" + dob_month + "/" + dob_year;
        mCurrentUserDOBTV.setText(mCurrentUserDOB);
        getAge(dob_year, dob_month, dob_dayOfMonth);
    }
```

Code excerpt 33: Date of birth

Moreover, the getAge() function computes the age of the user automatically based on their date of birth.

Changing profile picture required its own intent which redirects user to the gallery. And again a listener which checks for when user has selected a picture and then displays it on screen.

```
// current user change profile picture:
public void changeProfilePicture(View view) {
    Intent goGallery = new Intent(Intent.ACTION_PICK);
    goGallery.setType("image/*");
    startActivityForResult(goGallery, requestCode: 1);
} // end of changeProfilePicture()

// getting data from action intents:
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    // current user profile image:
    if (requestCode == 1 && resultCode == Activity.RESULT_OK) {
        final Uri profilePicURI = data.getData();
        mCurrentUserProfilePictureURI = profilePicURI;
        mProfilePictureView.setImageURI(mCurrentUserProfilePictureURI);
    }
}
```

Code excerpt 34: Changing profile picture

The location is taken from the database and displayed on screen, too.

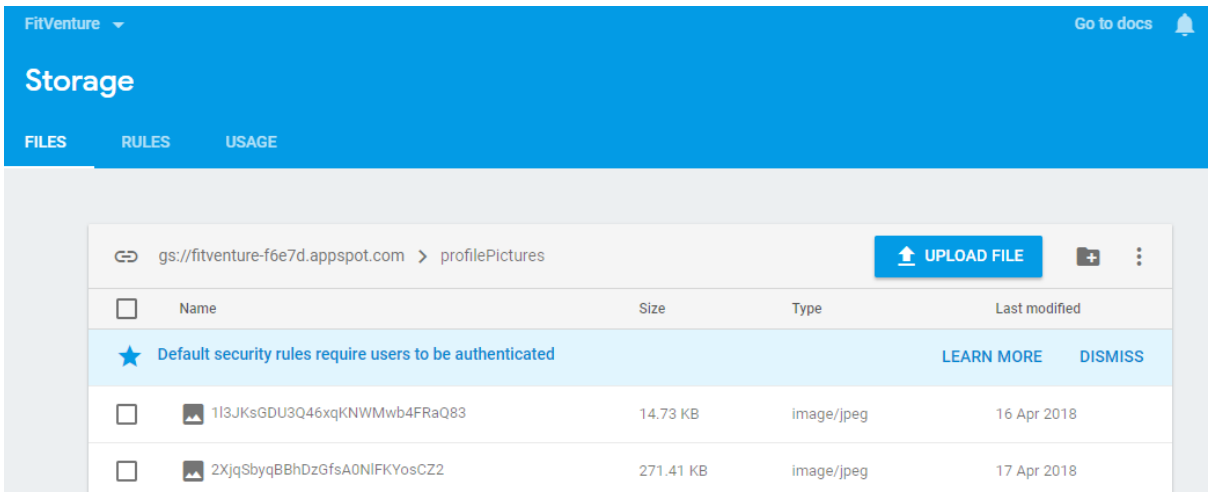
When user clicks on the Save button, all the data except for the location and profile picture are updated in a standard way using a hash map:

```
Map userInformation = new HashMap<>();
userInformation.put("Name", mCurrentUserName);
userInformation.put("Surname", mCurrentUserSurname);
userInformation.put("DateOfBirth", mCurrentUserDOB);
userInformation.put("Age", mCurrentUserAge);
userInformation.put("InterestsSportsMain", mCurrentUserInterestsSportsMain);
mCurrentUserDB.updateChildren(userInformation);
```

Code excerpt 35: Update the database with new information

Location cannot be changed manually by user so that data is not uploaded to the database.

Regarding the profile picture, it gets a bit more complicated. The application first checks if the profile picture URL is different from what is saved in the database so that it would not upload it unnecessarily. After that, it tries to upload it on the part of the server which is specifically designed for this purposes, called FireBase Storage.



Code excerpt 36: FireBase Storage

No matter how many times I tried though, it kept failing at the upload stage. I then researched the error and found out that there are certain file size limits, hence I had to compress the picture before uploading it.

```
// compress the profile picture:
ByteArrayOutputStream baos = new ByteArrayOutputStream();
bitmap.compress(Bitmap.CompressFormat.JPEG, quality: 20, baos);
byte imageData[] = baos.toByteArray();
```

Code excerpt 37: Compressing the profile picture

Once the photo has been uploaded successfully, the resulting URL gets updated in the database in the same way all other information was.

8.6. Testing

requirement	acceptance criteria	passed/failed

FR 2 – previous sprint	On the Main screen, only people from the same area/region as the logged in user are displayed.	passed
FR 1 – current sprint	User is able to change their profile picture, name, surname, main area of fitness interest and date of birth in the Settings screen.	passed

Table 5: Sprint four testing

8.7. Sprint evaluation

Since I had to bring forward the unresolved location system from the previous sprint, I had less time for developing Settings functionality. However, I managed to solve the location system anomalies in the middle of the sprint which gave me the entire week to implement Settings. During this, a new update of Android Studio became available for download. I decided to go for it and updated it. To my surprise, everything stopped working and from that moment, I was unable to run my project no matter what I tried. Since it was a brand new update, there was no help available regarding this issue on the Internet yet. After hours of trying, I downgraded the Studio and everything started working fine.

Despite these difficulties and the fact that most of the things I encountered were new to me and the time for development for Settings was halved, I managed to finish it by the end of week as planned.

9. Sprint five

9.1. Background

Since I only allocated a week for developing Activity tracker and a week for creating the UI and UX design of the entire application, I soon found out that this was not feasible and it took me almost the whole week to just design the tracker system solution. Since this was the last sprint and I could not afford to push the report writing, I decided to create a Balsamiq prototype to showcase the functionality instead. This will also come handy for future development outside the scope of this project.

9.2. Requirements

9.2.1. Functional

1. **Application should allow user to view, add and participate in a fitness programme**

Explanation:

Fitness programme is a set of one or more fitness routines. It defines what routines or activities should be performed at which day, how many times per week and for how long. User should be able to search through a list of existing programmes or create their own. Moreover, they should be able to enrol in one of their choosing.

Justification:

To attain a specific goal such as losing weight, gaining muscle or preparing for a marathon, user can enrol in a suitable programme which will ensure they keep on track towards their goal. Once set, the programme will insert the relevant trainings to the diary. This will help user stay focused and motivated. By seeing even the future workouts in the diary, not only will it help them set the right mindset and prioritise it over other things, but it will also greatly simplify the process of tracking the trainings. However, user does not need to be enrolled in any programme to track a one-off activity.

Priority:

Essential

Acceptance criteria:

User is able to view all their downloaded fitness programmes.

User is able to add a new programme by either downloading from the list of pre-existing ones or creating a completely new one.

User is able to enrol in a programme.

2. Application should allow user to view, add and modify a fitness routine

Explanation:

Fitness routine is a set of exercises forming one workout. It acts as a blueprint. Application should provide user with means of viewing, adding, modifying and creating their own routines.

Justification:

To simplify activity tracking and greatly speed things up, user can download or create several fitness routines consisting of multiple exercises based on their individual goals. Each routine can then be performed on different days.

Priority:

Essential

Acceptance criteria:

User is able to view all their downloaded fitness routines.

User is able to add a new routine by either downloading from the list of pre-existing ones, creating a completely new one or modifying an existing one.

3. Application should allow user to track activity

Explanation:

Whilst fitness programmes are in charge of scheduling the relevant routines in to the diary and fitness routines act as blueprints of what should be trained on each workout, the application should also provide user with means of actually tracking individual activities.

Justification:

In order for user to make progress and improve, they need to be able to record what they exercised and how many sets and repetitions with what weights they performed.

Priority:

Essential

Acceptance criteria:

User is able to track an activity. For each exercise in the activity, they are able to specify the number of sets and repetitions and weights they used.

4. Application should keep a log of past activities

Explanation:

User should be able to see the history of their activities with all the relevant details in the diary.

Justification:

In order for user to be able to improve at all times, they need to see how much they lifted during their last workout or how fast they ran in their last high intensity sprinting session. Moreover, in order for them to stay motivated, they need to be able to see how much effort they have already put into it.

Priority:

Essential

Acceptance criteria:

User is able to see both past and scheduled activities in their diary. By clicking on them, they are able to see the relevant details such as number of sets, repetitions and weights for each exercise.

9.2.2. Non-functional

1. Activity tracker should be easy to use

Explanation:

User should be able to navigate around the tracker without the need for any excessive documentation or guide.

