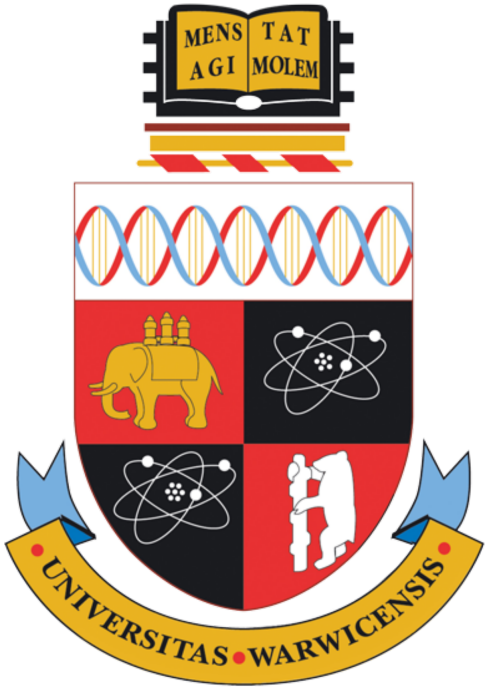


Skouted: *A Social Network Platform Connecting Football Players and Scouts*

Department of Computer Science

University of Warwick

Group: Billion Dollar Players

Supervisor: Dr. Claire Rocks

Year of Study: 2018-2019

Abstract

A significant number of young football players aim to develop their professional careers but often lack the appropriate connections or means to capture scouts' attention. The aim of Skouted is to bridge this gap between players and scouts by providing a digital platform for self-promotion. Scouts can then perform criteria-based searches to identify relevant talent and determine the quality by reviewing players' match videos and statistics. This paper describes the process of producing Skouted, exploring related platforms, ethical issues, and current software design trends. The paper then leads into the design and implementation of the platform, concluding by evaluating the software engineering process and final product.

Keywords: Social networking, Football, Self-promotion, Networking, Recommender systems, GDPR.

Acknowledgements

We would like to extend our thanks and gratitude to our project supervisor, Dr Claire Rocks, for the support and guidance she has dedicated to us. We would also like to thank our customer, Raj Sharma for providing us with a challenging yet exciting project idea which has allowed us to develop both socially and technically.

The team would also like to thank Adam Chester and Mike Joy for providing us with the guidance that allowed us to proceed with Skouted as our fourth year project.

Contents

Chapter 1

Introduction

The aim of this project is to develop a social networking platform called *Skouted*, which endeavours to connect young football players and scouts. Through *Skouted*, players will have the opportunity to upload videos from their official matches and training sessions, alongside statistics relating to their performance, to showcase their talent. Furthermore, players will have the chance to search for scouts and follow them to explore their criteria, whilst scouts will have the option to perform a criteria-based search of players and assess their gameplay footage. They will also be able to explore recommended players based on analytics, as well as contact players that they are interested in. Therefore, *Skouted* provides a digital means for scouts to efficiently search for youth players of all career levels, saving football clubs and agents a significant amount of time and financial resources.

The project team consists of 6 members, namely:

- Daniel Alarms
- Ignacio Borrego Melendez-Valdes
- Samuel George Ogden
- Akriti Pathania
- Harjot Singh
- Aliyah Stevens

This document sets out the background and motivation for the project, leading into the necessary

investigation and market research performed to coherently and suitably design the platform. An exploration of the legal, social and ethical issues is provided, followed by the system requirements specification and the design specification of the platform. The project's design and implementation is then discussed in detail, naturally leading into a discussion of the testing performed. Finally, the document concludes with an overall evaluation of the platform and a discussion of possible limitations and the scope for future work.

1.1 Background & Motivation

Traditional approaches to scouting young, aspiring football players typically consist in professional clubs employing full-time scouts to visit the highest number of possible matches, therefore maximising their chances of finding new talent [1]. This particular type of scout is known as a player scout, whose sole focus is to discover new talented players, as opposed to a tactical scout who focuses on analysing a club's opposition in certain competitions in order to obtain a competitive edge.

Albeit top-tier clubs having an extensive network of player scouts, often in the order of 500 - 700 [2], these numbers are insufficient for a club's network to visit every single relevant youth match. Scouts typically focus on the top-tier youth leagues and in particular the U15 - U18 range [1], constraining the players they analyse to those participating in these leagues. Though the concentration of talent is highest in these leagues, there is still a high volume of undiscovered talent in lesser-known official competitions. A clear example of this is world-famous ex-Chelsea striker, Diego Costa, who until age 16 was playing "street football" and had never been coached professionally, and only at age 18 was he reported by a scout [3]. Although there is a suite of scouting and coaching software tools available in the marketplace as described in section 2.1, these focus on providing access to player statistics for top-tier leagues. They facilitate the analysis of elite players above the U18 age range, but do not aid in any way in the process of finding young, undiscovered talent.

Therein lies the motivation for *Skouted*: a social network for young, aspiring football players to connect with scouts. Players can upload video clips of their match highlights, performance statistics, as well as providing the traits and characteristics of their playing style. Scouts can then view their data, follow them to keep track of their performance, and message them for any further professional opportunities. This 'self-promotion' model fundamentally differs from any other digital scouting platform, and does not limit the number of accessible players to the size of a club's scout network, as any football player can sign up. Hence, *Skouted* provides the digital means for aspiring football

players in a range of different youth leagues, from low to top-tier, to promote their own skills and connect with scouts. In turn, scouts would have the chance to efficiently search for players according to their preferred criteria, such as 'strikers with 10 or more goals in the last 12 games'. Additionally, the platform aims to apply analytical techniques to allow scouts to have an "explore" feed, through which they can explore recommended players based on their preferred search criteria.

1.2 Problem Statement

The fundamental problem the project attempts to solve is the lack of software platforms or digital means providing the chance for aspiring football players to promote their own skills, and in turn allow scouts to discover new talent.

1.3 Project Aims & Objectives

The aim of the project is to solve the aforementioned problem of a lack of digital venues by developing a software platform that connects players and scouts. The platform allows players to create their own profiles to describe their playing style as well as core characteristics such as their age and height, and upload videos of their matches and training sessions. In turn, scouts can then search for players according to their preferred criteria, follow players to track their performance and progress, and message them for further professional opportunities. As a whole, the project aims to democratise the chance for young football players to showcase their skills and progress in their professional careers through providing worldwide visibility to scouts.

1.4 Project Stakeholders

The identified stakeholders are listed in table 1.1. The table describes their individual interests in the project and highlights their influence over the project direction and their likelihood to cause change.

SID	Stakeholder	Category	Interest	Influence Rating	Likelihood to Cause Change
1	Customer: Raj Sharma	Internal	To be presented with a social networking platform which connects aspiring footballers to scouts in accordance with own business model.	High	High
2	Users: Players	External	Access to a medium which showcases their talent to scouts and allows them to communicate directly.	Low	Medium
3	Users: Scouts	External	Access to a medium which eases the process of discovering talent amongst young and aspiring footballers.	Low	Medium
4	Development Team	Internal	Members of the team are interested in succeeding academically and forming strong industrial relationships.	High	Medium
5	Supervisor: Dr Claire Rocks	Internal	Overseeing how the project develops and ensuring the direction does not stray too far from the overarching aims and objectives.	Medium	Medium
6	Department of Computer Science	External	Ensuring that the project lives up to the University's academic reputation	Low	Low

Table 1.1: Stakeholders

Chapter 2

Investigations & Research

Prior to beginning development of the application, research into existing systems, statistical models and ethical issues was conducted. This section outlines the research conducted into these topics and how the information discovered will be applied to the *Skouted* app during the project and after completion.

2.1 Related Work and Existing Systems

There are a number of mobile and desktop software products available in the market offering scouting solutions. These are, however, exclusively aimed at scouts and clubs, hosting data from elite leagues and players, and do not provide a self-promotion model for aspiring football players.

The most widely used scouting software is the Italian-born Wyscout [4]. Wyscout features a web and mobile platform providing a centralised database of player data, statistics and useful analytical tools for scouts. Their methods of collecting player data vary, but often come from clubs and agents paying for Wyscout's use and registering their own players and associated performance statistics. Hence, their data collection method is inherently conditioned on club and agent purchasing power, with no self-promotion model available. This highly limits players chances from lesser-known and less economically successful clubs, as well as underdeveloped socio-economic backgrounds, to reach scouts and succeed in their professional careers. Additionally, Wyscout's target audience continues to be first division clubs and leagues from the U18 range and above. Most scouting software operates in a similar fashion, other popular products being Scouting System Pro [5], or Scout7 [6].

Within this same category, a recently launched product named eye4talent [7] takes a slightly different approach. It allows players to become involved by uploading their own videos and then receiving feedback from their coach. However, eye4talent does not focus on a player's external visibility to other clubs, but in the process of internal promotion and feedback for players and coaches within a single club. *Skouted* fundamentally differs from this by allowing external self-promotion of young players. Again, eye4talent focuses on elite clubs and leagues from first divisions.

The products that most closely resemble *Skouted* in their conceptual approach are Sportifico [8] and Tonsser [9]. Sportifico aims to be a social network for athletes and coaches from a variety of sporting disciplines. The product is still in its early stages, and though similar to *Skouted*, its social connections have a different aim. Sportifico's focus is to allow player-to-player connections, and coach-to-coach connections, not allowing players' self-promotions in order to connect with prominent scouts.

Tonsser is also a social network aiming to give football players exposure and visibility to a variety of entities, such as agents, clubs and also scouts. In a similar way to *Skouted*, Tonsser offers the chance for players to create online profiles and upload match statistics in order to gain visibility through their platform. Therefore, whilst Tonsser does provide a similar self-promotion model its focus is not to provide scout-player connections, but rather to simply provide exposure to a more generic audience. *Skouted* however, focuses on and emphasizes player-scout connections for players' professional progression, featuring a dedicated scout search tool for players and a dedicated player search tool for scouts. Players can search for scouts and message each other, whilst scouts can do the same: both these features are not present on Tonsser. Therefore, *Skouted* continues to cover a specific and necessary market gap, that of facilitating player-scout connections to maximise player's chances of success.

Hence, though a sporting social network, *Skouted*'s approach is fundamentally different and covers a different market gap not addressed by any other popular existing products. Namely, it offers young, aspiring players access to a digital tool for self-promotion and connection with scouts, and in turn allows scouts to efficiently search for new talent.

2.2 Data Collection

Through use of the *Skouted* app, a lot of data becomes available about users from the information they supply, or relating to interactions of users with the app. This section outlines the data being collected and the collection and storage processes.

It is good practice and now a legal requirement[1] to provide explanations in how and why their personal data is being collected and used. Given *Skouted* has a young target audience, it is necessary to provide these explanations in a manner that is appropriate and meaningful to children. To do this, *Skouted* will describe the data collection procedures and reasoning in simple language, and where necessary, through diagrams and videos[2]. In addition to this, where data is being collected (for example through text boxes, check-boxes etc.) the platform should introduce 'information buttons' which, when clicked provide a description of why the data is being collected[3].

User Information

When a user signs up to the platform they will input information about themselves. This includes data such as name, age, gender, email addresses, the club a user plays for and their general location.

User Interactions

When a user interacts with the app, useful data can be extracted to help provide better content which is more suited to that specific user. Data extracted might include include liking and commenting of posts by other users, spending a large portion of their in-app time watching specific types of videos or consistently searching for users of specific demographic.

Collection Methods

Data will be collected automatically through forms for the case of user information or through triggering events which log information on the server when an action is executed by the user, for example if a user marks a message as 'spam' this will be recorded against the user that sent the

[1] See Section 3.1.2

[2] This is a stretch goal and must be implemented prior to release of the application in order to comply with the current GDPR legislation

[3] This is another stretch goal and must be implemented prior to release of the application

message for future use. This will allow large amounts of data to be collected quickly and be put to use to help provide a better user experience.

Storage

Data will be stored in a way the puts the user interest at heart. The aim is to use the data generated in order to improve user experience without violating any user's privacy. The data will be stored in the cloud, specifically Google FireStore. Issues relating to data compliance are discussed in Section 3.

2.3 Statistical Models

Within the *Skouted* app there are a number of areas which open the app up to potential abuse. Areas which, with a large user base, would be impossible to manually moderate. Areas such as messaging (introduce the possibility for spam) or the upload feature (introduce the possibility for graphic content). In this section different statistical models are outlined showing how they can be used to reduce the amount of manual work required through classification of content as 'good' or 'bad'.

Spam Detection

Irrelevant and unsolicited messages sent through the Skouted platform are not acceptable. With a young, impressionable audience it would be relatively simple for businesses to take advantage of users on the platform to encourage them to buy products/services. Not only this but spam introduces a large security issue with many links in spam messages leading to phishing schemes or websites with malware [10]. To enable the Skouted team to efficiently deal with such messages, spam detection could be put in place to protect users and keep the platform cleaner [4]. In order to do this, machine learning can be applied in the form of a classifier to classify messages as 'spam' or 'not spam'.

Current methods for spam detection include blacklisting users/IP addresses if they are suspected of spamming users and detecting bulk messages [11].

Detecting bulk messages would provide a good indication of spam on the Skouted platform since

[4]Spam detection will not be implemented in this university project

there is little reason for a user to send the same message multiple times (unlike in email where that would be quite common for company wide "email blasts"). Bulk detection could easily be implemented by storing a hash of every message (which reduces memory requirements since *hash(m)* is usually less than m) in size. By storing the hash of every message in a hash table, lookup for an existing email can be executed in constant time providing a very quick check. Messages that are older than a certain time period could then be deleted from the table to conserve space.

The method mentioned above would only be useful if the spam messages were exactly the same. An intelligent malicious user could easily avoid this method by slightly changing each message resulting in a different hash each time.

A better approach may be to combine bulk message detection along with a classifier. There are many possible models that could be used to classifier messages as spam such as decision trees, K-nearest neighbour, Bayesian methods and many more [11]. Here the K-nearest neighbour method is presented as an effective way to detect spam.

K-Nearest Neighbours

The K-Nearest neighbours is a simple classification algorithm which classifies an instance by majority vote of its K neighbours in the feature space. K is typically small, for example, if k=1 then a new instance is simply classified to the same class as its nearest neighbour. An instance has multiple attributes (such as sender, receiver, time, message) and the distance between two different instances is calculated using this information.

For a spam classifier, the message content is going to play a key role in the prediction process. However, a message may be long and contain many words that don't provide information to whether a message is spam (words such as: a, is, and, the etc). In order to reduce the dimensionality of the data such words can be removed which provides a more accurate classifier.

Figure 2.1 shows a KNN classifier example. For k=1 the green dot would be classified to the same class as the triangles. But for k=5 the green dot would be classified to the same class as the squares, assuming Euclidean distance is being used.

A significant amount of research has been carried out in this area establishing that KNN's often perform well in spam detection, consistently achieving an accuracy of over 99% [12].

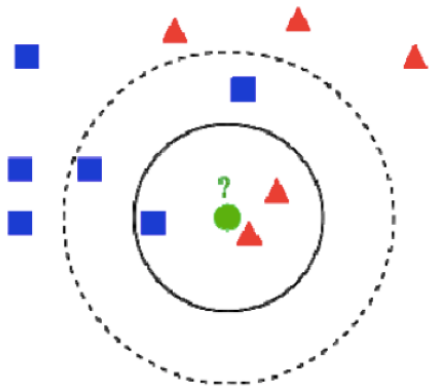


Figure 2.1: K-Nearest-Neighbour example where each colour represents a different class.

Inappropriate Video Detection

In order to keep the content on the platform relevant it will be necessary to moderate what is allowed by users. This would come in the form of content guidelines. However, some users may still upload videos that are inappropriate or graphic in nature. To automate the handling of this issue, machine learning algorithms can be put in place that process each upload and determine if it is relevant.

The problem of graphic content on social media is not new, any many solutions have previously been researched. One successful method is outlined here, along with how the results can be interpreted and used.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep neural network often applied to analysis of visual content. CNNs work by identifying important features in the visual data with respect to what the model is trying to achieve, having been demonstrated as effective models for understanding image content. For the application here, the CNN will also have access to a complex temporal element relating frames in the video. Andrej Karpathy et al [13] investigated the application of CNNs to sporting video data.

The Sport-1M public dataset provides 1 million videos of sporting events for 487 classes of sports. This data can be used to train a CNN classifier to spot football videos, outputting a probability that a video is not football related.

Using The CNN result

Given a CNN and a new video upload, the CNN can output a probability of the video being sport related. If the video gets a very small probability, then the upload can be blocked. If the probability is not informative (i.e. 50/50 chance it is or isn't relevant), then it can be passed on to a human moderator who will determine if the content is deemed appropriate for the platform. This approach greatly reduces the work needed to be completed by the human moderator by automatically rejecting the most irrelevant content and ordering new uploads for moderation by their risk level.

2.4 Recommender Systems

In the *Skouted* platform there are multiple points where players and scouts will consume information provided. Initially, the information provided is independent of the user, there is no personalisation. The aim of recommender systems is to provide a personalised view of the data on the system which maximises some utility function. Adomavicius and Tuzhilin [14] formalized the idea of recommender systems as follows:

> "Let C be the set of all users and let S be the set of all possible items that can be recommended. Let u be a utility function that measures the usefulness of item s to user c, i.e., u: CxS $\rightarrow$ R, where R is a totally ordered set (e.g., non-negative integers or real numbers within a certain range). Then, for each user c$\in$C, we want to choose such item s$\in$S that maximizes the user's utility."

This definition of recommender systems tells us that the aim is to optimize the experience of each user individually, not represent group consensus.

Content Based Methods

The utility function for content based methods uses previous ratings information supplied by the user for other items. The utility $u(c, s)$ of item s for user c is estimated based on $u(c, s_i)$ assigned by user c to items $s_i \in S$ that are "similar" to the new item s. For example, a video upload recommendation system would would try to understand similarities among the videos that a user c has rated highly in the past e.g. (specific publishers, genre, subject matter, etc.). Given this information the system would only recommend videos with a high degree of similarity to those

which the user rated highly.

An *item profile* is a set of attributes which characterises an item s, computed by extracting a set of keywords from its content. The importance of each word k_i in item d_j is determined by some *weighting* measure $w_{i,j}$. The most common measure for keyword importance in an item is the *term frequence / inverse document frequency* (TF-IDF) measure.

The *term frequency (TF)* for a keyword k_i in document d_j is defined as

$$TF_{i,j} = \frac{f_{i,j}}{max_z f_{z,j}} \tag{2.1}$$

where $f_{i,j}$ is the number of times keyword k_i appears in item d_j. This on its own does not provide much information about the importance of a keyword since some words generally appear more often in a document. The *term frequency* of a keyword in a given item is offset by the *inverse document frequency*, i.e. the number of items in the database that contain that same keyword

$$IDF_i = log\frac{N}{n_i} \tag{2.2}$$

where N is the total number of items that can be recommended and n_i is the number of items that keyword k_i appears.

Finally the weight (importance) of a keyword k_i in document d_j is defined as

$$w_{i,j} = TF_{i,j} \times IDF_i \tag{2.3}$$

and the *item profile* of document d_j is a vector of weights

$$itemProfile(d_j) = (w_{1,j}, w_{2,j}, ..., 2_{k,j}) \tag{2.4}$$

The preferences of a user c will be represented in a similar way. The $userPreference(c)$ of user c will also be a vector of weights $(w_{1,j}, w_{2,j}, ..., 2_{k,j})$ where each weight represents the important of keyword k_i to user c. This is obtained by analysing the ratings supplied by the user and then averaging for each keyword across all rated items.

Given a users preferences and the *item profile* weight vector for a new item s, the utility of recommending item s to user c can be calculated using a similarity score. A common approach for calculating similarity between two documents is the *cosine similarity* measure. The utility $u(c, s)$ of recommending item s to user c is now defined as

$$\begin{aligned} u(c,s) &= score(userPreference(c), itemProfile(s)) \\ &= cos(\vec{\mathbf{w}}_c, \vec{\mathbf{w}}_s) = \frac{\vec{\mathbf{w}}_c \bullet \vec{\mathbf{w}}_s}{\| \vec{\mathbf{w}}_c \| \times \| \vec{\mathbf{w}}_s \|} \end{aligned} \tag{2.5}$$

With equation 2.5, a utility score can be calculated for every item s in the system, and then present the highest scoring items to the user [14].

2.4.1 New User Problem

Content based methods require a sufficient amount of ratings from a user before being able to provide recommendations, as the system needs to understand their preferences. This means the system has to wait a period of time until they have rated enough items for the system to make good quality recommendations.

2.4.2 Collaborative Filtering

Collaborative recommender systems predict the utility of an item for a user based on previous ratings by other users. Formally, the utility $u(c, s)$ of item s for user c is estimated based on $u(c_j, s)$ assigned to items s by users similar to c, $c_j \in C$.

This is different to content based recommender systems since the recommendations are obtained through similarity of users rather than similarity of the content rated by a user.

An estimate for a rating of an item s by user c, $r_{c,s}$ can be obtained by averaging the rating of the N most similar users to c who have previously rated item s, this is known as memory based collaborative filtering. The aggregate function can be weighted by the similarity of the two users:

$$r_{c,s} = k \sum_{c' \in C} similarity(c, c') \times r_{c',s} \tag{2.6}$$

Where C is the set of all users who have previously rated item s, and k is a normalising factor. $r_{c,s}$ is then the estimated rating of item s for user c and the highest rated estimates should be recommended first.

The similarity of users can be calculated using the same method as the content based approach, using the cosine similarity method as shown in equation 2.5 where each user is represented as a vector. This vector could represent information such as previous ratings, demographic information and other interactions.

New Item Problem

Collaborative methods rely entirely on users preferences to make recommendations. Therefore, until a new item has enough ratings, the recommender system would not be able to recommend it.

2.4.3 Hybrid Approach

A hybrid approach combines both collaborative and content based methods. This combination helps to reduce the limitations of them individually.

One approach to combining these two approaches is to leave them separate and simply take an average of the ratings supplied of a given item for a specific user.

Another approach would be to use the collaborative method as the main system and add in content based characteristics to it. For example, the user preferences of a specific user c described in the content based method could be added to the user vector in the collaborative method. Thus, when the similarity of two users is calculated, it also takes into account the content based user preferences [14].

2.4.4 Applying Recommender Systems To The Explore Feature

The explore feature in the *Skouted* app provides a way for users to discover new content by displaying videos from users that are not currently followed. To make this feature effective it would be beneficial to recommend content that the user is likely to enjoy. This can be done through a hybrid recommender system as discussed in Section[5] 2.4.3.

Application of a recommender system within the *Skouted* app would be slightly different to examples given above as *Skouted* does not deal with explicit ratings. Instead, implicit information about how much a user liked a specific post can be used. Transactional data such as time spent watching a video, liking, commenting and sharing of a video can be compiled together to be used as an estimate rating for an item by a specific user.

These estimate ratings can then be used in a hybrid approach to calculate similarity of users and recommend videos v to a user c whose most similar neighbours enjoyed v.

[5]Recommender systems will not be implemented during the university project

2.4.5 Applying Recommender Systems For Player Suggestion

Scouts using the platform expect to see players that are of good quality and having skills relating to positions they are looking to fill. Scouts will have the ability to rate players, which can be supplied to the recommender system which in turn will provide more suggestions for the scout to look at.

Chapter 3

Legal, Ethical and Social Considerations

This chapter outlines the legal, ethical and social issues that must be considered prior to the release of the application. The chapter begins by outlining the legal responsibilities that the application must comply with before discussing the social and ethical impacts the application will have on its userbase, and how *Skouted* will enable users to exercise their rights to privacy, expression and access to information in the future[1].

3.1 Legal Considerations

3.1.1 Data Protection Act 2018 (DPA)

The data protection act aims to allow individuals to take control of their personal data by outlining the lawful processing strategies organisations should follow in regards to personal information. The 2018 act came into force in May of the same year, and updates the 1998 version. The data protection act is a supplement to the GDPR as it extends data protection laws in areas the GDPR fails to cover or makes adaptions to the rules defined by the GDPR to reflect national requirements. The data protection act does not write the General Data Protection Regulation into UK law. This means when the UK leaves the EU, the GDPR will be converted (with amendments) into UK law.

[1]The application will not implement all of the requirements outlined in Section 3.2 before project closure in May 2019. Instead these requirements will be enforced prior to the release of the application to the public.

Elizabeth Denham - Information Commissioner stated in Beyond 2018 - data protection laws built to last [15]:

> "The previous Data Protection Act, passed a generation ago, failed to account for today's internet and digital technologies, social media and big data. The new Act updates data protection laws in the UK... [and]... provides tools and strengthens rights to allow people to take back control of their personal data."

Children and the DPA

The Data Protection Act 2018 does not explicitly mention children but the provisions laid out within it are meant to apply to them. The lack of distinction between the right to privacy based on age undermines the rights of children. It is common for parents and guardians of a child to act on their behalf because the child is not considered to hold the mental capacity that would allow them to understand or exercise their rights to privacy.

Unlike the GDPR, the DPA 2018 does not require for children's data to be processed in a clear and transparent way which a child is able to understand. This is not the case under the GDPR.

Under article 8 of the GDPR, consent will only be valid if the child is at least 16 years old. Under the Data Protection Act, the UK has applied for a lower age of digital consent. In the UK this limit is now 13 years old.

3.1.2 GDPR

Skouted processes personal data relating to both European scouts and players and thus must comply with the GDPR. There are a number of obligations that are enforced by in the GDPR that affect the way data must be collected and processed.

Generally speaking, all data needs to be processed lawfully, this means the data subject (scout, player or the players parent if necessary) must have given consent for their personal data to be processed. It also means their data should only be collected for specified, explicit and legitimate purposes. *Skouted* will not be able to process data outside of its original scope. If changes to processing do occur, clear descriptions of the changes will be delivered to the users of the platform and their consent to the new processing procedures will be obtained.

Prior to the *Skouted* application being released to the public, all the required rules and procedures

laid out by the GDPR will be followed to comply with current laws and regulations. The following subsection will outline how the GDPR effects how the platform interacts with children.

Children and the GDPR

Children [2] need additional protection where their personal data is being collected and processed as they may not completely understand the procedures, consequences and risks involved. The *Skouted* platform processes children's personal information, so it is necessary to be aware of and comply to current legislation. In doing this, *Skouted* will be able to incorporate the necessary safe guarding procedures into the design of the system[3]. *Skouted* must ensure transparency of processing by informing everyone what the application is doing with their personal data, describing the risks and safeguards involved and by proving them with easy access services which enable them to report anything that they believe is a breach of their rights.

Prior to the General Data Protection Regulation, there was no law defining the minimum age at which a user can sign up for a social networking service in the UK. Instead, the minimum age of 13 was inherited through the COPPA (Child Online Privacy and Protection Act) [16] legislation in the US. Since the GDPR and DPA 2018 came into force however, the 'age of digital consent' has officially been set to 13 years old in the UK. The general rule laid out in article 8 of the GDPR is for parental consent to be gained by youths under the age of 16 years old [17], the rule is laid out as follows:

> 1) Where point (a) of Article 6(1) applies in relation to the offer of information society services directly to a child the processing of the personal data of a child shall be lawful where the child is at least 16 years old. Where the child is below the age of 16 years, such processing shall be lawful only if and to the extent that consent is given or authorised by the holder of parental responsibility over the child.
>
> Member states may provide by law for a lower age for these purposes provided that such lower age is not below 13 years.
>
> The controller shall make reasonable efforts to verify in such cases that consent is given or authorised by the holder of parental responsibility over the child, taking into

[2] Children are those under the age of 18 as defined by the UN convention

[3] As already stated, these safe guarding procedures will be implemented prior to the release of the application. This is not achievable within this project which closes in May 2019.

consideration available technology.

Under the Data Protection Act, the UK has applied for a lower age of digital consent. In the UK this limit is now 13 years old. The differing age restriction across each of the member states can be seen in Figure 3.1[4]. Where values are enclosed within parenthesis, the state is considering changing the age of consent to that age. These differing age restrictions must be respected, should *Skouted* be rolled out across countries with differing age restrictions. In practice, this means when a young person registers with the platform from any state within the EU, they should select their country of residence so the correct age restriction rules are applied.

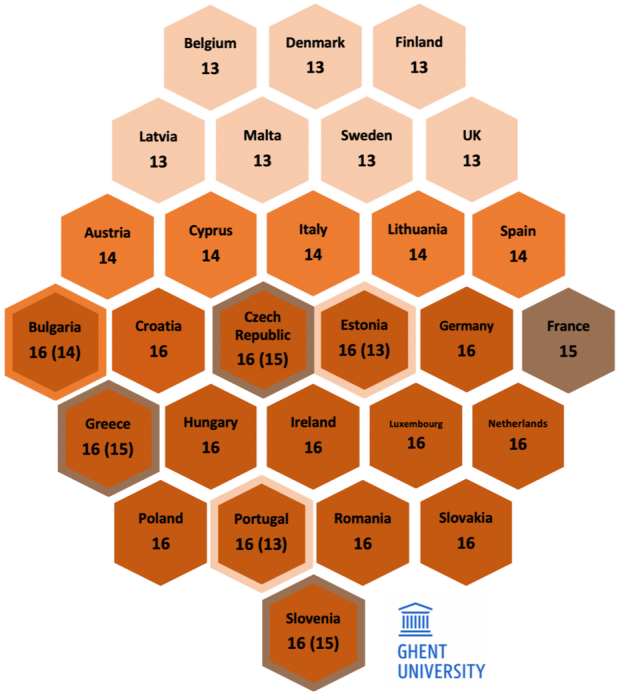


Figure 3.1: The differing age restriction across each of the EU member states. Where values are enclosed in parentheses, the country is considering changing the current age restriction.

For children under this age, consent needs to be provided by the person or people that hold parental responsibility over the child. Thus the *Skouted* application must rely on legitimate consent being granted before lawful processing of a child's personal information can take place. *Skouted* must develop a system which can verify a child's age and their right to consent and\or verify the age of the person providing consent on the child's behalf.

[4]Image Available from https://www.betterinternetforkids.eu/web/portal/practice/awareness/detail?articleId=3017751

Age is not always an adequate indication of maturity and understanding. It would be more appropriate for the application to determine whether the individual child has the level of competence expected of a person that would be able to provide consent themselves. To do this, *Skouted* could include some form of 'competence test' after an initial age verification to safeguard vulnerable people by testing their understanding of the implications of the collection and processing of their personal data. In the case that a child is not considered competent by the standards of the test, parental consent must be gained.

If consent is gained from a holder of parental responsibility, the child will be informed how they can withdraw that consent when they are able to make a competent decision.

3.2 Ethical and Social Considerations

Interactive social media technology has revolutionised the way people connect and interact with others. Social platforms like Facebook, Twitter and Instagram have skyrocketed in popularity due to their ability to connect children, adults and vulnerable persons. Such platforms offer many positive opportunities for people of all ages to communicate, interact and share their interests, memories (such as images and videos) and other content. Although social media brings an array of advances and benefits, it also introduces a range of potential safeguarding risks to children, young adults and vulnerable persons.

Given the *Skouted* platform targets young and vulnerable persons under the age of 18, there is an implied duty of care. It is necessary to ensure and enhance the safety of its users by providing a secure online environment for them to use. The *Skouted* platform collects and processes personal information, therefore care must be taken to ensure that such data is processed and handled lawfully and within the bounds of current data regulations such as the GDPR as described in Section 3.1.2.

The remainder of this chapter will discuss why social networking platforms are popular amongst young people, before outlining the framework the *Skouted* platform will follow in order to empower children to exercise their rights to freedom of expression, privacy and access to information. It will then proceed to highlight the potential risks and safety concerns applicable to the *Skouted* platform before suggesting safeguarding techniques that should be enforced before the application is released.

3.2.1 Why are social networks popular with young people?

We are living in a digital age. The ubiquity of electronic communications and various other computing technologies has provided an extensive array of technical advances that improve our quality of life and affects the way we socialise and connect with others. The use of social networking services has become a compelling activity for many, especially young people.

Social networking services like *Skouted* are popular among young people for many reasons such as keeping in touch with friends and making new ones, but most importantly, social networking services provide young people with a space to express themselves without their parent or guardian's full knowledge. Children and young people are often reluctant to disclose their online personas and activities to their parents or guardians as they feel they are capable of manoeuvring social networking platforms themselves. Given the age and maturity level of *Skouted's* audience, there is a clear duty of care that needs to be provided by the application to protect these young people.

3.2.2 *Skouted* Framework

This section outlines the framework the *Skouted* platform will follow to ensure its young and vulnerable users are safeguarded and able to exercise their rights of freedom of expression and privacy.

Minimum Age

Skouted will primarily be available to users in the UK. Like many other service providers [18], the platform has set the minimum age at which a user can sign up to the platform to 13 years old. Prior to the General Data Protection Regulation, there were no laws defining the minimum age at which a user can sign up for a social networking service in the UK based on their own consent. Instead, the minimum age of 13 was inherited through the COPPA (Child Online Privacy and Protection Act) [16] legislation in the US. Since the GDPR came into force however, the 'age of digital consent' has officially been set to 13 years old in the UK.

The legal age of digital consent varies across each of the member states as described in Section 3.1.2 and shown in Figure 3.1. These differing age restrictions must be respected if *Skouted* is to be rolled out across countries with differing age restrictions. In practice, this means when a young person registers with the platform from any state within the EU, they should select their country of residence so the correct age restriction rules are applied.

Content Moderation

As will be mentioned later in this chapter, content posted on the application will be moderated to protect younger users from inappropriate or harmful content. The effort must be made to ensure the level of moderation does not overstep the line between safeguarding children and violating a user's right to privacy and access to information. Where a post is being reviewed, an internal team of moderators will take the post and determine whether the post is inappropriate or offensive.

The following two approaches should be included as part of the *Skouted* social media governance policy to monitor and moderate user generated content (UGC):

- Pre-moderation
- Reactive moderation

Pre-moderation Pre-moderation is the process of moderating user generated content before it appears on the *Skouted* application. This technique guarantees inappropriate content never reaches the platform and cannot be viewed by any of its users. Although this ensures vulnerable users are safeguarded, users can become frustrated or confused when their content is not viewable instantly and they may make multiple reposts.

Pre-moderation should be used[5] in the cases where a user has received a warning, or has been recently reported. By pre-moderating a user's posts, they will be unable to cause any offence without being completely removed from the application without trial. If they are found to repeatedly generate inappropriate content, they will then be removed from the application.

Reactive moderation Reactive moderation is dependent on users reporting inappropriate content if they see it. The platform will provide a 'report content' button next to a post, when this is pressed, the user should be redirected to a report abuse or content form. The form presented will be understandable and meaningful to the user based on their age. The downside to this method is that there could be a lot of false positive reports, meaning people could be reporting content that is not inappropriate. In addition to this, an inappropriate post can only be removed once reported so at least one person could see this harmful content, but to ensure users exercise their rights to freedom of expression, the level of moderation should not go higher than this.

[5]Moderation will not be enforced during this project but will be prior to the application being released.

Child's right to Privacy

There is responsibility to realise and respect a child's right to privacy across the *Skouted* platform. It is only recently that the attempt has been made to recognise a child's right to privacy. Prior to the adoption of the GDPR, the Data Protection Act of 1998 made no distinction on the right to privacy based on age; this undermined a child's right to privacy. It was only through implication that children were given the same rights as adults. It is common for parents and guardians of a child to act on their behalf because the child is not considered to hold the mental capacity which allows them to understand or exercise their rights to privacy. This then raises the debate as to whether a parent's right to manage their child's privacy is their duty, or an invasion of privacy [19].

Providing a single definition of privacy has proven to be difficult in the digital age. Westin [20] provides a definition for privacy that applies strongly to the digital we live in today:

> "The claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others."

While Westin's definition of privacy encompasses the interconnections between privacy, surveillance and liberty, it fails to differentiate between the dimensions of privacy. UNICEF 2018 [21] outline four different dimensions of a child's privacy:

Physical Privacy is affected by situations and technologies which track, monitor and/or live broadcast a child's live image, behaviours, activities or locations.

Communications Privacy can be violated when unintended recipients access the child's posts, images or message logs.

Informational Privacy can be breached when the child's personal data and information is collected, stored and processed without their consent or total understanding.

Decisional Privacy can be breached when a child is restricted access to information which may benefit them. This in turn forbids the child from making informed decisions.

It is important for the *Skouted* platform to provide young and vulnerable persons with information and tools that help them understand the concept of privacy. Collecting personal information from children and vulnerable people should and will be limited to what is necessary for the use and delivery the *Skouted* platform. In accordance with the guidelines outlined by UNICEF, the *Skouted*

platform has and will continue to recognise a child's right to privacy by incorporating it into the design of the application by exercising the following procedures:

- The personal information collected from the user will be limited to what is necessary for use and delivery of the *Skouted* platform. Information which is not relevant or required by the platform will not be collected[6].
- Explanations will be provided to the users of the platform describing how and why their information is being collected in a manner that is appropriate and meaningful to them.
- Fresh consent will be sought from users where a change in how their information is being used or what information in required from them, has occurred. Consent will be requested through application notifications or via email link, which will redirect the user to their settings profile.
- User data will be stored securely. Should a security breach occur, meaningful information about the breach and how users should respond will be provided.
- The *Skouted* platform will provide a means for children and parents to change or delete personal data stored by the platform.
- The *Skouted* platform will provide a means for children and parents to delete personal images or video content which was previously supplied by the child or contains content which relates to the child.
- Upon closure of a *Skouted* account, all personal data collected from the user will be fully deleted.

Children's right to Freedom of Expression and Access to Information

Like adults, children have a right to freely express themselves. *Skouted* allows its users to express themselves through text and video uploads, while also allowing them to view information about other football players who are using the platform. *Skouted* should ensure that its young and/or vulnerable users can exercise their right to freedom of expression and access to information under protective measures. These protective measures should be developed in a manner which do not undermine a child's capability to navigate the platform safely by enforcing overly restrictive measures. Following the guidance provided by UNICEF [21], *Skouted* will seek to exercise the

[6]Further details on the information collected can be found in Section 2.2

following procedures to ensure children and vulnerable persons can exercise their right to freedom of expression and access to information:

- Content will be moderated[7] to protect users from negative events, such as exposure to worrying or nasty comments or behaviours from other users. Moderation should be proportionate to the level of competence *Skouted's* users are expected to hold.
- Although *Skouted* aims to be a professional platform which allows children to showcase their talents to scouting professionals, pseudonyms will be warranted. Permitting the use of pseudonyms provides children with the option to remain anonymous online, is particularly important on this platform as children could be vulnerable to predators and child exploitation. Children should have the option to remain anonymous until they believe it is acceptable not to.
- *Skouted* will incorporate parental control mechanisms across its platform. These mechanisms will operate on a transparent and easily adjustable 'opt-in, opt-out' basis. The 'opt-in, opt-out' basis ensures that parents/guardians have been explicitly informed about any decisions to be made and have made a conscious decision to filter or moderate the content their child can view.
- Parental control mechanisms should be easily adjustable to adapt with the child's evolving capabilities.
- Parental control mechanisms should be transparent to both the child and the parent by providing details as to what content is being blocked and what information is being relayed to the parent/guardian from the private messaging facilities provided by the platform.
- *Skouted* will present its terms and conditions[8] in an understandable and meaningful way. T&Cs will be adapted to the level of competence the user is expected to have and where appropriate will be communicated through diagrams, images and videos. Guidance will be taken form the Children's Commissioner, Tes and Schillings [22].