Justification:

The whole point of an activity tracker on a mobile device is that it makes it so much simpler and more practical than having to carry a paper with a pen. However, if it is too complicated or difficult to use, it may very well deter user from using it.

Priority:

Desirable

Acceptance criteria:

User can track activity without the need for any excessive documentation about how to use the tracker.

9.3. User story & tasks

User story

As a person concerned with my fitness condition who always strives to improve, I would love to be able to track my activities so that I can stay motivated and improve gradually.

Tasks

Fitness programme system – develop functionality to allow users view, add, create and enrol in various fitness programmes depending on their personal goals.

Fitness routine system – develop functionality to allow users view, add, create new or modify existing routine.

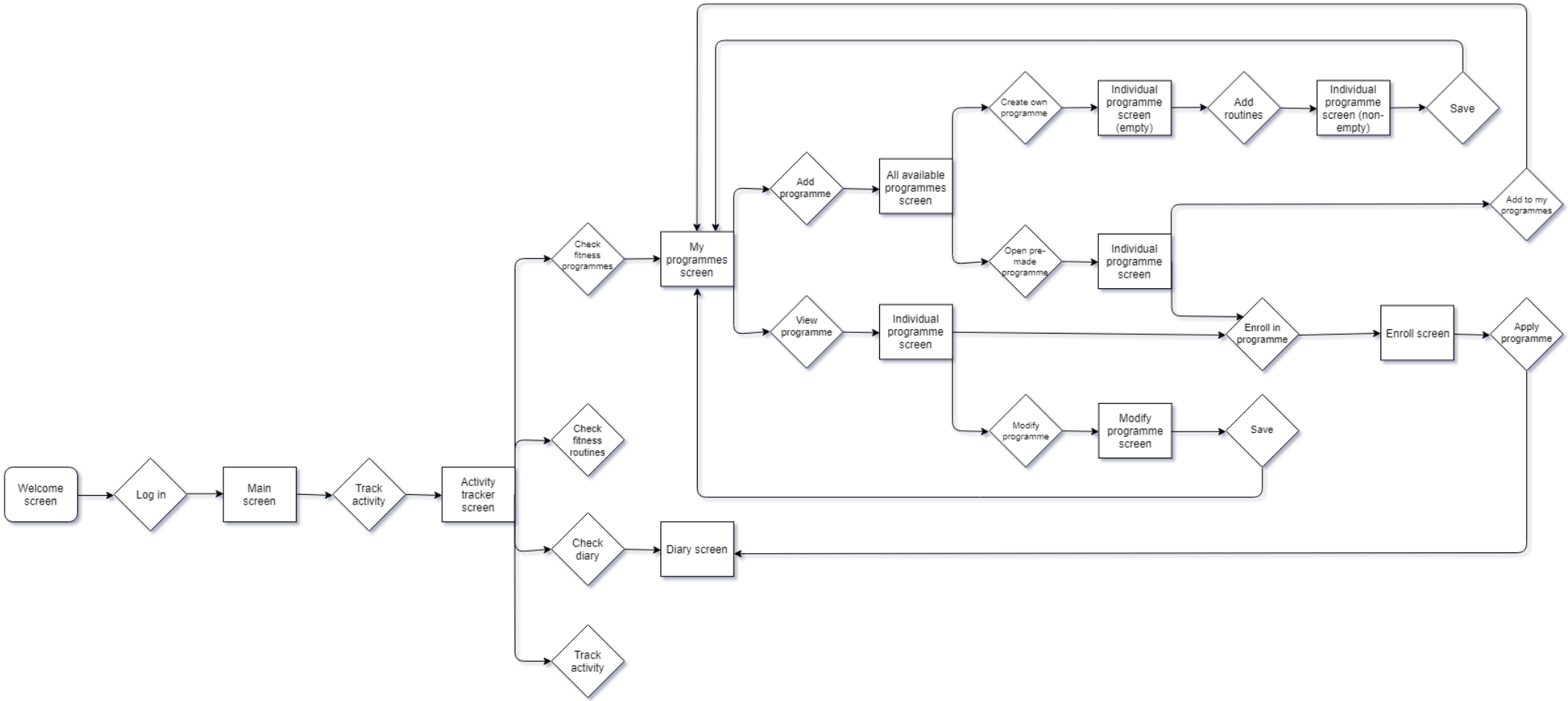
Fitness tracker system – develop functionality to provide users with means of tracking their activities, be it a calisthenics workout, a tennis match or a dancing session.

Fitness diary system – develop functionality to display both past and future activities on a simple diary.