[7] See Section 3.2.2, Content Moderation

[8] See Appendix B

Children's right Protection of Reputation

A child's online presence can shape their real-world reputation and can have an impact on their employability or access to education in the future. Children rarely realise the data they provide to websites and online platforms can 'follow' them from childhood to adulthood. It is therefore necessary to empower children and vulnerable users of the system to protect their online reputation by providing them with the following facilities and procedures:

- User's will have the ability to check, contest, rectify, erase and edit the personal information stored about them, as per the recommendation of Livingstone [23]. *Skouted* does not wish for any of the data collected to be used beyond its original intention and thus providing users with the facilities to observe and modify the data collected about themselves is necessary.

Protection of a Child's rights in accordance with Evolving Capacities

A child's understanding of how the world works rapidly evolves. Considering this, it is necessary to provide *Skouted* users with protective mechanisms that mature with them.

- Assistance in understanding how and why their personal information is collected will be at a level appropriate to their registered age and maturity.
- Parental control mechanisms should be easily adjustable to adapt with the child's evolving capabilities.
- Parental controls mechanisms will be automatically adjusted at suitable ages to allow the child to make informed decisions on their own accord, for example, on a user's 16th birthday, private messages between a player and a scout will no longer be relayed to the user's parents or guardians. Parent's will be informed of this change and will be able to revert back to messages being relayed to them if the child agrees.
- Users will not be required to provide informed consent where they do not possess the capacity to do so. Instead parental consent will be required.

Child's right to Access a Remedy for Violations of Rights

All children should have the right to seek an effective remedy if their rights have been breached. All users of the *Skouted* platform should be aware of their rights, how they should be respected and how they can exercise them. The *Skouted* platform will provide its users with the following

mechanisms and procedures to ensure that they can seek remedies where their rights to privacy, expression and protection of reputation have been violated:

- *Skouted* users can easily access effective user and content reporting facilities.
- Reporting mechanisms will be provided which adapt to the competence level of the user based on their ages and maturity.

3.2.3 Risks to Children using *Skouted*

There would be no risk in using the *Skouted* platform if all registered users used the application as expected. This is an unrealistic belief to hold. Although most people use social networking platforms positively and as expected, there are a minority of people that do not. This group of people put users, especially young and vulnerable persons, at risk. The following list provides an overview of the potential risks young and vulnerable people could exposed to through the platform (this list is in no way exhaustive):

- Bullying by other users of the platform
- Exposure to inappropriate/harmful content, e.g. images, videos, comments
- Having personal information which could reveal the child's location offline
- Theft of personal information
- Sexual grooming, exploitation and abuse through contact with dangerous users
- Exposure to information that encourages self-harm
- Exposure to racist, sexist or hateful material
- Developing a negative body image through content seen on the platform
- Encouragement of violent behaviour
- Glorifying activities such as drug taking or excessive drinking
- Encouragement to post videos which place the young people or others in harm's way, e.g. performing risky stunts

The following sections provide a discussion of some of these risks, and present the procedures and

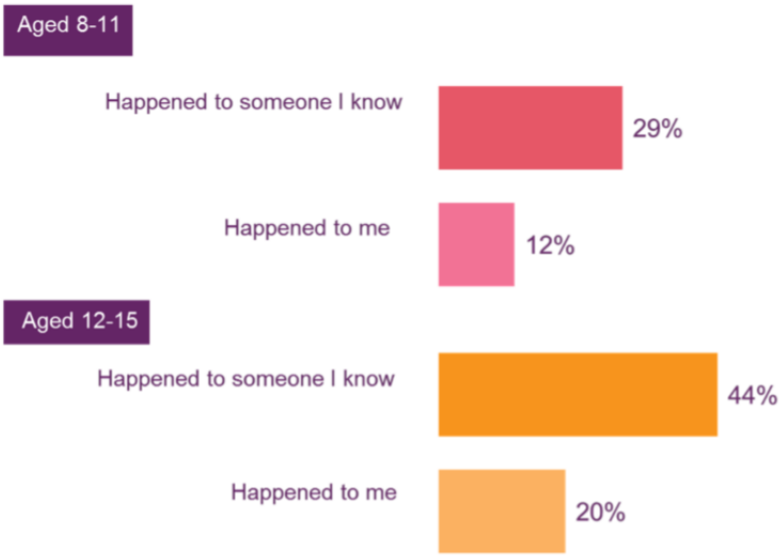
mechanisms the *Skouted* platform will exercise[9] in order to safeguard its young and vulnerable users.

Cyberbullying

The risk of cyberbullying is a growing concern among parents and teachers [24]. Suicide amongst young people is increasingly being caused by cyberbullying attacks [25]. It is important for a platform like *Skouted* to be aware of the power cyberbullying has over its users in order to combat it. Cyberbullying has the power to bring distress and harm to a person, and it can exacerbate problems such as negative self-image and suicide.

Figure 3.2 shows the number of 8-11 year olds and 12-15 year olds that have experienced or know someone who has experienced bullying[10]. In both age groups, children are twice as likely to know someone who has been bullied.

Figure 3.2: Experience of being bullied, by age: 2017 [26]



Following this, Figure 3.3 shows the type of bullying children experience[11]. Bullying through social

[9]These procedures and mechanisms will be implemented and enforced prior to the release of software to the public. They will not be implemented before project closure in May 2019

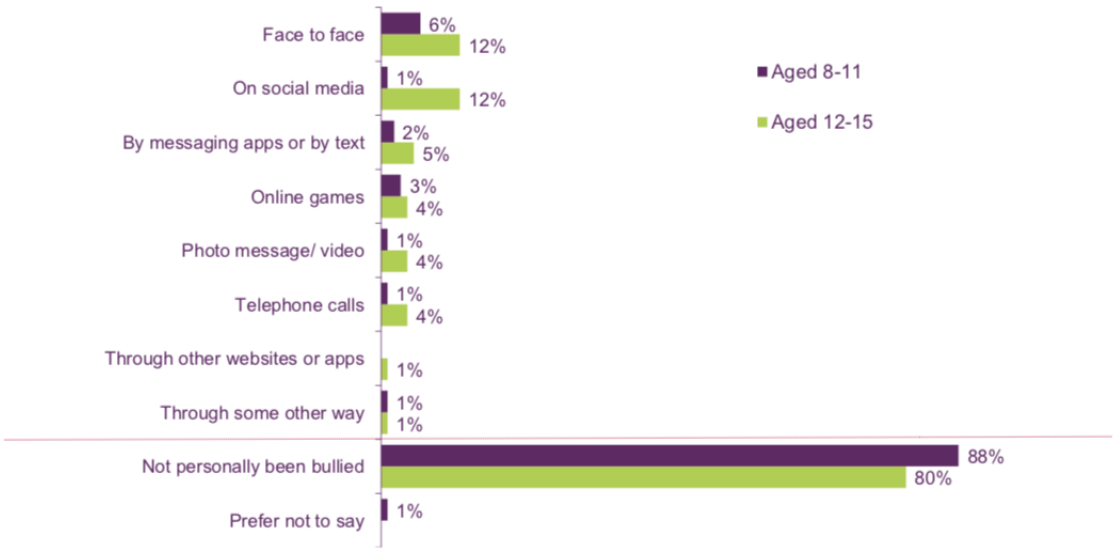
[10]Figure originally published by Ofcom in Children and Parents: Media Use and Attitudes Report 2017 [26]. Base: Children aged 8-15 who opted to answer the question (429 aged 8-11, 439 aged 12-15).

[11]Figure originally published by Ofcom in Children and Parents: Media Use and Attitudes Report 2017 [26]. Children were asked to respond to the question "When somebody was nasty or hurtful to you did it happen in any of these ways?" (prompted responses, multi-coded). Base: Children aged 8-15 who opted to answer the question (429 aged 8-11, 439 aged 12-15).

media platforms is just as common as bullying through face-to-face contact between children aged 12-15. Children aged between 8-11 are consistently less likely to have experienced any type of bullying.

Skouted is a social media application that allows users to upload videos clips and leave comments on clips they see. Given that children experience bullying through social media, messaging and video-content sharing applications most significantly, *Skouted* could be used as an environment to bully others.

Figure 3.3: Type of Bullying Experienced, by age: 2017 [26]



To combat bullying online, the *Skouted* platform intends to exercise the following procedures and mechanisms:

- Users will be told at sign-up and through regular reminders, what content and behaviours are expected and what constitutes abuse or misuse of the platform. Rules and community standards will be available and comprehendible by all users. The rules will be laid presented both textually and in the form of diagrams, videos and images so users of all ages are able to understand them.
- Users will be able to report users and content that they feel causes them distress or offence easily and transparently.
- Users will be able to block content and users they do not wish to be in contact with or exposed to. Users will be able to disable private chat facilities with a person and will be able to disable

comments being left on their posts.

Exposure to inappropriate or harmful content

Online services and social networks are increasing the number of young people being exposed to inappropriate or harmful content including pornography and illegal child sexual abuse images.

In a report by Livingstone[12] (2014) [27], when asked what children find upsetting online, pornographic and violent content constituted 40% of all children's main concerns. Livingstone also reports in this same document, EU children aged between 9-16 state video-content sharing platforms like YouTube and Social Networking Sites like Twitter and Facebook, as being the online applications where they are most likely to see inappropriate or harmful content. To add to this, Ofcom [26] reports:

> **One in ten 12-15's say they have seen something of a sexual nature online or on a mobile phone.**

Given that *Skouted* is a video-content sharing platform, it is necessary to enforce sanctions and mechanisms which prevent video-content outside of what is expected on the platform. The *Skouted* platform only expects football related content, i.e. actual match play or training footage, other content should not be viewable on the platform.

> **Three in ten 12 to 15 year olds and one in six 8 to 11 year olds who go online say they have seen online content that they found worrying or nasty**[13]

It is comforting to know that in another statistic released by Ofcom

> **At least nine in ten of 8 to 11 year olds and 12 to 15 year olds who go online would tell someone if they saw something worrying or nasty online**

Figure 3.4 shows who children say they would speak to when they have been exposed to online content they consider to be worrying or nasty[14]. A high proportion of children believe they'd feel most comfortable approaching a close family member if they saw something online that made them uncomfortable or worried. Others would seek support from their friends. The Figure also shows that 1% of children are unsure who they would tell, 4% of children do not know if they would tell

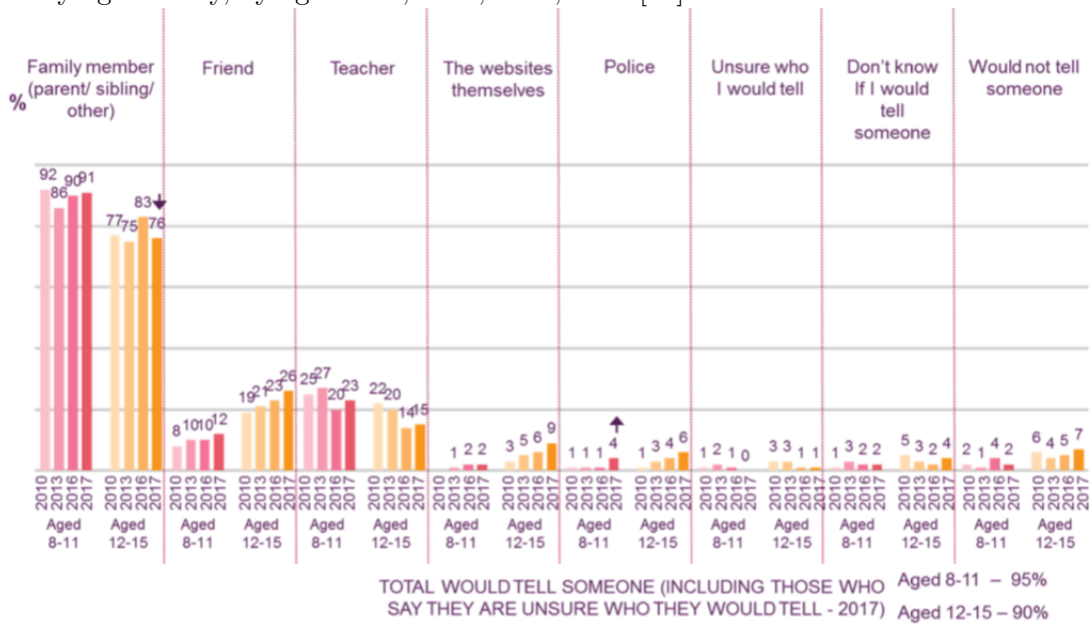
[12] S. Livingston, EU Kids Online, 2014

[13] As reported in Children and Parents: Media Use and Attitudes Report by Ofcom in 2017 [26]

[14] Figure originally published by Ofcom in Children and Parents: Media Use and Attitudes Report 2017 [26]

anyone and 7% of children would not tell anyone at all.

Figure 3.4: Who children speak to when they have been exposed to online content they consider to be worrying or nasty, by age: 2010, 2013, 2016, 2017 [26]



Based on these statistics, *Skouted* will make every effort to ensure content and user reporting mechanisms are transparent and easy to access. To combat exposure to inappropriate or harmful content through the platform, the *Skouted* platform intends to exercise the following procedures and mechanisms:

- All users should be able to report any user or content they are exposed, which they deem inappropriate or offensive by clicking the flag icon next to the clip or comment they have seen. Users are empowered to freely exercise their rights to freedom of expression and privacy.
- Users of the *Skouted* platform will regularly be reminded of their right to report content or users that worries or offends them.
- Information about what is acceptable behaviour and content, and how to report content or users will be provided through images, diagrams and videos, so younger users can understand what is acceptable behaviour.
- *Skouted* will enforce a content monitoring system. Content will be moderated to protect users from negative content or inappropriate behaviour from other users. Moderation will be proportionate to the level of competence *Skouted's* users are expected to hold, so not to

overstep the line between safeguarding children and violating their right to privacy and access to information.

Revealing Offline Location

Initial releases of the *Skouted* app will request users to state the football club they currently play for. This, along with other personal information, could be used as a basis to uncover the geographic location of a child or vulnerable person. In addition to this, when a user uploads a video clip of themselves and/or their friends, abusers can identify and locate them by recognising tell-tale signs of their locations, such as street signs and recognisable monuments.

Many camera apps on smart phones are location aware. This means they are capable of tagging images with longitudinal and latitudinal coordinates; this is known as geotagging. Geotagged or EXIF data embedded images and videos allow people to uncover a vast amount of information relating to location. Users of the *Skouted* platform are able to upload profile images and videos. Care must be taken to ensure that when these images are uploaded to the platform, the EXIF data and/or geotag is removed to prevent other users being able to locate the exact origin of the image.

To prevent users of the *Skouted* application revealing their geographic location, *Skouted* will take the following precautions:

- Implement an API which scrubs any geotags and/or EXIF data when a user uploads a photo or video to the platform.
- Equip users with the ability to limit the information they share, such as location data and football club, etc.
- Provide information and guidance on video posts, highlighting the importance of not posting videos that could reveal their location.
- Provide a parental control mechanism which allows parents and guardians to view and accept their child's posts before they are viewable by all users on the platform.

Sexual Exploitation of Young Persons

Sexual exploitation can include exposure to harmful or distressing content such as pornography and child abuse imagery or encouragement for children or vulnerable people to post inappropriate

images or videos of themselves [28]. There have been a multitude of reported instances where adults have used social networking sites and chatrooms online to prey on vulnerable children and people. These adults fail to use these services for their true intention and instead use it to groom vulnerable people for sexual abuse.

The purpose of *Skouted* is to connect young football players with scouts. Players are able to showcase their footballing talents in hope of being noticed by scouts and being selected to progress further in their professional career. Scouts are expected to be in a position of trust and any young footballer and their parent/guardian should have complete confidence that the person they communicating with is in fact a verified scouting official. All scouts that register with the application will be verified by provide a full and valid DBS check and all the necessary documentation proving their professional scouting authority.

The Home Office Task force on Child Protection on the Internet identified the following techniques to befriend and groom vulnerable children and people online [29]:

- Gathering personal information, such as age, name, offline location, contact number, name of school, name of football club and photographs.
- Offering gifts or material items including games, toys or music
- Promising meetings with sports idols or famous people
- Offering payment in return for the child to perform indecent and/or sexual acts
- Bullying or threatening the child, such as threatening to visit them as they 'know where they live' or contacting their parents.
- Asking to meet the child
- Sending indecent, explicit and or sexual imagery or videos to the child
- Masquerading as a child in order to gain their trust

The importance of protecting children and vulnerable users is substantial. *Skouted* will enforce the following mechanisms and procedures to safeguard its young and/or vulnerable users from sexual exploitation:

- Set young users' profiles private by default.

- Relay private messages sent between a child and another user to parents/guardians of the child.
- Monitor private messaging between scouts and children to ensure there is no inappropriate behaviour or activities.
- Provide regular reminders of online safety to users. This information should be understandable to users of all ages and can be presented in the form of text, images, diagram and/or videos.
- Provide users with a clear reporting and escalation process. All users should be able to report users who may be harassing them or behaving inappropriately. All reports of child sexual abuse should be taken with the upmost priority and handled with care.
- Users of the *Skouted* platform will regularly be reminded of their right to report content or users that worries or offends them.
- All scouts that register with the application will be verified by providing a full and valid DBS check and all the necessary documentation proving their professional scouting authority.

Mental Health and Wellbeing

Anxiety and Depression

The Royal Society for Public Health (RSPH) describes how social media use is linked with increased rates of anxiety, depression and poor sleep [30]. Anxiety and depression can a have detrimental impact on a young person's life. It can cause difficulty concentrating due to the overwhelming feelings of fear or worry. It can also make it hard for them to progress through education or work, as they are unable to perform to the best of their abilities.

Self-Image and Self-harm

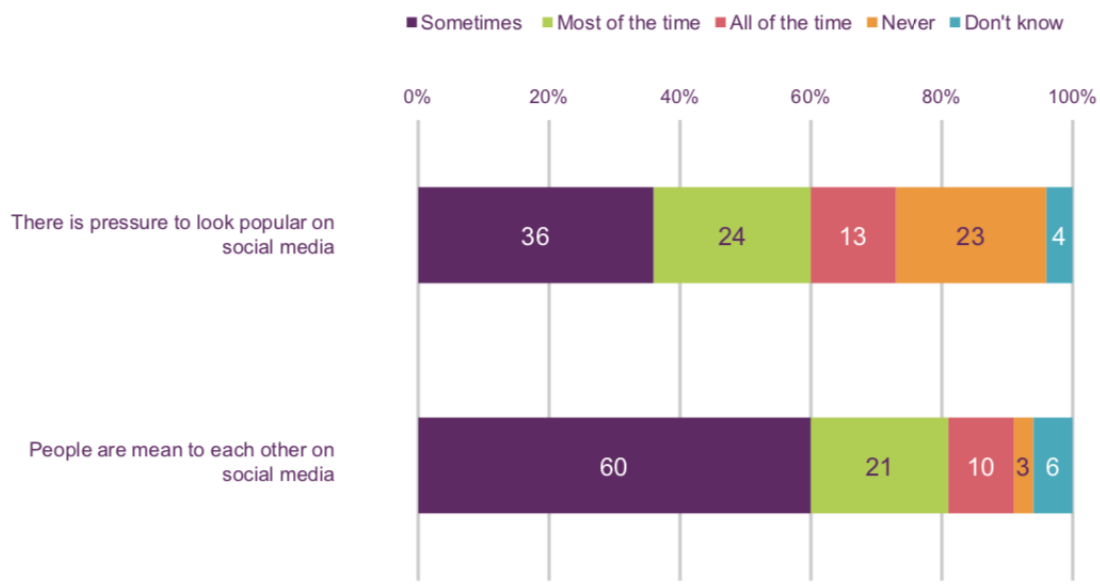
Images and videos posted on social media can set unrealistic expectations on body and self-image. Social media is commonly used as a platform to promote self-harm, eating disorders and even suicide. Being constantly surrounded by 'perfection' may leave people with feelings of low self-esteem and a compelling urge to become this image of 'perfection'. Pursuing perfection often manifests itself as an anxiety disorder such as Obsessive Compulsive Disorder (OCD).

As many as 9 in 10 teenage girls are unhappy with their body [31], and given the abundance of

images and videos they (and boys) able to make appearance-based comparisons with, it is clear why this statistic is so high. Figure 3.5 reveals the level of pressure influenced by social media[15]. 36% of children say declared they sometimes feel pressure to look 'popular' on social media. 24% experience this pressure most of the time, and 13% all the time. Nearly three quarters of children using social media feel some form of 'pressure' to look popular or 'perfect' when using social media. To add to this, 9 out of 10 children believe that people are 'mean to each other' on social media. This negativity (meanness) exacerbates anxiety, depression and negative self-image.

Skouted is a social media platform which enables users to upload images and videos of themselves playing football and training. Seeing so many videos featuring athletic and talented people could bring some *Skouted* users anxiety about their own image. Similarly, the platform allows football players to provide their weight and height measurements. This is another feature of the platform which could compel users to make comparisons between themselves and others.

Figure 3.5: The pressures and nastiness experienced by children aged 12-15 using social media: 2017 [26]



Skouted prioritises mental health and wellbeing and seeks to ensure all users are happy and are being exposed to only positive experiences when using the application. To promote a positive mental health and wellbeing, the *Skouted* platform will:

[15]Figure originally published by Ofcom in Children and Parents: Media Use and Attitudes Report 2017 [26]. Base: Children aged 12-15 with a social media profile or account (343).

- Introduce a pop-up heavy usage warning. Under the knowledge and acceptance of the user, *Skouted* will track the user's usage of the application and provide a pop-up warning when they approach or breach some level of usage that is deemed considerable and potentially harmful or worrying.
- Equip users with the ability to limit the information they share, such as weight and height
- Skouted will enforce a content monitoring system. Content will be moderated to protect users from negative content which promotes self-harm.

3.2.4 Exploitation of Children using *Skouted*

The question of whether *Skouted* exploits children is an intricate one to answer. The overarching aim of *Skouted* is to connect aspiring football players with scouts to help players advance in their professional career. Self-promotion is possible through the application and could bring further opportunities to players as they are no longer outside the scout's network.

The concern of *Skouted* exploiting children stems from the following idea:

A child dedicates a significant proportion of their time to trying to get noticed and eventually gets scouted at a very young age say 7 or 8. As a result of their success, they miss a significant amount of education due to training sessions and match schedules. At the age of 15, they get dropped from the team, but the amount of learning they missed out on makes it difficult for them to get back into the flow of education. Has *Skouted* exploited them?

The intention of *Skouted* was never to exploit children for financial gain and *Skouted* will always encourage equal effort between sporting activities, school and application usage. If a child is passionate about succeeding in the football profession, and they and their parents are confident they have a chance in reaching their potential, *Skouted* is in no position to stop them from trying to attain their goal. In most cases, footballing clubs promote education and provide players with the support and guidance to gain an education as part of their training program, thus maintaining the quality of education they receive.

Chapter 4

Requirements Analysis

This chapter outlines the initial requirements elicitation process, where the team met with the stakeholders to derive the platform's objectives, requirements and high-level UI design. Following this, Sections 4.2.1 and 4.2.2 present the functional and non-functional requirements derived from the elicitation phase and subsequent discussions with the customer. Tables 4.1 and 4.2 were first included in the specification report, and now have 11 additional functional requirements.

4.1 Requirements Elicitation

When creating the requirement specification for this project, several Skype meetings were conducted with the customer, Raj Sharma, in order to comprehend the high-level business requirements of the platform and translate these into technical requirements.

On 14th October 2018, a meeting was conducted where both the parties involved met in London. In addition to discussing the platform's objectives and requirements, high-level UI mockups and wireframes were delivered by the customer to have a better understanding of their intended user experience. The outcome of this meeting brought about a set of requirements which are detailed in the subsection below.

4.2 System Requirements Specification

The requirements have been chosen through discussion with stakeholders. There are two requirement classifications, Functional and Non-Functional. Any requirements that are necessary for this

project duration will be described using 'must', and desirable, non-essential requirements will be described using the term 'should'.

Each requirement will be prioritised by implementation necessity using the low - medium - high scale. High priority requirements must be implemented before this project closes, a medium priority requirement is desirable for this project duration though not essential and a low priority requirement is beyond the scope of this project but should be considered now for future software versions.

4.2.1 Functional Requirements

FRID	Requirement Description	Priority
1	The system must provide accounts for the different user entities 1.1 Scouts 1.2 Players	High
2	The system must allow the user entities to sign up to the platform 2.1 Require players to provide player information 2.2 Require players under the age of 13 to provide the contact details of a guardian in order to use the messaging features 2.3 Require scouts to provide scouting preferences 2.4 Scouts may optionally select and upload documents that verify their identity and membership to a professional scouting organisation	High
3	The system must require authentication to interact with the platform 3.1 Scout and Player profiles may be viewed without authentication	High
4	The system must allow players to modify their profile by allowing players to 4.1 Upload video clips 4.2 Upload a header video 4.3 Upload a profile picture 4.4 Modify their player attributes 4.5 Upload their match statistics	High

5	Players and scouts must be able to interact with other players in the following ways 5.1 Follow or unfollow other players 5.2 View posts by players they follow in their home feed 5.3 Comment on another player's post 5.4 Like or unlike another player's post 5.5 Visit another player's profile and view their posts and statistics	High
6	Players must be able to interact with scouts in the following ways 6.1 View scouts' details 6.2 View the verification status of scouts	High
7	Scouts must be able to interact with players in the following ways: 7.1 Contact players directly 7.2 View a player's videos 7.3 View a player's profile 7.4 Comment and like a player's video 7.5 Contact a guardian through email	High
8	The system must provide scouts with a criteria-based search mechanism of players	High
9	The system must facilitate communication through a message hub 9.1 Players and scouts may communicate with each other 9.2 Players and other players may communicate with each other 9.3 The platform may directly communicate with scouts and players 9.4 Communication with players under the age of 13 should be relayed to the player's registered guardian	High
10	On first login, the system should provide the user entities with the following 10.1 Players - Follow popular players, in contacts, or located nearby 10.2 Scouts - Show players that are similar to their general signup criteria	Medium

11	The system should allow scouts to modify their profile 11.1 The system should allow scouts to upload a profile picture 11.2 The system should allow scouts to verify upload documents to verify their identity and membership to a professional scouting organisation 11.3 The system should allow scouts to update their scouting preferences	Medium
12	The system should provide an administrative platform which provides the following functionality 12.1 A dashboard for moderating signups, flagged messages and malicious accounts 12.2 A dashboard providing usage metrics	Medium
13	The system should notify players when a scout views their profile	Low
14	The system should provide useful insights on a player's performance	Low
New Requirements		
2	The system must allow the user entities to sign up to the platform 2.5 Users should be required to select their Country of residence so the correct age restriction rules are applied.	Low
9	The system must facilitate communication through a message hub 9.5 The system must detect spam 9.6 The system must prevent users suspected of sending spam from using the message feature	Low
15	The system should provide an explore feature to allow players and scouts to discover new users 15.1 The system should allow users to find content from users they do not follow 15.2 The system should adapt to users preferences automatically using a recommender system	Medium

16	The system should allow users to exercise their rights to privacy 16.1 Personal information collected from the user will be limited to what is necessary for the use and delivery of the *Skouted* application 16.2 Explanations will be provided at sign up describing how and why their data is being collected 16.3 Explanations provided should be appropriate and meaningful to users of differing ages and competence. 16.4 Users should be notified by email if the information they have provided is being processed differently. 16.5 Fresh consent should be sought from users where a change in how their information is being processed has occurred. Their response can be edited in a settings profile. 16.6 User data should be stored securely. 16.7 Users should be notified if a security breach occurs. Meaningful details must be provided to all users dependent on their age and competence level. 16.8 Users should be able to request their personal images or video content to be deleted. 16.9 Users should be able to close their account and have their information fully deleted.	Low

17	The system should allow users to exercise their rights to freedom of expression and access to information 17.1 Content moderation should be proportionate to the level of competence *Skouted's* users are expected to hold. 17.2 The system should allow users to be identifiable under a pseudonyms or their legal name. This should be easily changeable in their settings. 17.3 The system should enforce an 'opt-in, opt-out' parental control mechanism. 17.4 Parental control mechanisms should be easily adjustable to adapt with the child's evolving capabilities. 17.5 Parental control mechanisms should be transparent to both the child and the parent. 17.6 *Skouted* will present its terms and conditions in an understandable and meaningful way.	Low
18	The system should allow users to exercise their rights to reputation 18.1 Users should be able to check, contest, rectify, delete and edit personal information stored about them.	Low
19	The system should protect a child's rights in accordance with evolving capacities 19.1 The system should aid understanding how and why their information is being collected in a meaningful way. 19.2 The system should provide easily adjustable parental control mechanisms. 19.3 The system should provide a competency test which deems whether parental consent should be requested.	Low
20	The system should provide a remedy for violations of rights 20.1 The system should provide post reporting facilities 20.2 The system should provide user reporting facilities	Low

21	The system should combat bullying online 21.1 The system should describe acceptable content and behaviours at signup, emailed reminders and pop-up notifications. 21.2 The system should allow users to block content and users they do not wish to be exposed to.	Low
22	The system should prevent offline locations being revealed 22.1 The system should implement an API which scrubs any geotags and/or EXIF data from uploaded images and videos. 22.2 The system should allow users to limit the information they share, such as location data and football club, etc.	Low
23	The system should display a heavy usage warning as a pop-up message.	Low

Table 4.1: Functional Requirements

4.2.2 Non-Functional Requirements

Priority	NFID	Requirement Description
High	1.0	The system should provide a mobile-first interface, supporting iOS and Android
High	2.0	The system should provide a discoverable user interface and familiar mobile user experience
Medium	3.0	The system should be maintainable and extensible
Medium	4.0	The system should be robust to failure and exhibit no downtime
Medium	5.0	The system should allow for scaling effortlessly
Medium	6.0	The system should comply with data laws such as the General Data Protection Regulation and the Data Protection Act

Table 4.2: Non-Functional Requirements

4.2.3 User Stories

User stories are created as a part of agile based development. These stories are what make up the backlog tasks for the project. The user stories for this project are defined in table A.2 and can be found in appendix A. These stories have been derived from both the functional and non-functional requirements, stated in Sections 4.2.1 and 4.2.2, respectively. The user personas described in the user story and the user storied themselves are defined in appendix A in table A.1.

Chapter 5

Project Management & Methodologies

The first half of this chapter will talk about the Project Management side of the project. More specifically, it will outline the roles for each team member; discuss the approach taken in order to deliver the project; discuss the key work streams and finally discuss how the team managed the customer relationship. The later half of this chapter will talk about the methodologies used for the project including the software development methodologies used, the software life cycle and version control.

5.1 Roles and Responsibilities

Each member in the team has been allocated the following roles, to ensure the project's success.

Project Manager

The role of the Project Manager is to ensure that the project is on track with the decided schedule, in addition to leading their team in order to complete the project in a timely manner and to the best of their ability. This role was designated to and performed by Ignacio Borrego.

Scrum Master

The role of the Scrum Master manages the workings of the Scrum meetings. They are to help each

team member decide what is manageable for them to do in a certain period of time; they are to help the team remain focused on the tasks at hand and they are to remove any hurdles that the team may face, preventing them from carrying out their task successfully. This role was designated to and performed by Harjot Singh.

Data Compliance Lead

This role is important to ensure that the project solution is compliant with the relevant data regulations, including the Data Protection Act and the more recent GDPR legislation[17]. This role was designated to and performed by Aliyah Stevens.

UX Design Leader

This role oversees the design of the user interface and ensures user experience is appropriate for a production-ready application. This role was designated to and performed by Samuel Ogden.

Software Testing Lead

This role entails overlooking all unit, integration and system testing, in addition to user acceptance testing. Their role includes creating test cases for each individual component as well as appropriate end-to-end tests and ensuring that the final product is fault-free. This role was designated to and performed by Akriti Pathania.

Meetings Administrator

The team's secretary will be in charge of taking minutes for group meetings, in addition to recording backlog features and who the feature is designated to. This role was designated to and performed by Akriti Pathania.

Market Research Lead

This role entails finding similar marketable applications and talking to scouts and potential players to understand the target audiences' needs. This role was designated to and performed by Daniel Alarms.

Code Review Lead

The lead code reviewer will ensure that the codebase meets industry standards. This includes enforcing a consistent style of code, in addition to a high level of understandability and maintainability. This role was designated to and performed by Daniel Alarms.

Ethics Manager

The Ethics Manager will identify any social or ethical issues that must be considered prior to the release of the application. All issues identified must be translated into implementable requirements. The Ethics manager will also overlook the communication between U18s and the testing team when it comes to user acceptance testing and ensures all participants aged 17 or less are not in any vulnerable position whilst using the system. This role was designated to and performed by Aliyah Stevens.

Customer

Raj Sharma is both the customer and sponsor for this project, providing resources, industrial expertise and the high-level business requirements. Raj is an entrepreneur and an angel investor and is a member of the start-up funding club [32]. He has previously invested in technological businesses and can therefore provide valuable industrial expertise to the project, co-founding start-ups such as the well-known HiyaCar [33].

Business Analyst

This role entails interacting with the business stakeholders and gathering, documenting and analysing their problems, needs and requirements. This role was designated to and performed by Harjot Singh.

Development Team

The role of this team will be to implement the project to meet the requirements set by the customer. This role was designated to and performed by every member of the team.

Testing Team

The role of this team will be to test the project against the test cases created. This role was designated to and performed by every member of the team.

5.2 Project Approach

The team followed an Agile-Scrum based approach having regular meetings every Monday, Wednesday and Friday, providing a progress update of their workings as well as resolving any problems and working collaboratively on the code base [34]. Moreover, the team held regular meetings with their

supervisor, Dr. Claire Rocks, discussing any queries relating to the project and ensuring that the project direction successfully aligned with both the academic and customer requirements.

5.2.1 Tools Used

For source code version control, the popular software Git [35] was used through the web interface GitLab.com [36]. Initially, a sprint branching strategy was undertaken, with a specific branch created for each bi-weekly sprint in order to enforce a clear separation of concerns throughout the code base. This allowed each sprint, which concentrated on a specific set of features, to be safely developed and merged with the main development branch of the code base, therefore constantly subject to integration tests to progressively ensure software quality.

GitLab.com [36] was also used as a project management software to manage the backlog of tasks. The project manager ensured that there was a suitable number of boards, with appropriate task deadlines and assignees, to allow the state of the project to be clear to the entire team and aid in obtaining a clear workflow.

After adopting sprints as the main development approach for the initial 8 - 10 weeks, the team collectively decided to adopt a Kanban-based workflow [37]. The main contributing factor to this decision was that the Kanban-based approach provided much more flexibility than compared to the strict 2 weeks constraint placed on tasks by the nature of the bi-weekly sprints. Kanban consisted in a reduced number of boards, as shown in Figure 5.1, where each member chose a task based on what they believed they were the most suited to perform. Additionally, the team was separated into dedicated front and back-end divisions, with Akriti Pathania, Aliyah Stevens, and Samuel Ogden forming the former, and Daniel Alarms, Harjot Singh and Ignacio Borrego forming the latter. The team collectively agreed to have these 2 divisions, as members were naturally specialising in certain areas of the code-base after completing front or back-end features. Therefore, this separation allowed the team to work more effectively with members completing features in their areas of expertise.

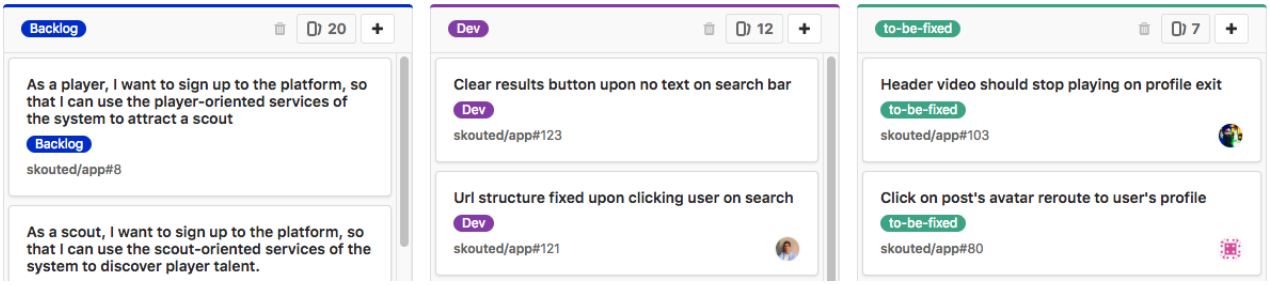


Figure 5.1: Example of kanban-style boards used

5.3 Key Work streams

5.3.1 Scheduled Meetings

Following the scrum methodology, the team had 3 weekly meetings each with their own stand-up. The nature of the meetings was as follows:

- **Meeting 1 (Monday):** Brief meeting of approximately 30 minutes duration with the purpose of performing a team "stand-up", where each member described the progress on their assigned tasks and any potential issues they may have been facing. Furthermore, the project manager discussed the main focus for the week, together with the deadlines that were due for the current week, in order to ensure that the project remained on schedule.
- **Meeting 2 (Wednesday):** Extended work session of approximately 2 - 3 hours, involving a variety of activities relating to the project. These typically involved a video conference with the customer, which occasionally consisted of a 20 - 30 minutes demonstration of the current progress made on the app. The feedback and perception of the customer were noted in the meeting's minutes, which were then used to iterate over the current features of the app and apply any advice or design preferences they may have to future features. The remainder of the session involved collectively and actively working on the project to design new features, resolve any potential issues, and often perform pair-programming to integrate certain front-end and back-end aspects. Minutes were also taken during these meetings, of which an example can be seen in Appendix G.
- **Meeting 3 (Friday):** The final meeting of the week replicates the structure of the initial

one, lasting 30 minutes and discussing the progress achieved during the week through a team "stand-up". Any tasks completed during the week were removed from the project boards, and any deadlines not met were discussed. The possible reasons for the delay in completing a task were discussed, and if deemed appropriate the deadline for the task was postponed due to its complexity, or the task was refined into several smaller tasks, each with their individual new deadlines.

Initially the meetings were designed according to the nature of the bi-weekly sprints, with Meeting 1 consisting of a sprint planning session for the following 2 weeks, and meeting 3 consisting of a sprint end and retrospective at the closing of those 2 weeks. Once the workflow was altered to adopt a Kanban style, the team adapted the meetings and used Meeting 1 to assign tasks often with a weekly deadline, and Meeting 3 for a discussion of the deadlines achieved or those requiring postponement.