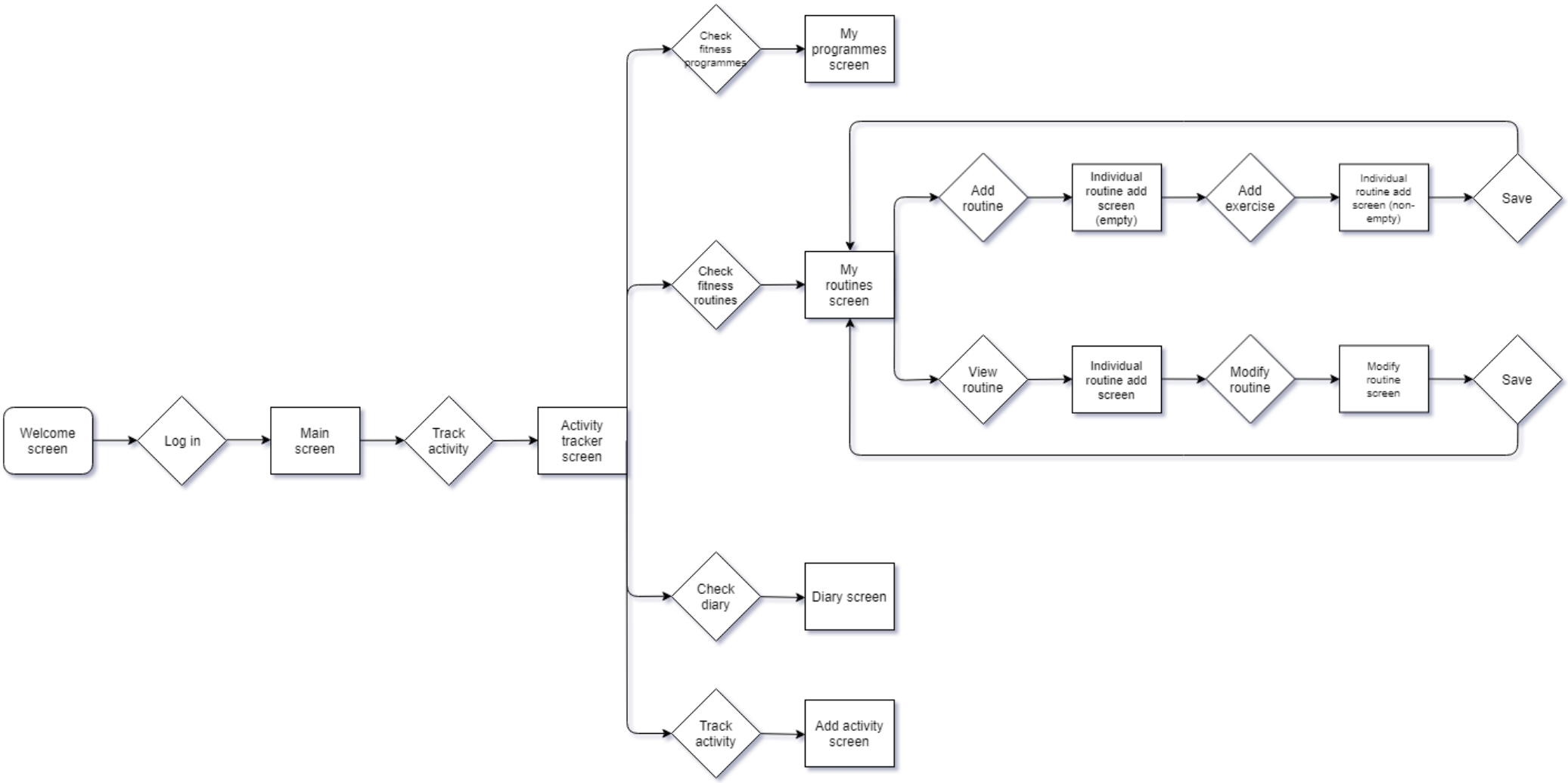
9.4. Design documentation

9.4.1. Flow chart

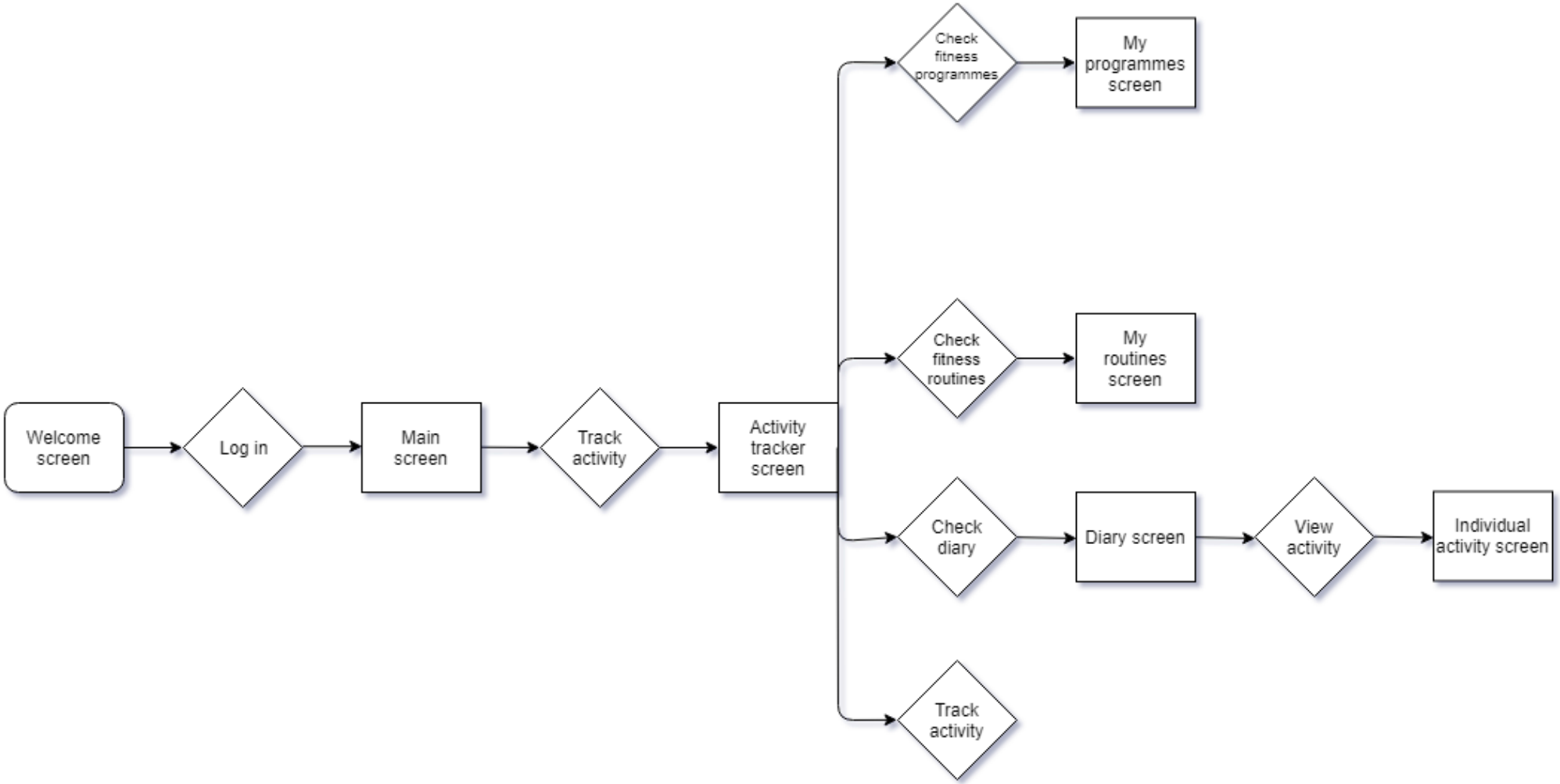
Please see the following pages.



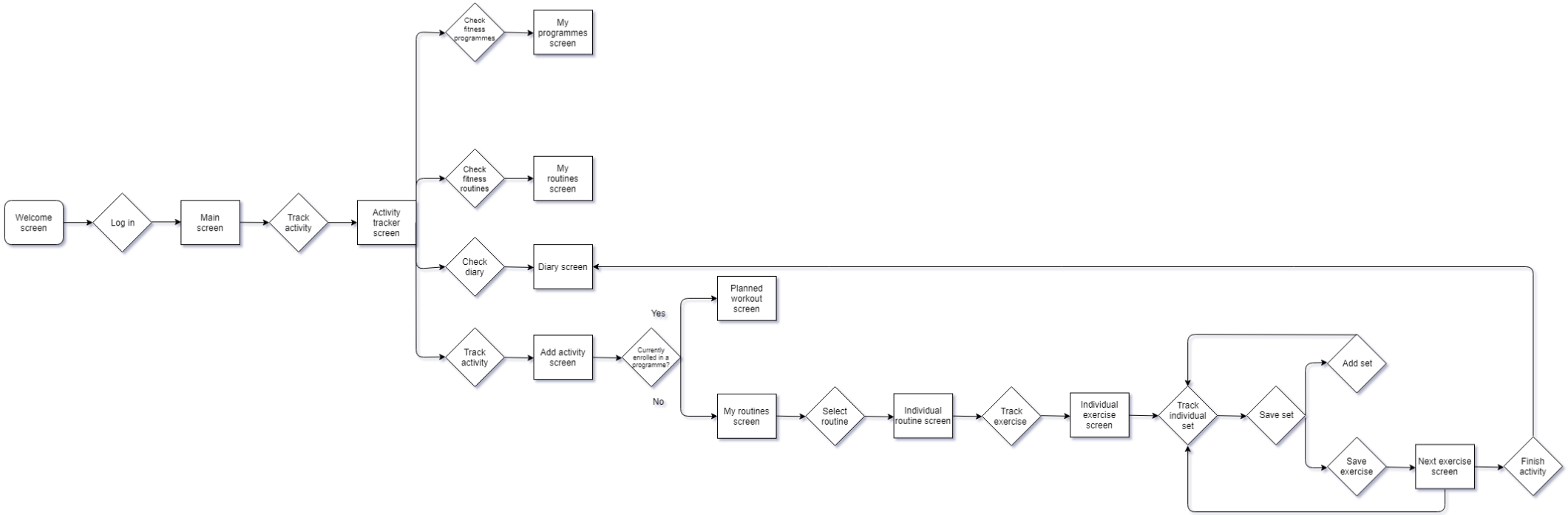
Flow chart 7: Fitness programmes



Flow chart 8: Fitness routines



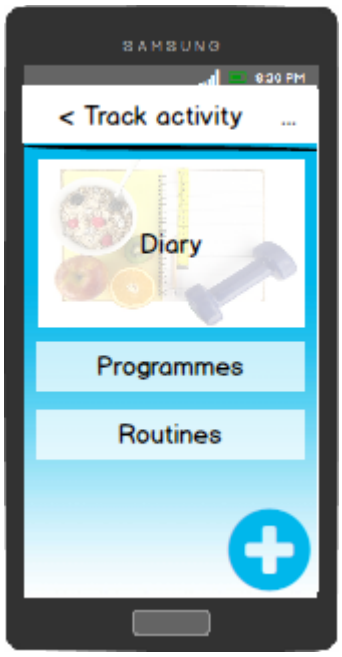
Flow chart 9: Diary



Flow chart 10: Activity tracker

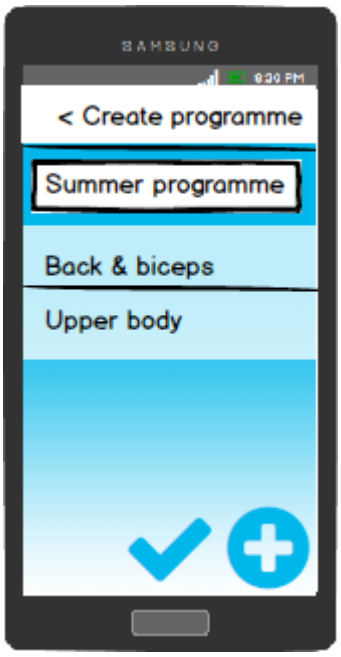
9.4.2. Mock-ups

Please see a sample of mock-ups below, the fully working prototype will be demoed during VIVA.



Mock-up 7: Activity tracker

Mock-up 8: Existing programme view



Mock-up 9: Create a new programme

9.5. Application UI & UX Design

Usually, users tend to uninstall an application not because of its (lack of) functionality, but because they get frustrated using it or because it fails to create an emotional response in them (Muellauer and Yu, 2018). Therefore, one cannot underestimate the impact of User Interface and Experience design. Having realised this, I dedicated the whole week just for creating and polishing the design, focusing on elements such as colour theory, typography, consistency or Android-specific design practices.