5.3.2 Project Schedule

Table 5.1 conveys the overall project schedule, presenting a high-level overview of the tasks completed with their respective deadlines.

Task Name	Start (Date)	End (Date)	Duration (Days)
Specification	08/10/2018	24/10/2018	16
Infrastructure setup	24/10/2018	29/10/2018	5
Market and Analytics Research	24/10/2018	26/11/2018	33
Sprint 1: Wireframing, Test harnessing	29/10/2018	12/11/2018	14
Sprint 2: Image, Video Upload, Profiles	12/11/2018	26/11/2018	14
Progress Presentation Preparation	26/11/2018	04/12/2018	8
Sprint 3: Video Upload UI, Post creation	26/11/2018	17/12/2018	21
Progress Customer Demo	02/01/2019	07/01/2019	5
Epic 1: Feed UI, Feed fan-out	07/01/2019	21/01/2019	14
Epic 2: Search UI, Search back-end	21/01/2019	04/02/2019	14
Epic 3, 4: Sign up, Analytics, Explore	04/02/2019	18/02/2019	14
Continuous Integration Testing	24/10/2018	04/03/2019	14
Group Report	18/02/2019	26/04/2019	53

Individual Project Report	01/04/2019	26/04/2019	25
Final Presentation Preparation	22/04/2019	07/05/2019	15

Table 5.1: Schedule Start and End Dates

The schedule suffered no substantial changes with respect to the tasks set out in Table 5.1 of the specification for *Skouted*. The main alteration to the schedule was the order in which several sprints and epics were undertaken, as is clear from a comparison of the final project schedule in Figure 5.2. During development and as discussed with the customer, it was decided that the core features of the application were to be prioritised, and therefore the functionality for image and video upload was to be completed prior to any authentication procedures. Similarly, the UI and functionality to create match and training posts was to be completed before the main feed. Nonetheless, these tasks were still completed in the later epics.

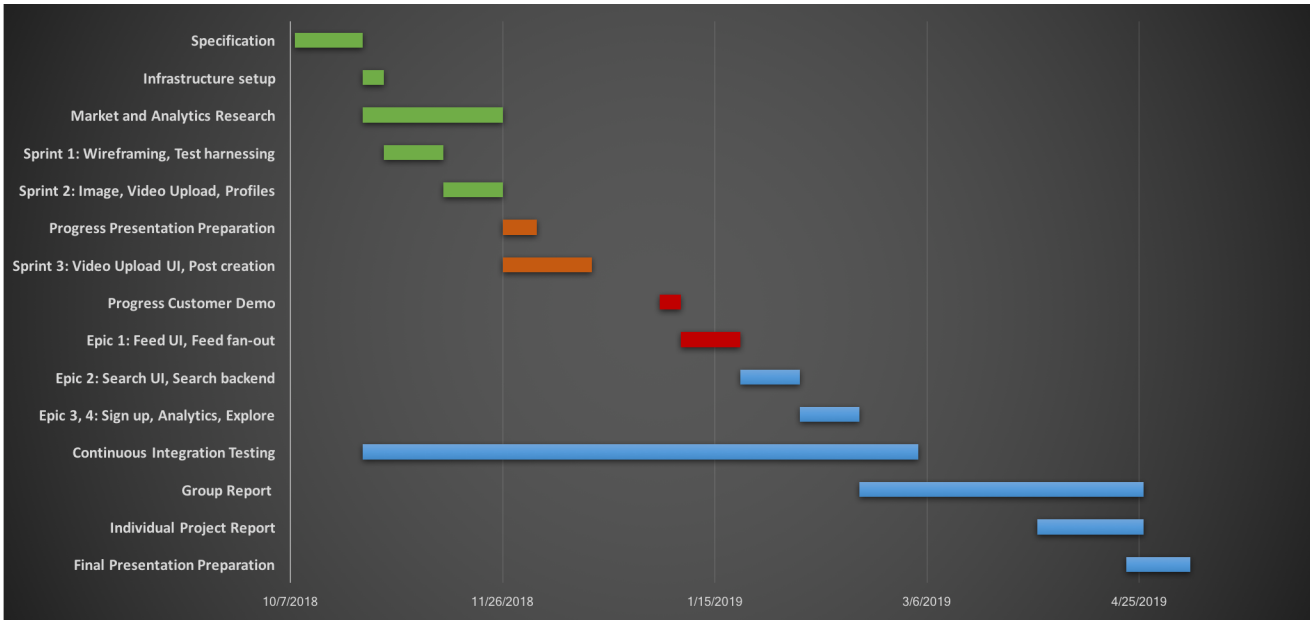


Figure 5.2: Project Schedule

When the team adopted Kanban and left behind the sprint-based approach, a natural way of grouping and scheduling tasks was using what is known as "epics". In agile methodologies, epics are large bodies of work that can be broken down into a number of smaller tasks and often represent a specific theme or area of work within the application's lifecycle [38]. For instance, in Figure

5.2, one can see how *Epic 1* relates to the feed of the application, and *Epic 2* to the Search functionalities.

5.4 Managing Customer Relationships

The project posed several challenges in relation to communicating and managing the team's relationship with the customer. The communication channels used were *Whatsapp* [39] and *Slack* [40] for written communication, and *Skype* [41] for video conferencing.

Balancing the academic requirements and objectives with the customer's business goals often was a complex process: though the academic and customer's specification of the project were aligned, the customer's perspective and feedback inherently influenced the overall direction of the project. A clear example of this was in the development of *Skouted*'s user interfaces for several parts of the application. The customer frequently placed a heavy emphasis on iterating over the user interface to achieve an appropriate market-fit and better suit the target audience. Whilst an effective look-and-feel was a functional requirement of the specification, it was often a laborious process to present several versions of a UI for an application segment, receive customer feedback, and iterate over the UI to achieve the customer's desired version. To prevent the project deviating from the schedule, a compromise was often reached over the communication channels to settle for a specific UI segment within the team's project schedule.

Another challenge faced over the course of the project was trying to achieve a balance between external academic deadlines and the agreed customer deadlines. The team agreed with the customer to have frequent, bi-weekly (if possible) video conferences to demonstrate the current progress of the application. The main issue faced in this scenario was that different team members studied different academic modules, and therefore had different academic deadlines, external to the *Skouted* project, making it difficult to align the team towards the video conference date. When this situation arised, the team solved it by speaking with the customer to alter the video conference's date slightly, with the customer typically being quite flexible to do so. Otherwise, the project manager would discuss each member's external academic workload and vary their internal project workload accordingly in order to successfully meet the video conference dates.

Overall, while sometimes challenging, the team benefited from constant communication with the customer to increase the overall quality of the software produced and the project itself. The

customer's domain knowledge of the football market was often used to consult on the design of features, to better adapt the platform to the target audience. With regards to the team, the described challenges increased internal communication and built upon positive team dynamics, qualities which became useful throughout the project's entire duration.

5.5 Software Development Methodology

The Agile-SCRUM methodology was used for the initial 8 - 10 weeks of the project, with Kanban being used for the remaining weeks as it was more suited to the team's schedule.

5.5.1 SCRUM

Agile-SCRUM makes use of sprints, which are defined time periods in which development of the system is carried out. The system requirements are divided into user stories which reside in the product backlog, within each sprint a number of user stories are selected to be completed in the space of the defined time period. These selected stories are placed in the sprint backlog or "story-board" which contains all the user stories being worked on in the current sprint. Once a user story is completed and reviewed it is removed from the storyboard and placed in the board consisting of closed user stories to indicate that no further work is required. This is repeated until all user stories are completed and the system has reached its completed state.

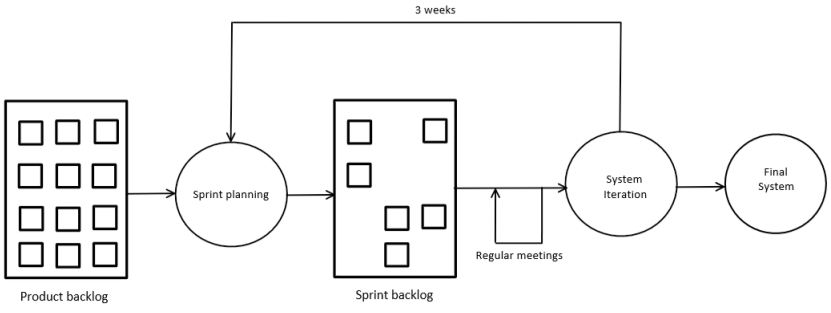


Figure 5.3: The SCRUM development cycle

Sprint planning

At this stage the sprint backlog is populated with new user stories to initialise the upcoming sprint. A meeting is held in which functions of the system are prioritised based on any changes in the requirements since the previous sprint.

Development

Development takes place over the course of two weeks. During this time, regular meetings are held to evaluate the current state of progression in development. Frequent contact is also made with the customer to ensure that development aligns with their requirements of the system.

5.5.2 Kanban

Unlike SCRUM, there are no sprints of fixed time. Rather, user stories are continuously pulled from the project backlog and placed in a queue containing the user stories to be completed. When a feature is being implemented, it resides in the development board showing which features are in the process of development. Finally, when completed, they are moved to the "in review" board where they stay until they have been reviewed and approved. Regular meetings are carried out during the process to monitor progress and make any priority changes. An example of the Kanban board can be seen in Figure 5.1, in Section 5.2.1.

5.6 Software Life Cycle

5.6.1 Current State

All high priority features have been implemented in the system, with the application being fully functional and satisfying the mandatory user requirements.

There are some features that need to be implemented in the future in order for the system to be complete such as the administration application, the web platform and restrictions on children aged 13 and under.

5.6.2 System Evolution

The evolution of the system is centred around the organisation of the sprints defined for the SCRUM software development methodology, and the features of highest priority. As well as this, particular elements of the system were dependent on others which created a partial ordering between the user stories.

Infrastructure Setup

The first sprint was dedicated to setting up the infrastructure of the system in order to be able to begin development. This involved setting up the Docker environment to ensure that the execution of the development plan would be as smooth as possible for all individuals in the project. Furthermore, the Google Kubernetes clusters was configured to allow the Docker containers to be accessible to multiple machines for testing and deployment. Finally, the Google Cloud Platform (GCP) services such as Firestore and Cloud Storage were configured to allow the system to store data in the future. Research was carried out at this time to determine what the application was required to do from a technological standpoint, and from this decide what infrastructure would be best.

Epics

Epic user stories, which can be seen in Appendix A, were used to describe the large functional elements of the system, and progress was followed based on which epics had been completed.

Profiles Epic The first and most fundamental epic user story to be completed was the profiles epic, which contained all profile functionality in the system. This included being able to generate, view and edit profile information, whilst also managing the uploading of media for posts and thumbnails. All other epics required the features implemented during this stage of development, so it was important to make sure this was dealt with first to retain consistency in future epics and the testing process. The objective was to complete the profile features within the first three sprints before moving on to the next area of the system, but development was carried over for a short period of time into the next sprint as they had not been fully completed to a satisfactory standard.

Feed The next step was to implement the feed features, which involved obtaining posts from users in the list of following and display them newest first and interacting with other users' posts.

This included implementing post liking and commenting. At this point, the concept of sprints had been replaced with epics and the goal was to complete these features within 4 weeks.

Search The search features were implemented alongside the features from the feed epic as many of the features required for this epic were independent of the feed epic. This allowed time to be regained from the delays during the profiles epic. It was estimated that the search epic would take two weeks to complete.

Explore The "explore" features were last to be implemented as they were dependent on the feed and search functions being fully functional. As there was not enough time to fully implement a recommender system using the research earlier in the project, Elasticsearch's search features were used instead to obtain content relevant to the user. At this point the system was at a stage where it could be demonstrated to the customers and various users for User Acceptance Testing.

5.7 Version Control

Version control was critical in ensuring the project's success, separating states of the system into multiple versions. GitLab was used to manage the software's git repository, with development branching from the 'dev' branch for each individual sprint. Each feature would then be worked on in a separate branch derived from the sprint branch, and then merged in once fully completed to integrate the new features into the system. A new version of the program would be generated at the end of each sprint including all the features to be implemented within the sprint. To finalise a version of the program, the sprint branch is merged into the 'dev' branch to ensure no further development of completed features. In the transition from SCRUM to Kanban, the sprint branches were changed into epic branches consisting of all the sub-features required to complete an epic user story. The version control management techniques used for the system remained unchanged however, with sub features branching from the epic branch and merging downstream to define a new version. The final stage of version control management involves merging the dev branch into the master branch, which defines a version of the system ready to be deployed in the public domain. This is only carried out once the system is ready, with all the necessary features having been implemented and working correctly. This was required when releasing the beta system for user acceptance testing to commence.

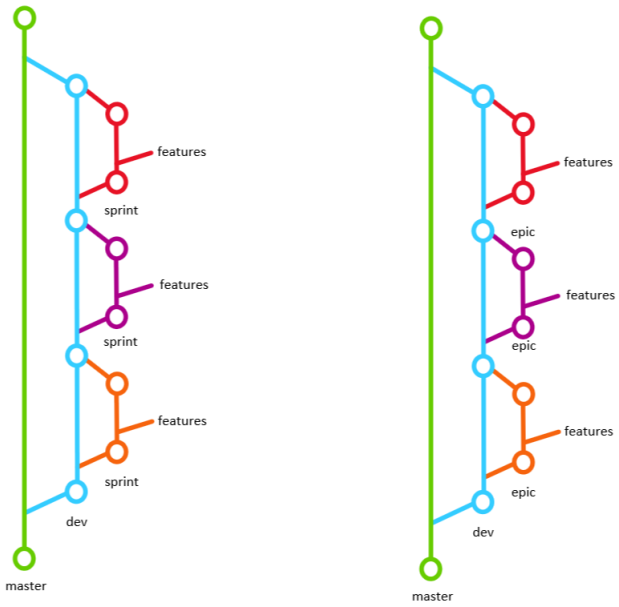


Figure 5.4: Version control management with sprints (left) and epics using Gitlab

Any feature branches to be merged into the sprint/epic branches required a merge request to be submitted and approved by designated reviewers. This was done to keep track of implemented features and to maintain the quality of the source code by preventing broken code being merged, which could disrupt the process of development.

5.8 Style of Code Implementation

There were various rules used to dictate the styling of source code for better readability by other programmers who may have to access the source code in the future and also, to optimise the performance and reliability of the system. Syntactical rules were enforced by ESLint, a pluggable linting tool [42] used to highlight sections of the source code that fail to follow the rules provided, whilst semantic rules had to be regulated manually. The syntactical rules used were the default ESLint rules provided and are present to improve readability of JavaScript coding.

5.8.1 Modularity

The system was designed to be modular, with global functions within a microservice being defined within the 'utils.js' file and global constants in the 'consts.js' file. Accessing any variables from these shared libraries required them to be imported using the 'import' syntax and in turn, reduced the

amount of duplicated code in the codebase. Functions are made to be as minimal as possible with a single-purpose, to reduce the complexity of the system. Informative names are used for variables, functions and classes so their intended purpose can be easily seen when using them across the system.

5.8.2 Reducing Unexpected Behaviour

To reduce the amount of unexpected behaviour in the system certain measures were put in place (semantic rules). Making use of the map() and reduce() functions rather than using a for loop for iterating through elements in an array or object is one of methods used to reduce the number of potential bugs in the system. This ensures that there are no 'ArrayIndexOutOfBounds' exceptions due to incorrectly defining the bounds of an array in a for loop.

The use of constants throughout the system ensures no variables are altered at any time during the execution of the program, which reduces the number of unexpected behaviours due to unexpected changes.

5.8.3 Code Readability

JSDocs are used for larger functions/elements in the system to describe their intended use, providing a description of the inputs and resulting outputs. By defining these, the programmer receives a tooltip which briefly describes the function when it is about to be used. They are also used for describing the endpoints in the system, which is useful in the integration stage, having to reference them from the front-end Controller library which sends requests to the back-end through these endpoints.

The default ESLint rules were applied automatically in the Visual Studio Code environment using the ESLint extension. This corrected minor errors such as incorrect spacings found within the source code and removing unnecessary code to improve readability and maintain consistent formatting throughout the codebase.

Functional components were used to define functions outputting React elements as stateless React components to divide the front-end elements into distinct objects. These objects could then be used like any other React elements in JSX syntax [43].

Chapter 6

Risk Management & Mitigation

This section defines a risk management plan to highlight the likelihood of risks occurring, project impact and response. This strategy will increase the likelihood of the successful project implementation and delivery. Sections 6.1 and 6.2 identify risks and opportunities that could affect the project, system or resources. Each risk will be allocated a risk score in accordance with their likelihood and impact category, and will be presented in the mitigation scheme provided for each risk. This section will then present table 6.6 which outlines if any of the identified risks occurred and the actual impact it caused. This table will also discuss the effectiveness of the mitigation scheme.

As the project progressed, continuous effort was made to assess the risks identified to ensure the assumptions made about their likeliness and severity remained the same and to check the predicted effects had not changed.

6.1 Risk Management

Tables 6.1 and 6.2 identify risks and opportunities that could affect the project, system or resources impacting schedule, scope, cost or the quality of the developed software. These risks and opportunities are separable in three categories as identified by Sommerville [44]:

1. **Project:** Risk/Opportunity that affect the project schedule or resources.
2. **Product:** Risk/Opportunity that affect the quality or performance of the software being developed.

3. **Business:** Risk/Opportunity that affect the organisation developing or receiving the software.

Since the specification report one additional risk (ID: 15) has been identified and mitigated against.

ID	Risk	Category	Likelihood	Severity	Risk Score
1	Project overruns deadline	Project	Possible	Significant	12
2	Project running behind schedule	Project	Likely	Moderate	12
3	Implementation beyond skills set of developers	Product	Possible	Moderate	9
4	Poor communication with stakeholders	Product	Unlikely	Significant	8
5	Hardware malfunction which causes loss of work	Product	Rare	Catastrophic	5
6	Key staff are ill or unavailable at critical times in the project schedule	Product	Likely	Low	8
7	A competitive product is marketed before the system is ready for release	Business	Unlikely	Low	4
8	Underestimation of system size and complexity	Product	Possible	Moderate	9
9	Client rejection upon reveal of the finalised platform	Product	Rare	Significant	4
10	Imbalance in workload distribution and/or contribution between developers	Product	Unlikely	Low	4
11	Disagreements surrounding product development strategies between developers	Product	Likely	Moderate	12

12	The software defect rate is excessive and repair time is underestimated	Product	Unlikely	Catastrophic	10
13	Changes in resources such as APIs causes unanticipated problems within the program	Product	Unlikely	Significant	8
14	Underestimation of system storage requirements	Product	Likely	Moderate	12
15	Project Management approach hinders progression	Project	Likely	Moderate	12

Table 6.1: Risk Register

Table 6.2: Opportunity Register

ID	Opportunity	Category	Likelihood
1	The development team gain insight and experience in a pseudo-real industrial situation	Business	Almost Certain
2	High potential for product commercialisation	Business	Likely
3	Opportunity to form a strong relationship with investors and their business contacts which could lead to future project endeavours	Business	Likely

6.2 Risk Analysis

Each of the risks identified in table 6.1 are assessed according to their likelihood and consequence of occurring, from this a risk score is calculated. Table 6.3 assigns a risk score to each cell in the matrix depending on the severity and likelihood, the higher the score, the worse the impact to the project's success. The scale ranges from 1 to 25. Table 6.4 shows the consequence of a risk occurring and the response that should be taken if the risk is seen.

Table 6.3: Risk Scoring Matrix

Severity	Catastrophic (5)	5	10	15	20	25
	Significant (4)	4	8	12	16	20
	Moderate (3)	3	6	9	12	15
	Low (2)	2	4	6	8	10
	Negligible (1)	1	2	3	4	5
		Rare (1)	Unlikely (2)	Possible (3)	Likely (4)	Almost Certain (5)
		Likelihood				

Table 6.4: Risk Scoring Matrix Legend

Catastrophic	16 - 25	Stop
Unacceptable	15	Urgent Action Required
Undesirable	8 - 12	Action Required
Acceptable	4 - 6	Monitor
Desirable	1 - 3	No Action Required

<table>
<tr><td colspan="3">Statement of Risk: ID 11: Disagreements surrounding product development strategies between developers</td></tr>
<tr><td>S</td><td>Consequence: (Cost, Schedule, Performance, Quality)</td><td rowspan="2">Risk Score: 12</td></tr>
<tr><td>NT</td><td>Time frame of risk: (Near-Term, Far-Term)</td></tr>
<tr><td colspan="3">Mitigation Strategy: Accept
Make decisions collectively and unanimously when possible
Vote where disagreements arise - Take the majority option</td></tr>
<tr><td colspan="3">Post Mitigation Risk Score: 12 (Likely, Moderate)</td></tr>
<tr><td colspan="3">Contingency Trigger and Action
Trigger: Disagreement between developers
Action: See supervisor for advice and reach compromise based on their judgement</td></tr>
</table>

Table 6.5: Risk Management Form: Risk ID 11

6.3 Risk Mitigation

A risk management plan is provided for each identified risk in table 6.1 and provides a scheme to manage them using the DOE template [45]. The mitigation strategy will take either a *reduce, accept or avoid* approach. Accepted risks are those that are expected to happen, contingency plans have been developed to provide a response to their happening. Table D.1 provides a sample risk mitigation strategies for the highest scoring risk identified. Mitigation strategies for each of the remaining risks can be found in Appendix D.

6.4 Risk Eventualities

Presented in table 6.6 are the actual eventualities of the risks identified in table 6.1. The table outlines which risks occurred and the impact they had on the project or product. This table also discusses the effectiveness of the mitigation schemes, which can be found in Appendix D.

Table 6.6: Risk Eventualities and Impact

ID	Eventuality	Impact	Comment
1	Avoided	None	This risk was mitigated against. The project scope was feasible and well defined in the project specification. All members of the team were assigned individual tasks and regular meetings were held to check each others progress.
2	Occurred	The team had more work to do on the project than expected. Implementation and integration of software components overran	This risk was controlled as enough slack time was factored into the schedule.
3	Occurred	Some team members had to learn new languages or technologies in order to contribute to the project.	This risk was reduced as the development team was divided into dedicated front-end and back-end developers. This reduced individual learning curves and improved project progression.
4	Did not occur	None	The Business Analyst arranged weekly meetings with the client to update them on the current state of the application and to receive their feedback.
5	Did not occur	None	This risk was mitigated against. No hardware malfunctions occurred during development but the teams frequent commit policy would have minimised the loss of work if it had occurred.

6	Occurred	When a team member or members were unavailable for any length of time, their workload was shared between other team members where possible.	This risk was controlled as enough slack time was factored into the schedule and other team members were able to take on a share of the missing members work.
7	Did not occur	None	No competitive product has been marketed.
8	Did not occur	None	This risk was avoided as the system architecture and data structures used were well considered and designed.
9	Has not occurred	None	This risk has not yet occurred. The team has maintained frequent correspondence with the stakeholders to gather the thoughts and feedback on the current state of the application. In doing this the likelihood of rejection is reduced.
10	Did not occur	None	This risk was avoided. Each team member was assigned tasks which were agreed by each member of the group. This avoided workload/contribution imbalance.
11	Occurred	Time was taken to discuss different development strategies.	This risk was accepted. Discussions regarding development strategies were held in the three times weekly meetings. These discussions were valuable as the best option for future progression was selected after deliberation.
12	Did not occur	None	This risk was avoided. Although software defects occurred, the defect rate was not excessive and repair time was factored into the schedule outlined in the specification.

13	Did not occur	None	No change of resources was required.
14	Did not occur	None	This risk was mitigated against. Data structure designs were well considered at the specification stage.
15	Occurred	Project progression was reduced for a short period of time while the sprint-based approach was being used.	This risk was mitigated by changing the project management approach to a Kanban approach. Following this, progression improved dramatically.

Chapter 7

Design Specification

This section explores and critically evaluates possible technologies, followed by a high-level design overview of the system architecture and data storage strategy. Finally, the section outlines any user interface design considerations and provides user interface mockups, preceding the implementation of the system.

7.1 Technologies

Research into appropriate technologies as carried out and evaluated, with a focus on achieving the functional and non-functional objectives within the time constraints of the project. Concisely, this subsection presents various options for developing a mobile application for both iOS and Android in a time-effective manner.

7.1.1 Front-end

The front-end refers to the user interface of the system. Several options exist for developing a modern user interface, each varying in complexity and time requirements.

Native Application

Native applications are those which are created for a specific mobile platform [46] and typically will use the operating system's underlying visual components to display a user interface. Traditionally, a native application requires programming in the language of the target operating system, but this

is no longer the case, with the advent of cross-platform technologies such as React Native [47] and Flutter [48].

True Native Most commonly associated with the idea of a native application, a true native application is built using the language and libraries of the target operating system. Thus, an app for both OS and Android would require two separate codebases, implemented in Swift and Java, respectively. The duplication of code that this entails is a disadvantage, and only realistically viable for larger teams. However, there are several benefits to this approach, namely, native applications have the highest performance, since the application is tailored for the platform, as well as typically providing a more intuitive user experience in line with the user's understanding of the user interface of the target operating system [49].

React Native React Native is a JavaScript framework developed by Facebook that allows for cross-platform development, using the functional idioms derived from the React library [50]. User interface elements are built in JavaScript in a similar way as developing a web application – as components. React Native components are all derived from the base components of the operating system, and therefore can provide the user with the same user experience benefits that a true native application can provide [47]. Although not as performant as True Native applications, React Native is still extremely performant. There is a single JavaScript codebase for both applications, making React Native a more viable option for rapid development, and bridges between native code and React Native code can be implemented to access the functionality of native modules with JavaScript [51].

Flutter Flutter is cross-platform framework by Google also touting native performance [48]. Notably, developing in Flutter requires the usage of Dart, an open-source, C-like language created by Google for mobile and web applications [52]. Although relatively new, Flutter provides many stunning user interface elements and patterns out-of-the-box, however the main barrier to usage is the Dart language, a language unknown to most.

NativeScript NativeScript is another common framework that enables the development of cross-platform, native applications, in JavaScript [53]. It has support for CSS and leverages the Angular and Vue frameworks used in web applications to ensure a smaller learning curve and reuse components that exist in both ecosystems. Although not as popular as React Native, the framework

offers commercial support options [54].

Progressive Web Application

A progressive web application (PWA) is a cross-platform mobile website that uses modern web technologies to provide a similar user experience to native applications [55], by making use of the latest APIs and features of web browsers. As part of the core guidelines, PWAs are installable and can work offline, offering a suitable avenue PWAs are built using the same technologies that any modern web application consists of, such as HTML, CSS, and JavaScript, and thus make use of the mature development ecosystem surrounding the web, such as CSS frameworks for rapidly developing user interfaces. The main disadvantage of PWAs arises from the poorer performance and limited access to certain functionalities of the device. An elicitation of the mixture of JavaScript and UI frameworks that were explored can be seen in below.

Angular Angular is a web framework developed by Google that acts as the controller and view in the model-view-controller (MVC) or model-view-viewmodel (MVVM) models [56] in traditional web applications. Views are built using Angular templates, an extension of HTML, with Angular-specific syntax and directives to bind data from the controller to the view [57]. Angular requires the usage of TypeScript, a compiled language that is a superset of and compiles to JavaScript, effectively adding a sound type system [58]. Angular provides the main benefit of being a fully-featured framework, including routing and other common functionality, however features a steep learning curve as well as forcing the use of TypeScript.

React React is a library developed by Facebook that allows the creation of reusable, modular components in JavaScript, sometimes referred to as the view in model-view-controller (MVC) architecture [59]. Compared to Angular, React is unopinionated and agnostic about technology choices, allowing a developer to select other technologies to work alongside React with minimum friction. Most importantly, the power of React stems from the approach of declaring HTML markup as part of an extension of JavaScript, known as JSX, and there is therefore no domain-specific language that a developer must learn [60]. Thus, React has a much lower learning curve than Angular, at the expensive of providing less functionality at the outset, but has an extremely rich open-source ecosystem, with many mature packages available for install.

Ionic Ionic is an open source, opinionated CSS framework used to build PWAs, with a large and stable component base [61] designed for iOS and Android. Ionic can be used with both Angular and React, however, Ionic's recommendation is to use the Angular framework, and therefore documentation for usage with React is scarce. Although Ionic contains many well-built components suitable for mobile user interfaces, development effectively requires the usage of Angular.

Framework7 Framework7 is another open source CSS framework that provides mobile-suitable components [62], compatible with React and Angular, by means of wrapping vanilla components implemented without the usage of a framework. Framework7 is extremely well-documented, with usage examples of every component available in React and Angular. It also does not force development with either framework. Although Framework7 is popular, it is largely used without a framework or library, where much of the community support is consequently oriented. Additionally, Framework7 was not initially architected for use with other frameworks, so may be more prone to bugs.

Hybrid Application

Hybrid applications combine the concept of developing a mobile website or PWA with a native container [63] that provides the PWA with access to the device's underlying APIs. Apache Cordova [64] is a free, and open source engine, offering such functionality in the form of JavaScript plugins. Thus, developing a hybrid application can only enhance a PWA, at the cost of introducing extra developer tooling. Apache Cordova is extremely popular and provides a better user experience over developing a pure web application, however, it introduces development overhead, as it is not possible to debug or simulate the native functionalities that the plugins expose without directly using the target device.

Evaluation

Although all the aforementioned technologies cannot be compared directly, it is possible to evaluate the appropriateness of each technology. A summary of the various attributes can be seen in table 7.1.

Ease of development refers to the learning curve and straightforwardness of a technology. True Native was determined to be complex, as it is required to maintain two separate codebases for iOS and Android. The best technologies in this category were React and React Native, both allowing

for cross-platform development, with no additional domain-specific language to learn.

Reusable team knowledge determines whether the technology fits into the team's existing knowledgebase of web technologies, which can influence factors such as code quality and development speed, particularly where the technology is completely new to the team. Web frameworks such as React, Angular, Ionic, Cordova, and NativeScript were most applicable to the team's current knowledge, whereas Flutter and True Native are developed in languages unpractised by the team.

Speed of development is determined by how quickly it is possible to progress the development of the application. True Native requires simultaneous development for both iOS and Android, thus, results in the slowest development speed. React and NativeScript are both cross-platform, component-oriented frameworks with large component ecosystems that will contribute positively to development speed.

Maintainability is a crucial aspect evaluating how extensible the application is, with respect to future changes and additions. True Native is the least maintainable, as any changes and fixes must be duplicated across two separate codebases. Additionally, it is possible for one codebase to contain bugs that the other does not. React offers the highest form of maintainability, due to a strong component-oriented architecture and a relatively lightweight library that empowers the developer to compose the application as desired.

Without clear and concise documentation, development can be frustrating and excessively time-consuming. True Native, React, and React Native are all technologies surrounded by a mature ecosystem, due to their longevity, and thus, have the clearest documentation. It can be noted that Ionic does not have documentation for usage with React, thus effectively forcing development in Angular.

As a final verdict, React and Framework7 were deemed to be the most appropriate technologies for developing the front-end, both complimentary, providing sufficient documentation, close to the knowledgebase of the team, and providing a satisfactory speed of development.

	Ease of Development	Reusable Team Knowledge	Speed of Development	Maintainability	Documentation
True Native	Low	Low	Low	Low	High
React Native	High	Medium	Medium	Medium	High
Flutter	Medium	Low	Medium	Medium	Medium
NativeScript	Medium	Medium	High	Medium	Medium
Angular	Medium	Medium	Medium	Medium	Medium
React	High	Medium	High	High	High
Ionic	Medium	Medium	Medium	Medium	Low
Cordova	Medium	Medium	Medium	Medium	Medium

Table 7.1: Rating of attributes for each front-end technology

7.1.2 Back-end

The back-end of the system refers to the service(s) that carries out all processing, which the front-end will communicate with to store data and execute user actions. This section will outline and evaluate the architectural choices available, as well as language frameworks suitable for developing web services.

Architectures

The architecture of the back-end will determine issues such as the overall structure of the codebase, modularity, and scalability. The various architectural options are explored, below.

Monolithic Monolithic architecture involves developing tightly coupled code to serve the needs of the system [65], grouping all the functionality the system may provide.

The main advantages are drawn from the simplicity of development, as monolithic architectures result in a singular codebase for the entire team, simplified testing procedures, and straightforward deployments [66].

The downsides of monolithic architecture become apparent at scale, as a result of the tight coupling. Updates to any component of the monolith requires a full redeployment of all code, and it is wasteful to horizontally scale the entire application when only certain components are frequently placed under load [67]. Additionally, all code must be retested for unrelated functionality, as changes could affect the entire application.

Microservice Microservice architecture improves upon monolithic architecture's shortcomings by introducing loose coupling, splitting the codebase into services separated by functionality [68]. These services are developed and deployed independently.

Consequently, the microservices architecture has a number of advantages over monoliths. They can be tested, deployed and scaled without any dependency on the rest of the codebase, thereby allowing for an increased development-deployment cycle. Microservices are relatively small, forcing modularity and separation of concern, thus increasing the readability and focus of code. Development can occur in increments, as microservices can be added when required. All of these advantages result in increased scalability of microservices, as components of the system can be independently scaled [66].

Microservices come with disadvantages surrounding the complexity of providing the additional flexibility. Services must be orchestrated to work together, requiring complex infrastructure. Calls to other microservices will also be less performant than their monolithic equivalent, as the request must be passed over the network [66].

Serverless Serverless architecture, also known as functions-as-a-service, involves the deployment of functions which are executed upon an event trigger, such as a user accessing a URL [69].

Crucially, the need to manage infrastructure and software required to execute a back-back-end is abstracted away, and the developer simply focuses on writing lightweight functions. This greatly simplifies the process of deploying a complex microservices-based system [66].

Serverless services are provided by leading cloud providers such as Amazon's AWS and Google, and cannot be provisioned by other providers, creating a dependency on the vendor. Additionally, serverless is event-driven, and therefore scaling is no longer adjustable by the developer.

Evaluation Table 7.2 compares the scalability, maintainability, flexibility, and complexity of each architecture. Scalability is the ability to increase the load that the service(s) can handle. Monoliths are the least scalable, with tight-coupling interfering with the ability to horizontally replicate a service. Microservices alleviate this entirely, and thus are extremely scalable. Maintainability refers to the ease of modifying the existing code to add new features or update components. Again, the tight coupling of monoliths means that any changes require extra care to ensure that unrelated code in still unaffected. Microservices can be modified and tested independently, so are the most maintainable. Complexity is determined by the steps required to develop with and deploy an architecture. By far, microservices are the most complex, needing complex development-operations (DevOps) pipelines and orchestration to deploy multiple services. Monoliths provide the simplest path to deployment.

	Scalability	**Maintainability**	**Complexity**
Monolith	Low	Medium	Low
Microservice	High	High	High
Serverless	Medium	Medium	Medium

Table 7.2: Rating of attributes for each back-end architecture

Microservices were consequently selected as the most appropriate choice, as the back-end will likely

handle a broad spectrum of unrelated functionalities.

Languages and Frameworks

Suitable languages with mature web development frameworks were evaluated, below.

Node.js (Express) Node.js is a cross-platform, open source JavaScript runtime environment used to develop server-side applications in JavaScript [70]. Applications are developed in the form of modules, and other Node modules can be installed with the npm package manager.

Node is particularly suitable for writing high-performance web applications, as a consequence of the non-blocking, asynchronous event loop it implements [71]. Additionally, the usage of JavaScript on the back-end reduces the mental fatigue of developing in a different language to the front-end.

Express is the most popular framework for composing web applications in Node [70], providing simple mechanisms to define endpoints and the corresponding code to run. Express is extremely flexible and can be easily customised by adding middleware to handle common operations, such as authentication.

Python (Flask) Python is a cross-platform scripting language featuring a simple yet powerful syntax, and access to a large repository of packages. Python is known for being a productive language to develop in [72] and offers a fast test-develop-deploy cycle.

Flask is described as a Python microframework [73], and similarly to Express, offers decorators for annotating functions to execute with endpoint declarations. Flask is extremely simple to use but does not offer the same flexibility as Express.

Java (Spring) Java is an object-oriented, cross-platform language that compiles to bytecode, and is executed by a virtual machine installed on the target platforms [74].

Spring is an open source framework for creating enterprise Java applications [75], containing everything needed to develop a web application, from WebSockets, all the way to writing a web-facing user interface with the MVC model.

Evaluation Each of the languages and corresponding frameworks were compared and evaluated, summarised in table 7.3.

	Productivity	Documentation	Performance	Modularity
Node.js (Express)	High	High	High	High
Python (Flask)	High	High	Low	High
Java (Spring)	Medium	High	Medium	Medium

Table 7.3: Rating of attributes for each back-end language and framework

Productivity is defined as the speed of development with a framework or language. Node and Python with their corresponding frameworks are the most productive languages, with both offering a simple development, both being interpreted languages, and seamless frameworks for creating web endpoints. The Spring framework is bloated for most needs, requiring lots of boilerplate code in order to produce a functioning web server.

All 3 languages and frameworks are documented extremely well.

Python has the slowest performance, as a result of its single threaded architecture. Spring has a reasonable level of performance; however, the Java Virtual Machine requires a relatively significant amount of memory. Node's performance is similar to that of Java but does not claim the same resources.

Both Python and Node are extremely modular, with a well-established, centralised package repository available for each language. External packages in Java can be facilitated by installing additional tooling, introducing a more complicated development and compilation process.

The language and framework selected was Node and Express, a strong combination of high performance, productivity, and flexibility.

7.1.3 Datastore

The system will be required to store data in order to persist information about users. The numerous technologies for data storage were discussed and assessed.

Relational Database

A relational database stores a structure representing the relations between data [76]. The data is stored in tables, which can be queried for information, and new information can be easily derived by applying operations such as joining two tables together. Most relational databases implement

the SQL language, used to express operations for the storage and retrieval of data.

MySQL MySQL is the most popular open source database management system, backed by Oracle [77]. MySQL has robust transactions support, is performant and simple to use.

PostgreSQL PostgreSQL is a fully open source database management system that provides more advanced features than MySQL, such as materialised views and high-speed reads [78].

SQLite SQLite is a file-based, self-contained, serverless database, suitable for embedding within applications that require local storage, with the same benefits of using a relational database [79]. SQLite should not be used in circumstances where concurrent reading and writing is important, such as in the back-end of web applications.

Non-Relational Database

Non-relational databases do not conform to a rigid schema, unlike their relational counterparts. They do not support common operations such as table joins, but this allows for high-speed reading of data.

Cloud Firestore Cloud Firestore is a NoSQL solution by Google featuring automatic scaling, replication, and high-performance, as well as some of the traditional features of relational databases, such as transactions [80]. Data is organised into collections, which contain documents. Documents can also contain sub-collections.