Welcome screen

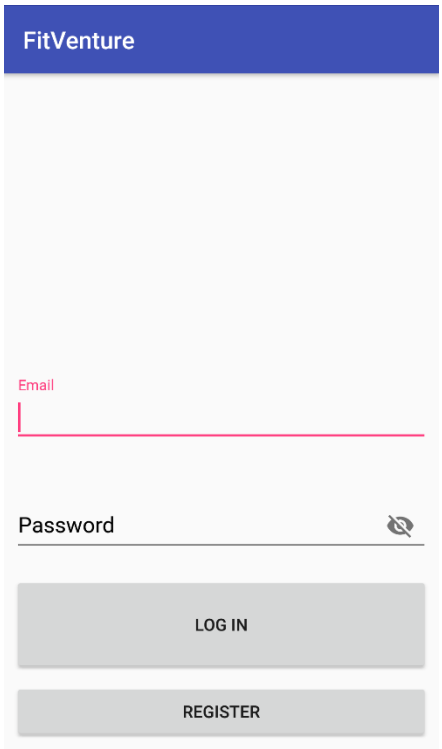


Image 5: Original welcome screen

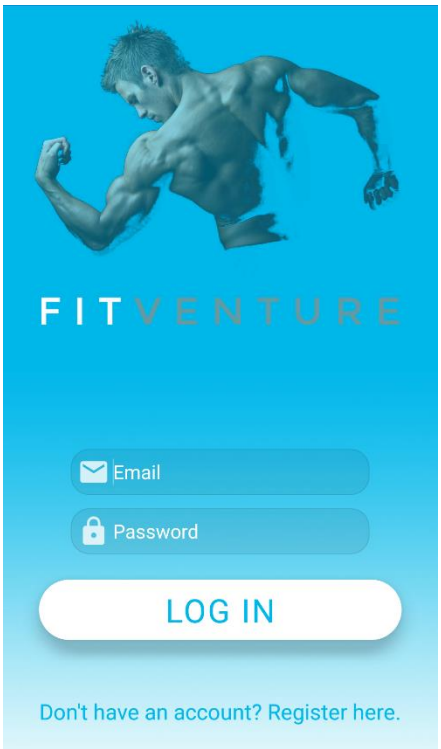


Image 6: Designed welcome screen

For the background, I chose a simple gradient using the same colour as the logo. In the top, the athlete image is followed by the application name. The image of an athlete has been chosen on purpose as some studies indicate that this can play a big role in boosting the motivation levels as mentioned in the 2.2 Problem analysis. Below these two elements, spacing is added to give the login form breathing space so that it can stand out and gain the user's attention. The login button below the form is dominating the screen and in combination with the less visible registration link acts as a call to action (CTA) button, drawing the user's attention towards the ultimate goal of this screen.

Main screen

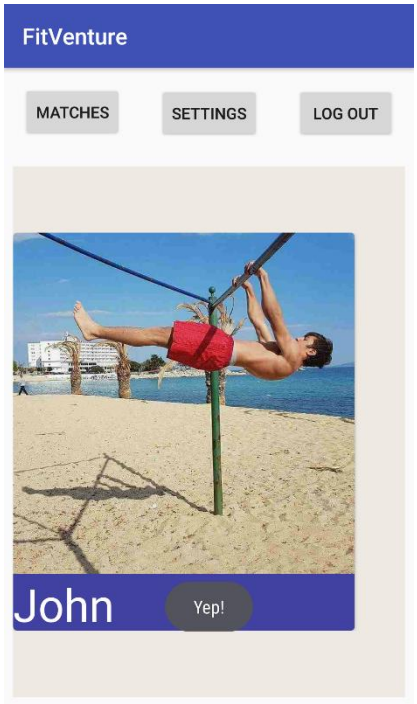


Image 7: Original main screen

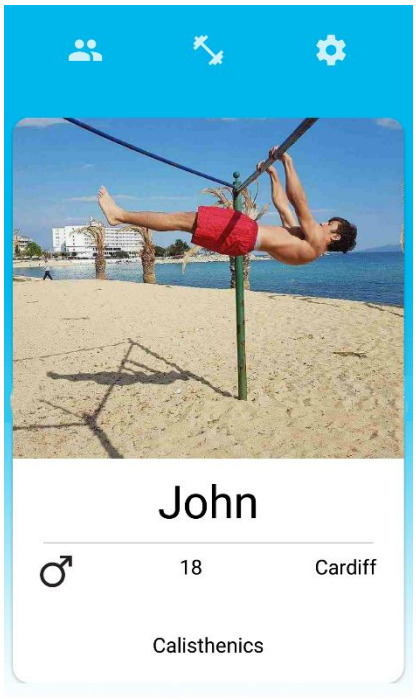


Image 8: Designed main screen

Regarding the menu, the three most common types are:

- sandwich menu (the one that can be swiped from either side)
- top screen menu (buttons laid out at top of the screen)
- bottom screen menu (buttons laid out at bottom of the screen)

Since most Android devices have one to three navigational buttons at the bottom of the screen or just below the screen the last approach was out of question. The best practice is to put menu either to top of the screen (as I did) or to use a sandwich approach (Muellauer and Yu, 2018). Since FitVenture only contained three buttons at the moment, I decided to go for the second option. The menu buttons have been chosen carefully to match their purpose as much as possible.

The biggest element on the screen is a “card” containing all the information about the potential fitness partner nearby. The rounded corners and elevation shadows help user to imagine that it is a card which they can interact with.

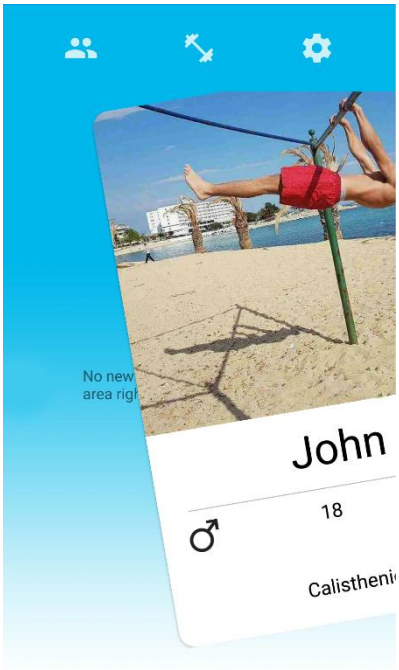


Image 9: Right swipe

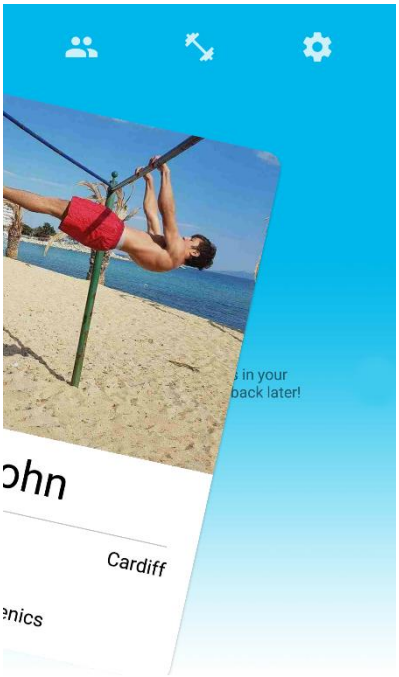


Image 10: Left swipe

The card can be swiped right to express interest or left to dismiss the card and never show again. The layout of the two was not random – all popular applications such as Tinder or Badoo are using right for interest and left for dismiss. Therefore, they have already created a pattern which should not be broken. Chances are some of the users will have used the other two applications before, thus they would get confused if it was the other way around.

9.6. Testing

requirement	acceptance criteria	passed/failed
FR 1	User is able to view all their downloaded fitness programmes.	not implemented
	User is able to add a new programme by either downloading from the list of pre-existing ones or creating a completely new one.	not implemented
	User is able to enrol in a programme.	not implemented
FR 2	User is able to view all their downloaded fitness routines.	not implemented
	User is able to add a new routine by either downloading from the list of pre-existing ones, creating a completely new one or modifying an existing one.	not implemented
FR 3	User is able to track an activity. For each exercise in the activity, they are able to specify the number of sets and repetitions and weights they used.	not implemented

FR 4	User is able to see both past and scheduled activities in their diary. By clicking on them, they are able to see the relevant details such as number of sets, repetitions and weights for each exercise.	not implemented
NFR 1	User can track activity without the need for any excessive documentation about how to use the tracker.	not implemented

Table 6: Sprint five testing

9.7. Sprint evaluation

I managed to research, design the solution and create a prototype in Balsamiq in a week and a half, leaving only half of the week for the UI and UX design. As a result of my inexperience with Android and Java, I would often get stuck and need to research the whole topic. Therefore, this proved to be unrealistic and it took me over a week to finish it. Hence, I finished this sprint a week later than projected.

10. User testing

10.1. Background

Once the application was finished, I created a survey and asked a small group of fitness enthusiasts who had prior experience using other fitness applications to test it and to provide me with honest feedback. The survey was aimed at testing the application's performance as well as usability.

10.2. Survey structure

The survey consisted of seven tasks.

Tasks:

1) Register to the app. Use fictional data for this.
2) Log in and change your profile picture.