Static Files

Media data such as photos and video are not suitable for storage in a database, thus a static storage solution is essential.

Local Filesystem The local filesystem can be used to save videos and photos to the disk. This is a simple approach, but in a distributed cluster, is not scalable, as servers may be in different physical locations to the stored data, resulting in slow reads and writes.

Google Cloud Storage Google Cloud Storage is an online storage provider specialising in the high-speed retrieval of data across replicated areas [81].

Content is uploaded to a Cloud bucket, and automatically distributed across multiple regions. This operation returns a URL which can be used to access the object. Google Cloud Storage will return the content from the closest located server, providing extremely high performance.

Evaluation

As shown in table 7.4, the attributes of speed, availability, and scalability were contrasted.

Availability refers to how accessible the data is. In the relational database category, SQLite has the lowest availability, as it is only available if there is access to the filesystem where it is stored. PostgreSQL and MySQL have the highest availability, as they run independently on servers. In the non-relational category, both MongoDB and Cloud Firestore are highly available. For object storage, the local filesystem provides low availability, with media being only accessible on a single server. Cloud storage is highly available, offering replication across various regions and interactable through an API.

Speed of data access is extremely important in achieving a positive user experience with the fast display of data. Cloud Firestore provides the highest speed but has many limitations around querying data in a non-trivial manner, both consequences of NoSQL technology. Cloud Storage replicates content across multiple regions, and so users will be served content from closer physical locations, providing the highest speed.

Scaling in Cloud Firestore is abstracted away as the service is seamlessly managed by Google Cloud Platform. SQLite is unscalable, designed for use a single instance of an application. Additionally, Cloud Storage is managed by Google, also abstracting away any scaling concerns.

	Availability	Speed	Scalability
MySQL	High	Medium	Medium
PostgreSQL	High	Medium	Medium
SQLite	Low	Low	Low
MongoDB	High	High	High
Redis	High	High	Medium
Firestore	High	High	High
Local Filesystem	Medium	Medium	Low
Google Cloud Storage	High	High	High

Table 7.4: Rating of attributes for each data storage method

In conclusion, Cloud Firestore for structured, textual data, paired with Cloud Storage for media data were selected for data storage.

7.1.4 Operations

In addition to selecting technologies for development, a production-grade application requires rigid processes for deployment and managing operations.

Docker

Docker is an open source, lightweight container technology with the same benefits of virtualisation, however, requires less system resources as it does not execute an entire operating system [82]. Instead, Docker uses shared layers, but isolates the execution of processes within a single environment, known as a Docker container.

Docker images are specified with the creation of a Dockerfile and can then be used to build containers. These containers are lightweight images that can then be executed in any environment that runs the Docker Engine. Thus, Docker utilises resources more effectively, and offers the same security and modularity benefits that a traditional virtual machine does.

Kubernetes

Kubernetes is an open source container orchestration platform, used to coordinate the deployment of Docker containers while abstracting away hardware resources [83]. Kubernetes pools the to-

tal resources available across multiple servers and automatically schedules deployments onto the appropriate physical server, based on a deployment definition supplied for a service. Kubernetes also routes incoming connections to the correct containers, with support for load-balancing across multiple instances of a container.

Google Kubernetes Engine is an offering that provides a managed Kubernetes cluster, and is commonly used for production Kubernetes deployments [84].

Gitlab Continuous Integration

GitLab Continuous Integration is a service that executes commands on certain triggers, such as a commit to a specific branch [85]. This will be used to build Docker containers, run tests, and deploy these containers to Google Kubernetes Engine on success.

7.1.5 System Overview

Making use of the evaluated technologies, the system was broken up into modular components.

Data Storage Strategy

Data will be stored in a denormalised fashion, where the microservices will process the data and store the output. The output will then be duplicated into all collections which may need to reference the output. Since the output is already available, clients can consume the data at high-speed, as no processing additional to reading the data is required.

Two primary data root collections will be created: *posts* and *users*. The *posts* collection will be the source of truth for any user posts, and each post document will be duplicated into the feed sub-collection of any user following the poster.

The *users* collection will store all user profiles, with each document containing the user's information and further sub-collections for their feed and own posts.

Back-end

Search As Cloud Firestore does not support full text search [86], Elasticsearch will be deployed to facilitate this. A n-gram tokeniser will be defined to index user names and email addresses [87]. This allows Elasticsearch to quickly match partial searches against a precomputed index of all the searches that are possible for a user's name or email address.

Feed The feed microservice solely supplies a user's feed and the data for the explore feed. Users will spend most of their time accessing their feed, and as such, the feed service will most likely require scaling. The feed microservice will support pagination, and given a timestamp, will only load a set number of posts to ensure that the service is always responsive.

Media The media microservice will accept picture uploads of any format, and video uploads in MPEG-4 format. It should carry out validation checks on the content, determining if a video is too large or the resolution of a photo is too small. Additionally, the media microservice will produce a thumbnail of an uploaded video clip, provided a timestamp.

Profiles The profiles microservice provides login and signup functionality for users. On login, the user is provided with an encrypted JSON Web Token as part of the Open ID Connect standard, generated by Keycloak, an open source identity and access management solution[88] that will be deployed. The JSON Web Token will be supplied by any client with their requests to any microservice, and the microservice will be able to decrypt the token and obtain the user's identity information without verifying the token with Keycloak.

The profiles microservice also allows the user to update their profile, storing the changes in Firestore. To update a profile picture, the microservice will upload the picture to the media microservice, which will return a URL to the new picture. The user's document is then updated with the new URL of the profile picture.

Messaging The messaging microservice will be responsible for fetching all the threads for a user, where a thread is a list of messages between two users. This will be retrieved from Firestore and will support pagination, where the most recent messages will be loaded. Threads will be stored as a sub-collection under the user's document, and each thread will contain a sub-collection of the messages between the two users.

Posts To upload and retrieve posts, the posts microservice will implement endpoints that interact with the media microservice to upload videos to Google Cloud Storage. Uploaded posts will then be duplicated into the feed sub-collection of followers. Comments and likes are stored in separate sub-collections of the post document, allowing for a immediate listing of users which have liked or commented on a post.

Orchestration

Docker will be used to wrap each microservice and the front-end into execution-agnostic containers, which will be connected together using the Docker Compose tool in development and Kubernetes in production. This allows individual services to operate independently of each other.

The system will be deployed using Google Kubernetes Engine [89], which allows for the administration of clusters.

Front-end

The front-end will be developed using React, and a separate controller to handle interactions across the user interface. Framework7 will be used to create mobile-compatible components that are reusable, such as buttons.

Portability

The use of Docker allows the containers for the system to run on any host with the Docker Engine installed locally, and on Cloud providers. Using the docker-compose command provides a simple mechanism for launching all the microservices on any system, and the Docker containers can be deployed to any system.

The system does have a dependency on Google's Cloud Firestore and Cloud Storage products, but the relevant buckets and locations can be defined upon deployment.

7.2 System Architecture

As previously discussed, the system was designed and architected with a microservices-oriented approach. Each microservice was assigned a specific functionality, interacting with other microservices to achieve a loosely coupled architecture. This can be seen in table 7.5.

Name	**Description**	**Depends on**
Profiles	• Responsible for all user-related actions • Creating a user • Updating a user's information • Updating a user's profile picture	• Media Microservice • Cloud Firestore
Search	• Search for name and email address, with fuzzy matching • Search for players with criteria: age, height, traits, level played at, foot	• Cloud Firestore • Elasticsearch
Media	• Responsible for converting and uploading media to Google Cloud Platform	• Cloud Storage
Posts	• Responsible for creating posts • Pushes posts to each user • Pushes comments and likes to each user • Retrieve post information • Retrieve comments and likes of a post	• Cloud Firestore • Media Microservice
Feed	• Retrieve user's feed • Retrieve the explore feed for a user	• Cloud Firestore • Elasticsearch
Messaging	• Fetch user's threads • Fetch messages in a thread • Create new thread	• Cloud Firestore

Table 7.5: A description of each micro-service

Each microservice will also be communicating with external services such as Cloud Firestore and Cloud Storage for storing and retrieving user data and media.

7.2.1 System Architecture Diagram

The system architecture diagram can be seen in Figure 7.1. The diagram displays the service dependencies outlined in table 7.5.

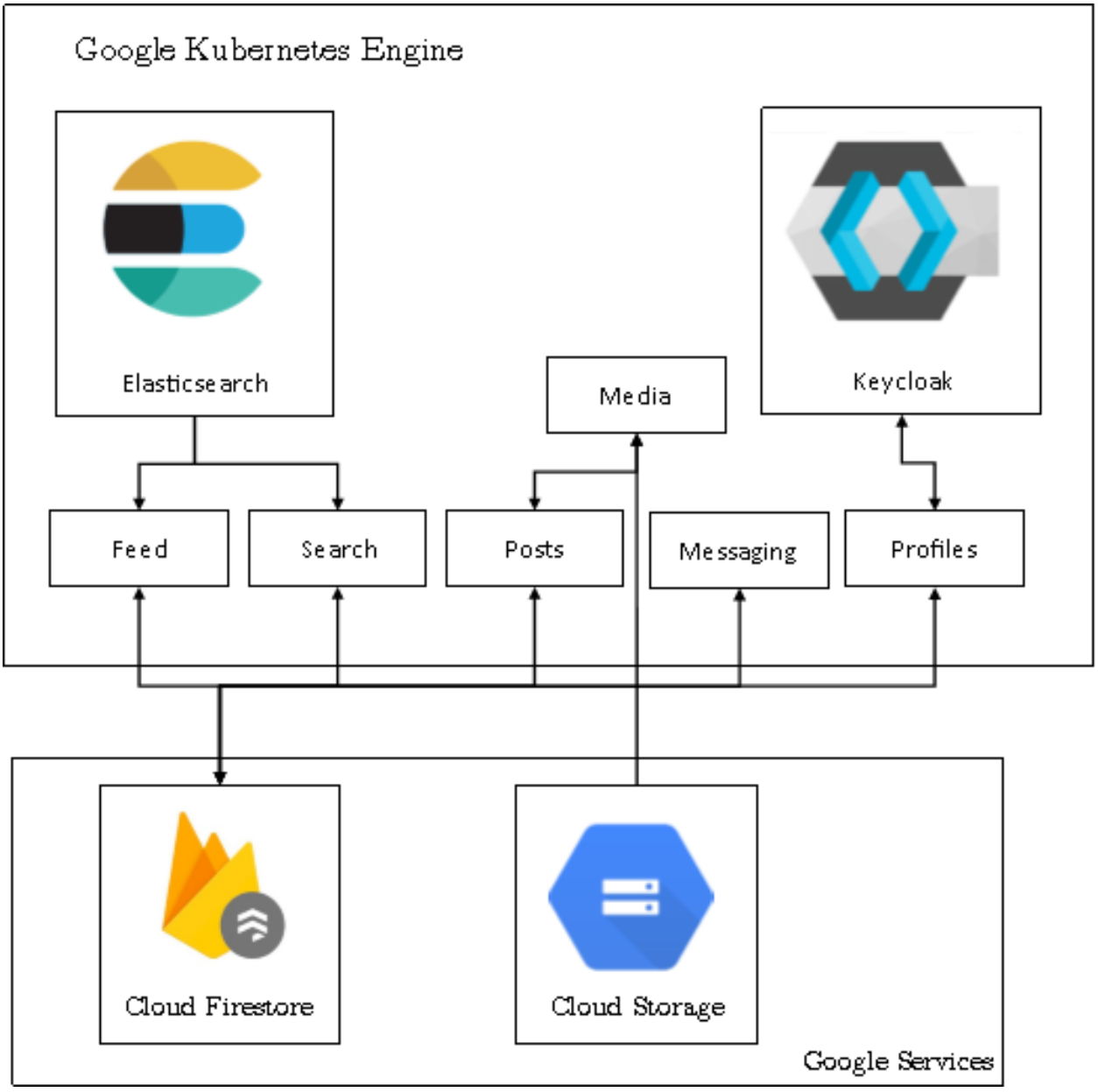


Figure 7.1: System Architecture Diagram

7.2.2 Data Push Architecture

The system applies a data push mechanism, whereby changes to user data and posts are pushed out to the followers of each user. When a post is created, the system should duplicate the post into the corresponding collections of each user, and any changes will also be propagated.

The advantage of this mechanism is that the services will be able to display the user's match feed with increased performance, as no additional lookups or data processing must be carried out in order to retrieve the final data that the user requires.

All profile data must also be pushed out and copied into the user's original post. This is to ensure that viewing the attributes of the post, such as the number of likes and comments is extremely fast.

7.3 Design Considerations

To ensure that the design process is efficient and will result in a usable product, certain considerations must be made. These considerations are outlined in this section.

7.3.1 Audience

The audience of the *Skouted* app will be both young football players (aged 13 to 20 years) and older, more professional scouts (typically aged 30 years or more). This wide age range of users introduces a challenge in terms of design as the product needs to be child friendly while also having a professional look for scouts.

7.3.2 Platform

The *Skouted* app is being developed as a mobile application which comes with many issues that must be addressed.

Screen Size

One important factor when designing for mobile devices versus desktop browsers is the limited amount of space available to work with. Mobile devices have a width of around 5 inches diagonally whereas laptops are typically larger than 15 inches diagonally [90]. With this limited amount of room, the amount of text and interactive elements on a page needs to be greatly reduced compared

to browser application. It can be seen by observing mobile applications such as Instagram and Facebook which have had tremendous success in the mobile world that having only a few buttons (4 to 5) across a page allows easy navigation as they can easily be tapped by a fingertip without touching any other elements. Additionally, mobile devices come with unique gestures such as swiping, 3D touch, pinching to zoom and pull down to refresh which can be employed to help save space. When designing the *Skouted* app for mobile, it will be essential to take these into account as well as using clean graphics and interactive elements that afford their designed role.

Devices

Mobile app design is generally specific to each device. This is because different devices have different operating systems which require slightly different user interfaces. The top mobile operating systems are iOS and Android which lead the market with a combined 97.3% share in 2018 [91]. Although it would be safe to design solely for these two operating systems, there are many different devices running these operating systems all with different screen sizes, resolutions, ratios and store and icon requirements. It would be essential to ensure that the design for the *Skouted* platform is flexible to fit into different sized devices and also be future proof for new, unique devices running on these operating systems. Generally, screen resolutions are becoming larger [92], meaning graphics designed for small screens will not scale well. For this scenario, designing vector graphics for the application will help reduce the impact of increasing resolutions since they scale to any size.

Standardisation

A lot of mobile app design is centered around a set of standard elements. This provides performance improvements as well as an optimised user experience. To provide comfort to the audience and make the platform easy to use and learn, custom elements should be kept to a minimum. Since the bulk of *Skouted* users will be familiar with social network applications, using similar UI elements as these popular applications will bring a level of familiarity to *Skouted* . This is easily achievable by using standardised GUI templates and packages with pre-built elements that can be plugged into a page and will scale across screen sizes.

7.3.3 Design Principles

Given the above issues, the design of the app should follow certain design principles which when followed should produce an app that is attractive to the *Skouted* audience.

Learnability

The system should be predictable, providing some response to every action performed by the user. The responses should also be familiar to the user and match their expectations; this can be achieved by using common UI elements in the design and by using elements which are compatible with prior knowledge such as metaphors [93] (for example using a speech bubble icon to represent a button to the messaging page, when a user taps the button they would expect to go to a page related to talking).

Flexibility

Flexibility enables expert users to speed up tasks they need to complete [94] as well as:

1. Allowing customisability to different user needs
2. Giving users control of the flow, permitting them to abandon, suspend and resume tasks without loss of work
3. Allowing multiple tasks to execute at once where possible to allow a user to be efficient

Robustness

The application should be robust, preventing users from making errors and allowing them to recover when an error is made. This can be achieved by providing meaningful error messages to help them understand the issue and prevent them from making that same error again [93].

Users should be able to determine where they are in the application, and discover how to get to another location through actions that can be performed on the current screen.

The application should be responsive with feedback commencing with the start of an action with time affordances when delays are unpredictable.

Time affordances

Users require reassurance that an operation will complete. Time affordances provide an indication of whether everything is ok or if something has gone wrong and help prevent a user from halting an action before it has chance to complete. If possible, time affordances should show the status of the operation so a user can estimate the amount of time left and provide a ‘heartbeat’ so they can

quickly observe if the task is still alive [95]. A simple approach to provide time affordances when loading data from a server is by showing a spinning arrow as in most mobile applications.

Evaluation Through Guidelines

The above principles can be evaluated without much time or resource investment by applying a set of explicit guidelines. For example, Nielsons 10 Usability Heuristics [96] provide a set of guidelines which can easily be followed and checked off for each page. The 10 guidelines are as follows:

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognise, diagnose, and recover from errors
10. Help and documentation

These guidelines should be applied throughout the design process to ensure an app with high usability is created.

7.4 Assumptions, Dependencies and Constraints

There are a number of assumptions that will be made in order to guide the design process, along with dependencies and constraints limiting what can be done.

7.4.1 Assumptions

1. Users will be aged 13 or over

2. Users will have had experience with mobile applications and have interacted with common user interface elements
3. Accessibility features for disabled persons are mostly built into mobile devices and therefore the app only needs to be designed to be compatible with these features
4. People wishing to use the application have an iOS or Android device
5. Users have a sufficient roaming data allowance for allowing infinite loading of information from a server

7.4.2 Dependencies

1. To design the application efficiently, a set of pre-built GUI elements is required
2. Tools must be available which allow quick testing and viewing of the application in a mobile setting
3. Mock data must be available to test the app in a realistic setting

All of these dependencies must be satisfied before development of the user interfaces can begin.

7.4.3 Constraints

1. There is a limited amount of time to complete the application, this may hinder the progress made in designing the user interfaces
2. The team only has limited monetary resources, this limits the tools available for development of the user interfaces and also the evaluation process
3. Tools being used for design will be new to some members of the team, introducing a learning curve which will lead to slow progress initially

7.5 UI Designs

The following subsection describes the different User Interfaces in the *Skouted* mobile application.

Sign Up Process UIs

The first screen the user encounters upon initiating *Skouted* is the Login screen. This screen is considered the application entry point. This is shown in Figure 7.2 below.

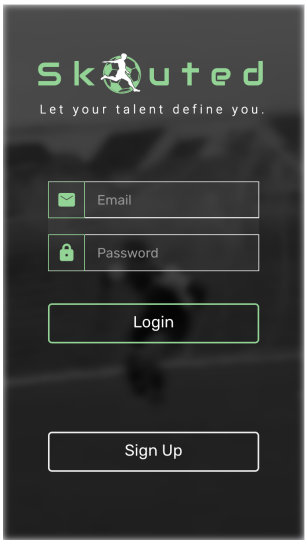


Figure 7.2: Login Screen

If the user is not logged in or does not yet have an account, they will be presented with the login screen. Otherwise, they will be directed to their feed displayed in Figure 7.6. As shown in Figure 7.2, the only information required to login is the user's email address and password given during the sign up process. In the case the user does not have an account and wishes to sign up, they will be taken through the sign up process starting in Figure 7.3.