3) Change your name and age. Again, use fictional data.
4) Express an interest in working out with another user of your choosing.
5) Skip a user of your choosing (express that you are NOT interested in working out with them).
6) Chat with any of the users you have matched with and agree on a workout.
7) Log out.

Image 11: Survey tasks

After completion of each task, the participants were asked the following questions.

TASK1 - Registration

Were you able to successfully complete the task?
On a scale 1-10, how easy was it to find the required task within the app?
On a scale 1-10, how easy was it to execute and complete the required task?
On a scale 1-10, how clear and timely (10 = VERY CLEAR) was the feedback the app gave you?
Did you experience any lags or speed issues?
Did you encounter any difficulties, crashes or bugs whilst performing the task? If so, when did they occur?
Do you think this task could be improved? If so, how?
Do you have any other feedback?

Image 12: Task questions

Finally, the last section allowed participants to rate the overall experience and to give any additional feedback.

On a scale 1-10 (10 = VERY COHERENT), how coherent is the app design overall?
On a scale 1-10 (10 = VERY EASY AND INTUITIVE), how easy is the app to use overall?
On a scale 1-10 (10 = LOVE IT!), how much do you like the app overall?
On a scale 1-10 (10 = VERY LIKELY), how likely would you be to download this app and use it once published? What would need changing or adding in order for you to consider downloading it?
On a scale 1-10 (10 = VERY LIKELY), how likely would you be to recommend this app to friends?
What do you like about this app in comparison with competition? What do you not like?
What other functionality/features would you welcome?
Do you have any other feedback?

Image 13: Overall feedback

For full answers, please see Appendix F.

10.3. Results & consideration of feedback

Task scores

These are the average scores for each task. The higher the score, the better.

task	easy to find	easy to execute
1. Registration	10	10
2. Log in & change profile picture	7.7	8
3. Change name & age	8.7	9.3
4. Express interest	7	5.7
5. Dismiss user	8.7	9.3
6. Chat with a match	9.3	9.7
7. Log out	10	10

Table 7: Survey results

As can be seen from the table, all tasks except the fourth have scored fairly high. The scores also indicate that I should focus on improving tasks number 4 and 2 first. The reason why the task number 5 has scored much higher than the task number 4 even though they are virtually the same is that some of the participants were not aware of how to interact with the user

card. However, once they found out, they learned it quickly. Since this was a fairly common issue, I decided to create visual cues to indicate how to interact with the card.

Image 14: Visual cues added

With a cross and a tick on each side, it is now clearer as how to interact with the card even to persons who have not used a similar application before.

Overall application scores

These are the average scores for overall experience with the application. The higher the score, the better.

area	score
How coherent and consistent is the design overall?	8.3
How easy is the application to use overall?	7.7
How much do you like the application overall?	9
Would you download it once released?	9
Would you recommend it to friends?	10

Table 8: Survey results of the overall application experience

These indicate that the participants like the idea of the application in general and are willing to recommend it to their friends. The usability scored a bit less mainly because of the issue mentioned above which I solved.

Bugs & issues

1. Back button crash

Once user logs out and presses the back button with intention to close the application, it tries to go back to the Main screen instead of closing. For a moment, it truly takes them there, but it is immediately followed with a crash. The crash is caused by the listener which ensures that only logged in users have access to that screen.

This will need to be researched and dealt with before launching the application.

2. Profile picture too big

From time to time, the application crashes when user uploads a profile picture which is too big.

Even though the application uses an external library for compressing and uploading the images, apparently some of the images are still too big and therefore I will need to look into settings of that library and see if it can be compressed even more without significantly losing the quality.

3. Messed up layout

On smaller screens (4.5" and less), some layout elements, especially the user information at the bottom of the user card, were covering each other.

A more thorough test of the user interface including all types and sizes of screen will need to be done before launching the application.

What users like about the application

Participants indicated that they like the whole idea of finding a fitness or sport partner without even having to step outside. In addition, they like the simplicity of the application as well as the fact that it performs well at what it is meant to do.

Suggestions for improvements

- after successful registration, add a short popup dialog listing the main benefits of using the application to draw user's curiosity even more
- create visual cues to indicate how to interact with the user cards
- once two users match, provide a more visual and explicit feedback
- create in-app notifications for both a new match and a new received message
- make the logout button smaller or move it beneath the save button
- on the main screen, display a loading animation until all the nearby users are found and loaded (currently, there is a "No new fitness addicts nearby" label shown for a very brief moment until the user cards are displayed. This led to confusion among a couple of survey participants.)
- in the chat screen of each individual match, add an option to view the user's profile

- in the chat screen of each individual match, add more details to the messages, such as when the message was delivered/received and read
- add more types of activities and fitness/sport level indicator
- add filtering options based on activity type and fitness/sport level

I consider collecting feedback as a great means for gaining insight and improving the application. Hence, I have taken all the feedback seriously and noted it down. However, due to time constraints, I only implemented the most pressing issue (the addition of visual hints on how to interact with the cards). The rest is out of scope of this project but will be worked on.

11. Evaluation

11.1. My own testing

As can be seen from each sprint's testing, all the functional and non-functional requirements except for the Activity Tracker have been implemented in the end, passing their relevant acceptance criteria.

As for Activity Tracker, I had to strip out its development, as there was not enough time and I could not risk pushing back report writing. However, I created a fully working high-fidelity prototype in Balsamiq which will be demoed during VIVA.

In summary, the MVP was meant to include the following functionality: find a fitness buddy (nearby), messaging system, user system, activity tracker.

Out of these, I managed to implement the first three features.

The "nice-to-have" features were not implemented due to time constraints.

Overall, I managed to develop a partly working solution with most of the essential functional requirements implemented. Considering the complexity, my degree course, time constraints and nature of the project, which was not purely about development, I consider it a success.

11.2. User testing

Once the development and UI & UX design were completed, I carefully prepared a survey collecting both qualitative and quantitative information about performance, usability and improvement suggestions. A small number of people were then asked to conduct user testing and fill in the survey.

Since it was out of scope of this project to release the application to the public, the whole purpose of this testing was to only give me an insight into how the application performs and how usable it is. However, further large scale thorough testing will need to be done before launch.

Despite the relatively small sample, the feedback gained from participants proved to be very useful, as they were carefully chosen to have slightly different demographics. For instance, one participant does not lead overly active lifestyle, another one did not have any prior experience of using such an application and the third one was a former athlete. In summary, a couple of bugs and more than dozen of suggestions for improvement and several suggestions for new features were identified.

Regarding performance, the application did well and users did not experience any speed or execution-related issues. However, two bugs were detected and noted down for future improvements.

With regards to usability, the application scored high as well, but all participants agreed on one main issue. The issue was the lack of visual cues as how to interact with the main element of the application – user cards. Therefore, the priority was given to this problem and small visual hints were added. Moreover, more than dozen of usability suggestions for improvement were identified.

All of this feedback will be learned from and used towards developing a production version of the application.

11.3. Future improvements

Apart from the improvements suggested by the survey participants, I identified these additional improvements, organised by their relevant area:

area	improvement	priority
registration	send an email to verify the account	high
registration	send an email to welcome user and confirm they account details	high
log in	provide more log in options, such as via social media	high
location	check if GPS is turned on and if not, redirect user to location settings	medium
location	in the onKeyExited() listener, support API level 23 and smaller when removing a list member	low
location	improve the algorithm that fetches all nearby users – get all users regardless of proximity radius, but sort them and display closest ones first	medium
search settings	provide option to filter out users based on sex preference, activity type and fitness level	high
chat	automatically scroll down to the latest message when the chat with multiple messages is opened	high
settings	allow user to terminate/delete an account	high
matches	order matches based on the last message received not the time of match	high
general	separate profile from settings (profile will contain things like reputation, badges and fitness programmes completed whereas settings will include things of technical nature such as resetting passwords, deleting account, etc.)	

Table 9: Future improvements

11.4. Project management

Originally (see Appendix G), I aimed to only have three two-week sprints with an additional time over Easer to complete anything unfinished. However, once I learned about and started working on all the necessary non-development activities as well, there was less time for development and it soon became clear that more sprints would be needed, even throughout the Easter.

Therefore, I quickly amended the plan in such a way that it would contain five sprints all together, spreading out the complexity more proportionally. I did this by reducing the allocated time for other tasks, such as user testing or bug fixing. With this new plan, I was able to finish all the sprints except for Sprint 3 in timely manner.

With hindsight, I can say I underestimated the complexity of each functionality and did not predict the right amount of time needed to finish each one. This was a great learning experience for me and in the future, I will try to be more realistic about my goals as well as taking my experience into consideration, too.

11.5. Reflection on learning

The last 11 weeks had a very steep learning curve for me during which I did not only gain a great deal of new skills and knowledge, but I also reinforced the existing ones and learned things about myself that I did not know before. The project also enabled me to put all the knowledge gained during the past few years at university into practice. All of the newly gained and reinforced skills will play an important role in my future.

Technical skills

Since I had no previous experience in mobile application development, I undertook a couple of courses on Android, Java and FireBase before the start of the development. However, even though the courses were very informative, no course can prepare one for everything. Therefore, I was essentially learning new things all the time on the go throughout the entire project. Every time I got stuck, I would research the topic first before asking for help.

Although I did have some knowledge about NoSQL databases thanks to several previous university courses, it was not until this project that I actually made use of this knowledge in real world. However, I still needed to research and learn about FireBase and its specific usage as there are plenty of different NoSQL databases, each having its own rules and specification.

By the end of the project, I gained skills in Android programming, FireBase database system, agile project management and tracking using Taiga (see Image 15 below) and also touched XML. I also improved significantly at software quality skills (such as maintainability or usability) and object-oriented programming. The whole project was coded in such a way (using classes, functions and variables where possible) that most parts of the code can easily

be modified in one place. I also ensured other best practices are adhered to – the code is also well commented, therefore anyone could pick the project up without any difficulties. Furthermore, the content was separated from the code, hence the application can easily be translated into multiple languages and all the graphics can be changed without having to go deep into code.

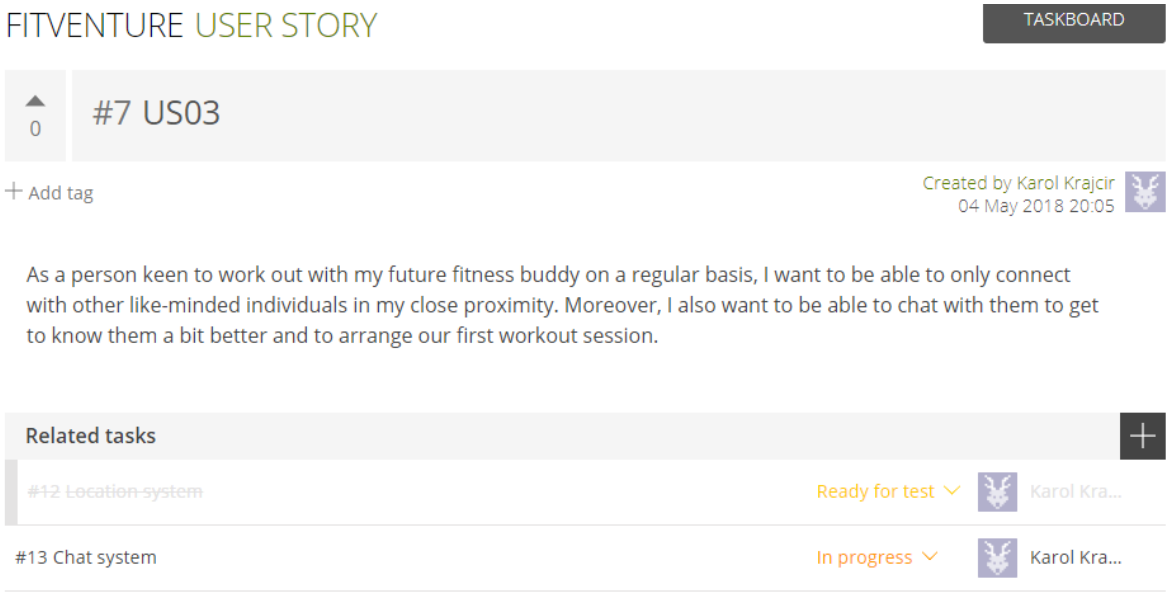


Image 15: User Story tracking in Taiga

Furthermore, I strengthened my WordPress development skills by amending an existing theme to build my blog as well as learning a lot about how to launch an application including idea validation, principles of marketing and PlayStore algorithm.

Professional skills

Undertaking such a complex project just on my own has also enabled me to get better at my transferrable skills significantly including communication, independence, ability to adapt, creativity and attention to detail.

As this was a project of rather short duration during which lots needed to be done, I had to plan everything carefully and manage my time effectively, improving my time management and scheduling skills considerably. I also tried to schedule and make use of my supervisor meetings as effectively as possible. For instance, I would not have them at regular times every week but rather every two weeks (as that was the length of one sprint) either mid-way or at the end of the sprint to discuss any problems I encountered.

Moreover, in order for me to complete this project on time, I had to pick up all the new things very quickly and be able to use them in practice. Being aware before the start of the project that I had no prior experience in mobile development, it also took a decent amount of courage and confidence in my skills and myself.

In hindsight, I believe the one transferable skill that I improved the most is problem solving. Over the course of this project, I went through and solved so many problems and challenges

that my problem solving abilities were tested and stretched out to maximum basically at all times.

12. Conclusion

12.1. Summary

This project gave me an opportunity to manage my first own big project whilst working on something that I am passionate about under the guidance of my supervisor. I have learned a great deal of skills that I will be able to use in the future.

Moreover, it has shown a great promise for future and if the launch of fully working solution is done properly and combined with effective marketing which teaches people about all the great benefits they get from exercising with a partner or in a group, it has great potential for success.

12.2. Project aims & objectives

Acquire knowledge about developing Android applications

Since I had no previous experience of using Android or Java, I started to learn it well before the project start. I undertook the complete Android & Java – Mobile app development | Beginning to End course by Philipp Muellauer and Angela Yu on Udemy and partly some of the free Google courses on Udacity, such as Developing Android Apps or Android Basics: Multiscreen Apps. The first course covered not only the development of an application but also the whole application lifecycle, from design documentation, development, UI and UX design to marketing and launching. I learned that mobile applications are not just about development and therefore amended my goals accordingly, too.

In addition, I spent hours of reading and researching many different Android and Java techniques and topics on other webpages as well. Furthermore, I also read the whole FireBase documentation.

Even though these helped me significantly, apparently, they could not prepare me for everything and I had to do lots of research and self-learning on the go, too. I finished the main Udemy training by the end of week 2. That is when I delved into actual coding.

Despite the fact that Android development is a huge topic and takes years to master, I can honestly say that considering the time constraints of this project I have gained lots of skills and knowledge on which I can build in the future.

Gain an understanding of the problem background

Since I have been leading healthy lifestyle in general for a couple of years now, I knew the basic principles that underpin and influence one's motivation even before the start of the project. However, it was not until I researched this topic even more thoroughly during the project when I realised that motivation can be influenced and boosted in so many more ways than I could ever think of. By reading different studies, I discovered the differences between habits of successful and habits of unsuccessful people, I gained a much deeper understanding

of why certain things work and others do not, and why people struggle to keep leading a healthy and active lifestyle. By working out with and talking to others regularly, I managed to get an insight into fitness users' needs and mind-sets.

Considering that I will be working on the application even after university, all this information will be very useful and will drive the application's functionality and features to help user lead a long-term active lifestyle.

Gain an understanding of the market & the need for my application

As mentioned in Section 2.4: Project justification, the current fitness market is huge and that includes the number of different applications available. Therefore, it is hard for one to justify the need for a new application to enter such a big market.

However, even after using and studying applications of some of the big players, I was still not convinced that they would help me to keep leading a healthy lifestyle in the long term as they were not focusing on intrinsic motivation too much but rather on extrinsic, i.e. the appearance. Hence, I believe I have found a valid reason and justification for the need of FitVenture. Not only it is solely built around and based on factors that directly increase one's intrinsic motivation, but it also assumes that not everybody has someone to work out or do sports with.

All in all, if the launch of FitVenture is combined with effective marketing campaigns as well as a big vivid community of fitness enthusiasts who will drive the downloads, it has a decent chance of success even in today's oversaturated fitness market.

Acquire an awareness of the possible approaches and tools & choose a suitable one

Regarding the software development methodology, my hybrid approach consisting of both waterfall and agile elements proved to be very useful and effective whilst working on the application. It allowed me to firstly give the project a stable structure and then deliver new working versions of the application in fixed regular intervals. Developing in smaller steps building on top of each other gave me confidence which was especially important as I did not have any prior Android or Java experience. Moreover, since I am used to planning things for long term first and then breaking that down into smaller goals or steps for the next week or so in real life too, I enjoyed doing sprints.

After all, it was not a matter of finding the "correct" approach, but rather a matter of adopting something that would work well for me. And overall, this hybrid approach worked proved to be highly effective and of great help.

However, it was not so straightforward when it came to writing the report, as I had to go back in time and explain what exactly I did in that sprint even though in some cases that was not part of the final solution. For instance, in one of the sprints, I partly developed a functionality and then I would completely rework it in the following sprint. In the report, I therefore wrote about the whole functionality in detail even though I knew this was not the correct or final solution.

Regarding the tools I chose to use, with one small exception (the update of Android Studio and subsequent downgrade), I cannot complain about a single one as they all fulfilled their purpose.

For the ones I have never used before, such as Android Studio, Taiga, Draw.io or FireBase, it was a great learning experience during which I gained enough skills and knowledge to keep using them even after the university and on different projects, too.

For the ones I have used before, such as Balsamiq or GitHub, I got significantly better at using them, too, as I have never used them for a project as complex as this.

Gain an understanding of user needs

I had certain functionality in my mind even before the start of the project, but by conducting a thorough research on what other similar applications offer and combining it with the findings from my informal talks with various fitness enthusiasts and with my newly gained knowledge about motivation, I was able to gain a good understanding of user needs. I then transformed these into user stories and requirements.

Outline a graphical representation of the app's functionality

With the help of Draw.io and Balsamiq, I was able to construct simple, yet powerful and clear representations of each application's functionality to help visualise the process before delving into development.

Build a community around the idea

Up to the date, the Facebook community has gained 168 and Instagram over 1000 followers over the last month and a half.

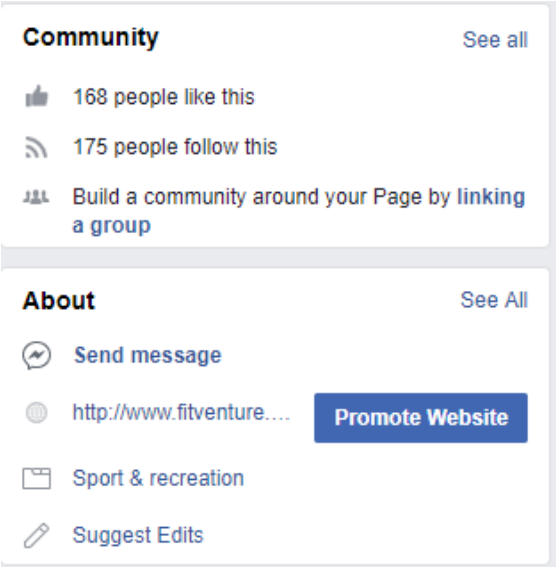


Image 16: Facebook page

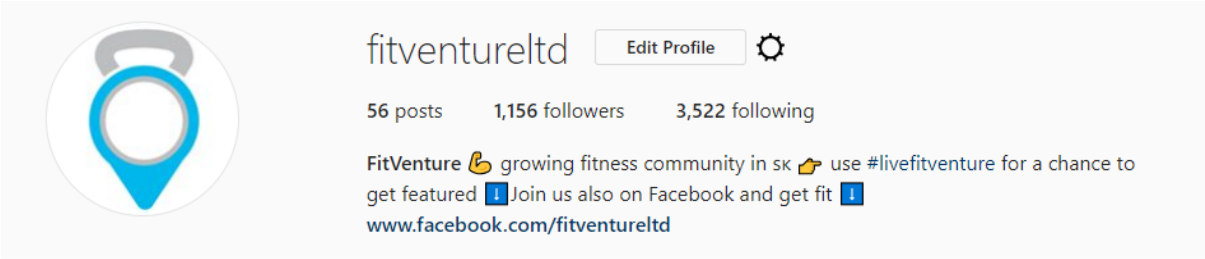


Image 17: Instagram page

I managed to organise two open training sessions which were attended by more than 10 fitness enthusiasts each.

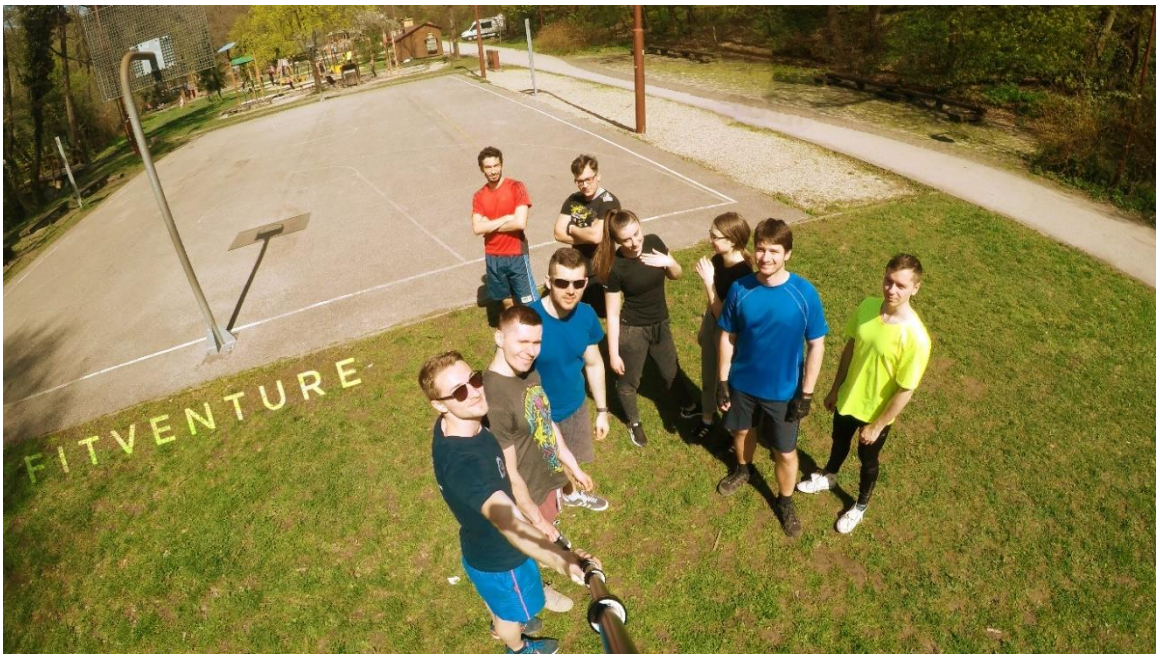


Image 19: First open training

Image 18: Second open training

I also managed to create a blog website and contribute with one article. Below is the list of all the articles so far and the number of times they have been read. Please note only the fourth article was promoted (for 3 days on Facebook). The columns from the left are article name, author name, category, tags and number of reads.

Názov	Autor	Kategórie	Značky	
Ako sa stať najrýchlejšiou verziou seba samého	admin	Fitness	latest	144
Chceš schudnúť rýchlo a efektívne? Stačí ti 20 minút denne	admin	Fitness	—	445
Fitnes – všetko alebo nič? (časť druhá)	admin	Fitness	—	94
Nemáš čas na cvičenie? Riešenie je jednoduchšie, ako si myslíš!	admin	Motivácia	—	1946
Fitnes – všetko alebo nič? (časť prvá)	admin	Fitness	—	220

Image 20: Blog articles

Since all of this was achieved without any excessive use of marketing and the numbers are steadily growing every day, there is an apparent interest in this kind of idea.

12.3. Future work

Since my idea validation has so far indicated that the project has potential, I am going to work on it even after university.

Application development

Regarding the mobile application development, there is huge room for additional functionality and as such provides me with coding material for the next few years.

Apart from the activity tracker and all the functionality marked as “nice to have”, the application could be enriched with the following features:

- Daily tracker
  The daily tracker would monitor user’s daily intake of food, macronutrients, water, vitamins as well as the length of their sleep and number of steps they walked. This is based on a study (Zhou et al., 2015) which suggests that intermittent exercise is as

effective as continuous exercise session, if not better. Since many people simply do not have time to dedicate one full hour to exercise, the application will count even discontinuous periods of exercise, such as walking to or from work. Together, these small details such as food and water intake or tracking even intermittent activity will all add up to the overall result. Of course, for this to work, the application needs to let user know that every little bit counts towards their health and goals and that it does not matter if it is a one-hour walk or 6 ten-minute walks spread throughout the day.

- Find a fitness partner at a slightly higher level
  The desire for competence at a certain activity has proven to be another critical factor affecting one's intrinsic motivation (Ryan and Deci, 2000). If one incorporates exercising with someone at a slightly higher level than himself or herself, they are much more likely to adhere to their plans. By enabling user to add their own level of competency at a certain activity and view others', this will ensure they will connect with the right partner. For this to work, again, the application will show hints explaining this to user.
- Newsfeed system
  The more people know about one's commitment to fitness, especially their friends and family, the easier it is for her or him to stick to their goals in the long term. Especially, if one has committed to regular workouts with their friend or fitness partner, it then becomes much more difficult and energy consuming to make up excuses or skip a workout than the exercising itself. The application could provide users with a Facebook-like newsfeed where friends could see one's workout and meal logs as well as enrolments in various fitness programmes.

- Ranking system
  Healthy competition is another driver of motivation and ranking system would enable users to compete for certain prices or application benefits. The more reputation points user gains for a certain period, the higher they will rank.

- In-person user adding system

  The application could provide users with ability to quickly add another user they meet (e.g. during an event or workout) as their friend by scanning their unique profile barcode to arrange future workouts together. Since people do not tend to give a stranger their Facebook straight away, this would be a great way of how to stay in touch without revealing too much private information.

Community

As far as all other activities are concerned, I am going to carry on posting high quality content with added value including inspiring pictures, exercise video guides and various articles about sticking to healthy and active lifestyle to further grow the community.

Once the application has been launched and the community grown bigger, local meetups could be organised to further strengthen the community. These would be special local events only available at certain time and place offering users both in-app and real world benefits only

obtainable throughout the event. This could be organised once a month in the capitals or in the application's trending cities (the cities with the largest amount of active members for the past month).

Moreover, various global challenge events could be organised, too. These would be accessible to everyone worldwide for a limited amount of time and participants would get special benefits obtainable for the completion of the challenge. For example, there would be a certain amount of reputation points available to collect when exercising at parks. This would serve both as a great motivation incentive to go out and exercise as well as opportunity to meet other like-minded individuals. Apart from other benefits, participants could also obtain a unique event badge which would only be obtainable during that specific event and would subsequently be displayed on their profile.

Blog

I will also be working on improving the blog webpage, mainly its usability and design.

Landing page & marketing

As mentioned in the beginning of the report (Chapter 3**:** Non-development activities), application landing page and marketing were out of scope of this project due to its complexity and time constraints. However, they are both essential and I will definitely look into them as soon as possible. I have already started doing so by undertaking a course on landing page creation.

13. Bibliography

Adam, H. and Galinsky, A. (2012). Enclothed cognition. *Journal of Experimental Social Psychology*, [online] 48(4), pp.918-925. Available at: https://www-sciencedirect-com.abc.cardiff.ac.uk/science/article/pii/S0022103112000200 [Accessed 24 May 2018].

Biddle, S. and Fox, K. (1989). Exercise and health psychology: Emerging relationships. *British Journal of Medical Psychology*, 62(3), pp.205-216.

Dearden, L. (2017). *This is the most overweight nation in Western Europe*. [online] The Independent. Available at: https://www.independent.co.uk/news/health/uk-obesity-rate-rising-overweight-worst-country-western-europe-world-us-ranking-oecd-research-a8049451.html [Accessed 24 May 2018].

Esplin, C. (2016). *Firebase Data Modeling*. [online] How To Firebase. Available at: https://howtofirebase.com/firebase-data-modeling-939585ade7f4 [Accessed 25 May 2018].

Firebase. (n.d.). *Structure Your Database*. [online] Available at: https://firebase.google.com/docs/database/android/structure-data [Accessed 25 May 2018].

General Data Protection Regulation (GDPR). (n.d.). *General Data Protection Regulation (GDPR) – Final text neatly arranged.* [online] Available at: https://gdpr-info.eu/ [Accessed 25 May 2018].

Holt-Lunstad, J., Smith, T., Baker, M., Harris, T. and Stephenson, D. (2015). Loneliness and Social Isolation as Risk Factors for Mortality. *Perspectives on Psychological Science*, [online] 10(2), pp.227-237. Available at: http://journals.sagepub.com.abc.cardiff.ac.uk/doi/full/10.1177/1745691614568352 [Accessed 24 May 2018].

IHRSA. (2017). *The 2017 IHRSA Global Report*. [online] Available at: https://www.ihrsa.org/publications/the-2017-ihrsa-global-report [Accessed 24 May 2018].

Kerr, N. and Hertel, G. (2011). The Köhler Group Motivation Gain: How to Motivate the 'Weak Links' in a Group. *Social and Personality Psychology Compass*, [online] 5(1), pp.43-55. Available at: https://onlinelibrary-wiley-com.abc.cardiff.ac.uk/doi/full/10.1111/j.1751-9004.2010.00333.x [Accessed 24 May 2018].

Leach, R. (2016). *Introduction to Software Engineering*. 2nd ed. CRC Press LLC, pp.14-16.

Lidwell, W. (2014). The Science of Logo Design. [course] Lynda. Available at: https://www.lynda.com/Logo-Design-tutorials/Science-Logo-Design/149123-2.html [Accessed 24 May 2018].

Muellauer, P. and Yu, A. (2018). Android O & Java - Mobile App Development | Beginning to End. [course] Udemy. Available at: https://www.udemy.com/android-app-development-with-java/ [Accessed 25 May 2018]

Obesity Update 2017. (2017). [ebook] OECD, pp.6-7. Available at: https://www.oecd.org/els/health-systems/Obesity-Update-2017.pdf [Accessed 24 May 2018].

Plante, T., Coscarelli, L. and Ford, M. (2001). Does Exercising with Another Enhance the Stress-Reducing Benefits of Exercise?. *International Journal of Stress Management*, [online] 8(3), pp.201-213. Available at: https://www.psychologytoday.com/files/attachments/34033/exercise-another.pdf [Accessed 24 May 2018].

Powell-Morse, A. (2016). *Rapid Application Development (RAD): What Is It And How Do You Use It?*. [online] Airbrake Blog. Available at: https://airbrake.io/blog/sdlc/rapid-application-development [Accessed 25 May 2018].

Ryan, R. and Deci, E. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, [online] 55(1), pp.68-78. Available at: http://ovidsp.uk.ovid.com.abc.cardiff.ac.uk/sp-3.30.0b/ovidweb.cgi?QS2=434f4e1a73d37e8cc073476c0173649e516284f6a7c11d2121c94b2f3092c96d91a6abf14e895fe876c3f09dadfaf4dd51e254349247c1a9d92a504fa0f8b25097b074da4ffd9f15ffdb2a018315d9d142d02231d64edb498d75a99dd16fe7569fa162314cf72670e3959369b4e09dbc4e550271588ad4338164b6167184a7376b0968662b726941ce25b614357bfbf7a73b93666f655bd4ef95dcf141d6c90cd9708f3fd414da0d1b67cca70e1a6cec96ba661e51cd8447e92fe02a1f0b01bba414bb0477a29b7c8819858011813c0f6e3c0c598f774ceae6228e670fe19ecd6bce5a83de35eca46125d2c9efb423b17c3d7f41395a9a23764dd0e35a90c91d5c62cde000dfdf19f65f5df02da736ee. [Accessed 24 May 2018].

Ryan, R., Frederick, C., Lepes, D., Noel, R. and Sheldon, K. (1997). Intrinsic Motivation and Exercise Adherence. *International Journal of Sport Psychology*, [online] 28, pp.335-354. Available at: https://selfdeterminationtheory.org/SDT/documents/1997_RyanFrederickLepesRubioSheldon.pdf [Accessed 24 May 2018].

Scrum.org. (n.d.). *What is Scrum?*. [online] Available at: https://www.scrum.org/resources/what-is-scrum [Accessed 25 May 2018].

Silva, M., Markland, D., Minderico, C., Vieira, P., Castro, M., Coutinho, S., Santos, T., Matos, M., Sardinha, L. and Teixeira, P. (2008). A randomized controlled trial to evaluate self-determination theory for exercise adherence and weight control: rationale and intervention description. *BMC Public Health*, [online] 8(1). Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2483280/ [Accessed 24 May 2018].

Statista. (2018). *Average number of new Android app releases per day from 3rd quarter 2016 to 1st quarter 2018*. [online] Available at: https://www.statista.com/statistics/276703/android-app-releases-worldwide/ [Accessed 24 May 2018].

Statista. (2018). *Number of smartphone users worldwide from 2014 to 2020 (in billions)*. [online] Available at: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ [Accessed 24 May 2018].

Thompson, W. (2017). WORLDWIDE SURVEY OF FITNESS TRENDS FOR 2018. *ACSM's Health & Fitness Journal*, [online] 21(6), pp.10-19. Available at: https://journals.lww.com/acsm-healthfitness/Fulltext/2017/11000/WORLDWIDE_SURVEY_OF_FITNESS_TRENDS_FOR_2018__The.6.aspx. [Accessed 24 May 2018].

Vliet, H. (2008). *Software engineering principles and practice.* 3rd ed. John Wiley & Sons, pp.66-68.

Wankel, L. (1993). The importance of enjoyment to adherence and psychological benefits from physical activity. *International Journal of Sport Psychology*, [online] 24(2), pp.151-169. Available at: http://psycnet.apa.org/record/1994-07751-001 [Accessed 24 May 2018].

Zhou, Z., He, Z., Yuan, M., Yin, Z., Dang, X., Zhu, J. and Zhu, W. (2015). Longer rest intervals do not attenuate the superior effects of accumulated exercise on arterial stiffness. *European Journal of Applied Physiology*, [online] 115(10), pp.2149-2157. Available at: https://link.springer.com/article/10.1007/s00421-015-3195-8 [Accessed 24 May 2018].