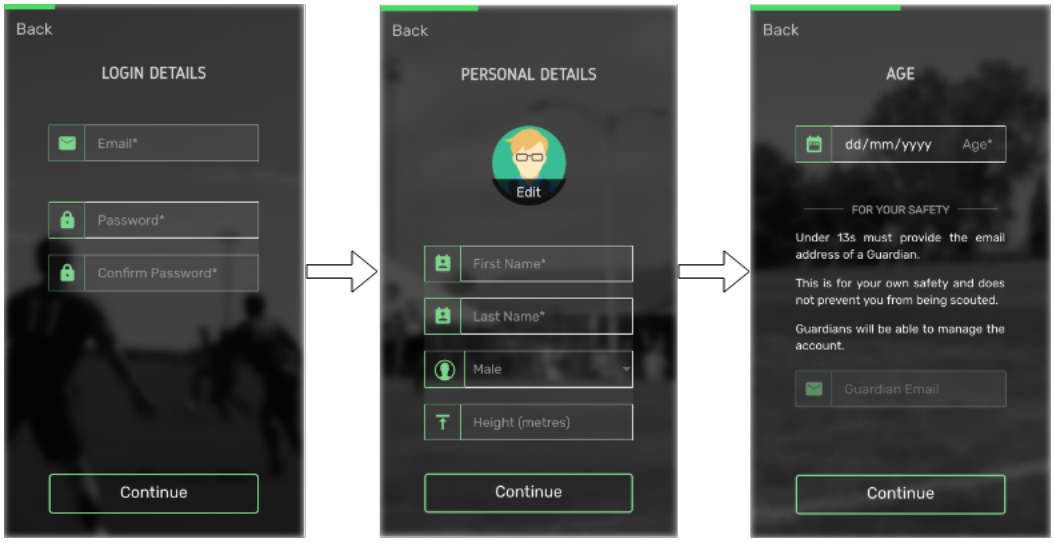


Figure 7.3: Sign Up Process Part 1

The first screen in Figure 7.3 asks for a player's login details, namely their email and password, which as explained above are both required to login. The system verifies that the password is confirmed correctly to avoid input errors. Moreover, the progressive green bar at the top of the screen and the back button together exemplify the aforementioned concept of "robustness" in the human-computer interaction domain. The top-green bar always allows the user to know where in the sign-up process they are, whereas the back button allows them to return to previous sections, therefore forming an example of robustness.

The second screen in the process asks for a player's personal details, specifically their first and last name, their gender and their height. All these details are core player characteristics required to construct their profile which is then accessible by scouts.

The third screen focuses on asking the player's age, paying special attention to under 13 (U13) players. Any child in the U13 age range must also provide their guardian's email address in order to proceed through the sign up process, in order for their guardian to have full management access to the account. Furthermore, this prevents any possible inappropriate behaviour from scouts, and enables the future implementation of the monitoring of all player-scout communications by the player's guardians.

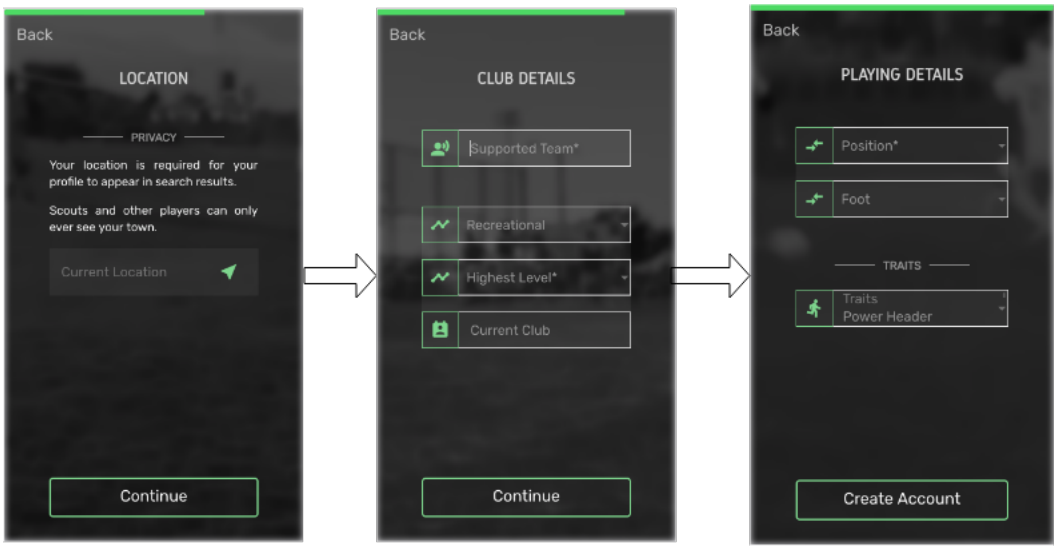


Figure 7.4: Sign Up Process Part 2

The sign up process continues in Figure 7.4, continuing with a screen that requires the user to enter their preferred location. This shall typically be the town of the club which they play for in order for scouts to be able to perform a location-based search of nearby players, a feature deemed useful by scouts according to the customer's domain expertise.

In the future, if the application is made available internationally, the location page will be required before the person's age. In doing this, the correct minimum age of consent restriction will be enforced in accordance with the GDPR[1].

The user is then directed to a screen where they can enter their "Club Details", starting with their "Supported Team" which refers to the team they support. The user can then enter the current level of the league in which they play, and the highest level at which they have ever played, in order for scouts to gain an understanding of the player's experience and expertise. Finally, the user is asked for the current club they play for, again a key element in scouts' search criteria. The user also has the option to specify "Free Agent", meaning that they are currently not playing for any club.

The final sign up screen asks the user to provide details about their playing characteristics. Namely, it asks the user to specify the position they play at, as well as their preferred foot, where they can also specify "Both" if they consider themselves equally talented with both feet. Finally, the user

[1]See Section 3.1.2

may select up to three traits that they believe accurately describe themselves, with examples being "Power-Header", "Speedster" or "Power-Shot".

In addition to the illustrations in Figures 7.3 and 7.4, the sign up process UIs feature a blurred background video of a youth football match, which was agreed after several iterations of the interface with the customer. The final UIs were considered to strike an appropriate line between a modern and professional look.

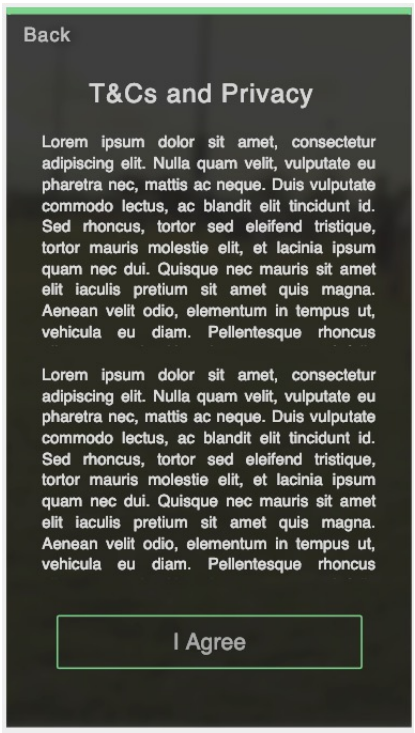


Figure 7.5: Sign Up process: Terms and Conditions

Upon completion of the data collection process, the user will be redirected to a Terms and Conditions page. This page will outline the what content and behaviours are expected from them as users of the *Skouted* application and what constitutes misconduct or abuse of the platform. The terms and conditions will also feature more general details such as user rights and opt-in, opt out procedures. The presentation of this information should be dependent on the age the user identifies as. For younger users, the information contained in the T&Cs will be understandable and meaningful to their age bracket and may be illustrated through diagrams or videos.

7.5.1 Feed UIs

Once logged in, the user's feed becomes their home page, as illustrated in Figure 7.6.

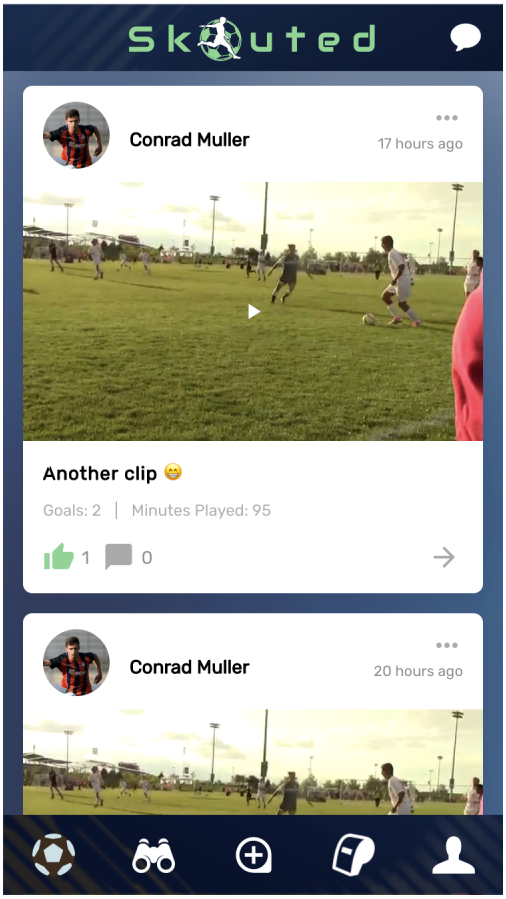


Figure 7.6: Main Feed

The feeds shows posts from players that the user is following, conveying only the most relevant information from the post: namely the caption, the match statistics, the number of likes, and the number of comments. To obtain more information on a particular post, the user can click on the arrow in the bottom-right corner of the post, which will expand into a "Clip-Viewing" page. This part of the interface applies the described principle of learnability, as an "arrow" is a common UI element recognised by users as expanding or continuing existing content. The flow for this action is shown in Figure 7.7.

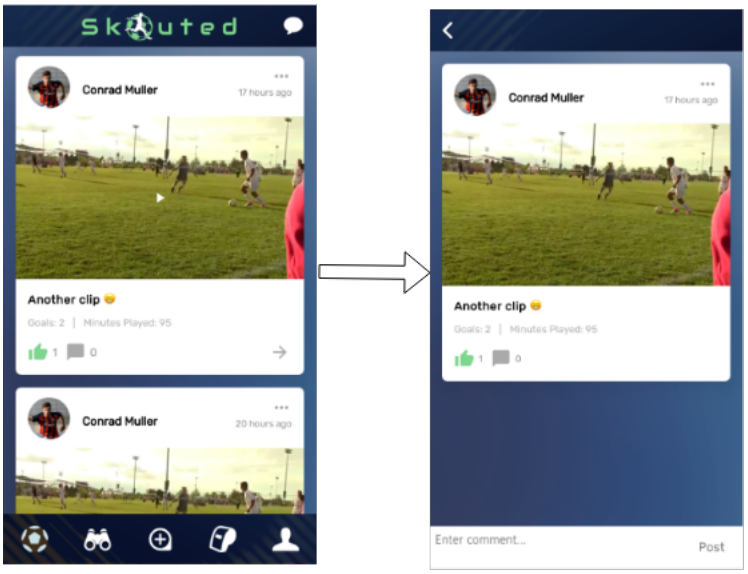


Figure 7.7: Expanding Post on Feed

The clip-viewing page contains any comments the post might have, though in this case we can see that the post does not in fact contain comments. If the currently logged in user wishes to enter a comment, they can simply do so through clicking on the bottom bar and pressing “Post” on the bottom right corner when they have finished typing the message.

7.5.2 Profile UI

Figure 7.8 below presents the user interface for a player’s profile.

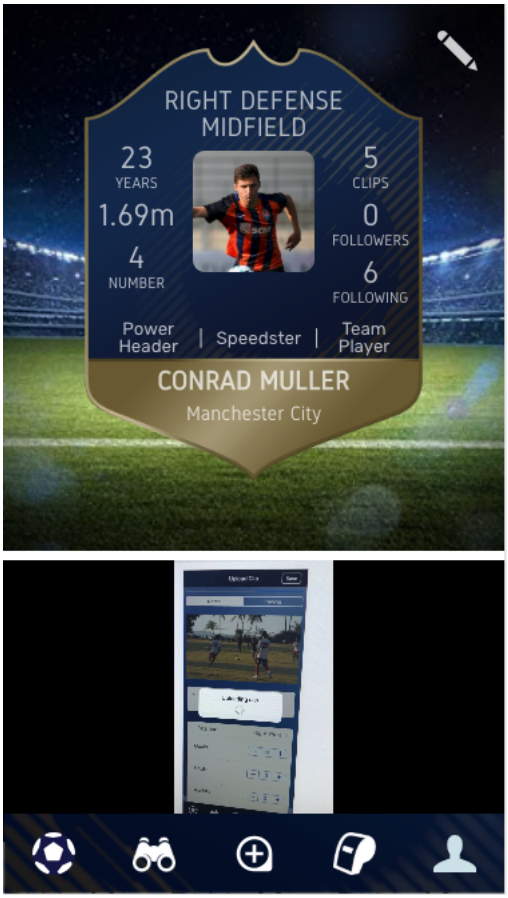


Figure 7.8: Player Profile UI

The central component of the profile is the player's badge, a format agreed with the customer to present all the relevant information. The badge was considered attractive for the *Skouted* target audience, and allowed presenting information in a relevant manner. The top of the badge features the player's position, given great importance by not only its central position but its use of a larger, distinctively spaced-out font. The left column presents the player's core characteristics, namely their age, height and the number with which they play on the field. On the right-hand side of the badge, a column presents information relating to the player's activity on *Skouted*, namely the number of clips they have uploaded, the number of followers, and the number of users they are following. Horizontally, at the bottom of the blue area, the player's 3 selected traits are rendered, with the example of "Conrad Muller" having "Power-Header", "Speedster" and "Team-Player" as traits. Finally, the bottom golden area of the badge presents with a clear, emphatic style the player's name, and the club they currently play for.

As the user scrolls down the page, they will be able to see the central header video, shown in Figure 7.8, and further down the page the thumbnails from all the player's posts will be rendered. Clicking on a thumbnail will direct the user to the previously shown "clip-viewing" page containing that post's information.

7.5.3 Search UI

The application's main search interface is illustrated in Figure 7.9.

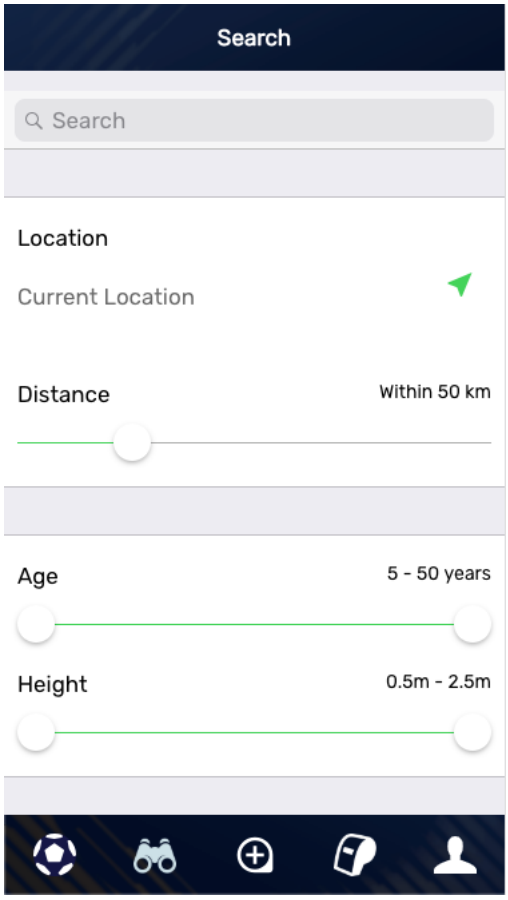


Figure 7.9: Search Page

The first set of filters allows searching for a player by entering a location, and specifying a radius from that location using the distance bar. In the example shown in Figure 7.9, there is a radius of 50km set on the current location of the user performing the search, which would filter all players within that radius.

As the user scrolls down, there are further filters that can be applied, first encountering those

for the players' core characteristics such as their age, height and gender. The final set of filters, where the user scrolls down further, allow filtering the search by highest and current level played, field position, traits and gender. In general, the search has a minimalistic UI with light tones to evoke simplicity and make the filters clearly visible for the user, as these are the main focus of the page.

Additionally, the bottom navigation bar in Figure 7.9 is a clear application of the robustness, learnability and flexibility design principles. Each icon is a metaphor for the different pages, chosen carefully for the user's recognition, thereby applying the learnability principle. For instance, the currently highlighted binoculars mirror a scout's search tool. As well as this, highlighting the icon which represents the current page the user is in is another example of robust design, as the user will always recognise which page they are on. Finally, the design can also said to be flexible since the navigation bar gives the user control of the flow at any point in the application.

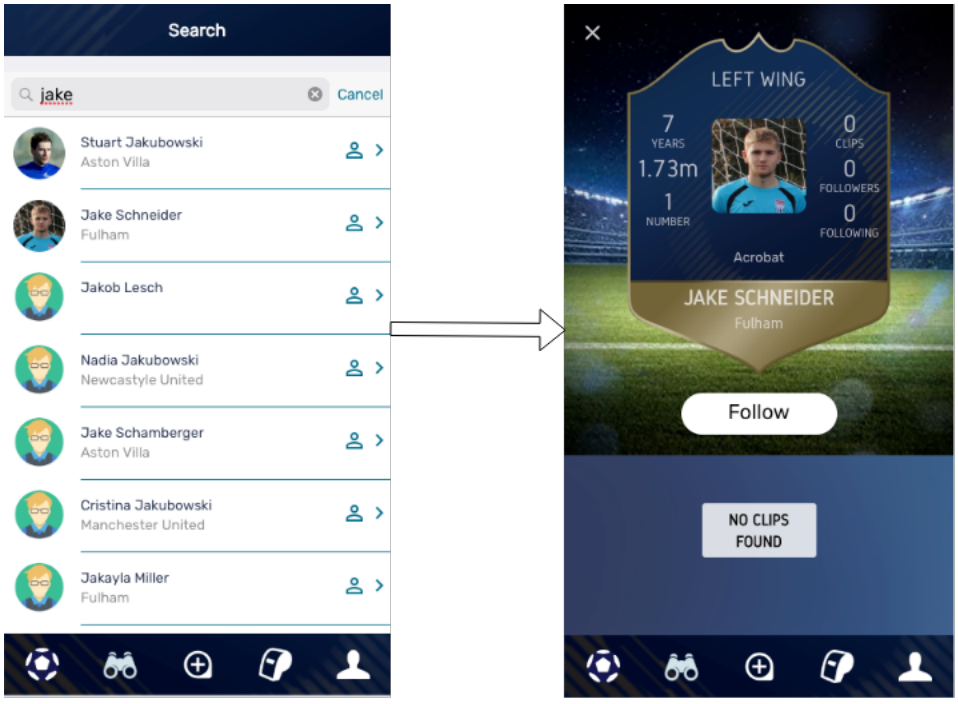


Figure 7.10: Search Flow

When the user types in a player's name, the search is also filtered further by the name, rendering a filtered list of results as that shown in Figure 7.10. By clicking on one of the results, the user will be taken to that specific player's profile page, where they will be able to investigate the player and follow them to track their performance if they have an interest. The example in Figure 7.10

illustrates how typing in “Jake” renders a list of results, where the user selected player “Jake Schneider”.

7.5.4 Messaging UIs

The user’s *Skouted* message hub is presented in Figure 7.11, hosting several features to interact with other users. The message hub can be accessed through the feed’s bubble icon, shown in Figure 7.6, being another example of a metaphor and therefore applying the learnable design principle. Users typically associate a speech bubble to a messaging hub or action, therefore being a suitable icon for easy recognition.

Figure 7.11: Message Hub

The top search bar allows searching and filtering by name through the different users with whom the current user has initiated conversations, which in the example in Figure 7.11 would simply be “Pierre”, a scout, and “Lynn”, another player. Clicking on a user in the list, for instance on “Lynn”,

will take the user to the corresponding conversation, an action flow shown in Figure 7.12.

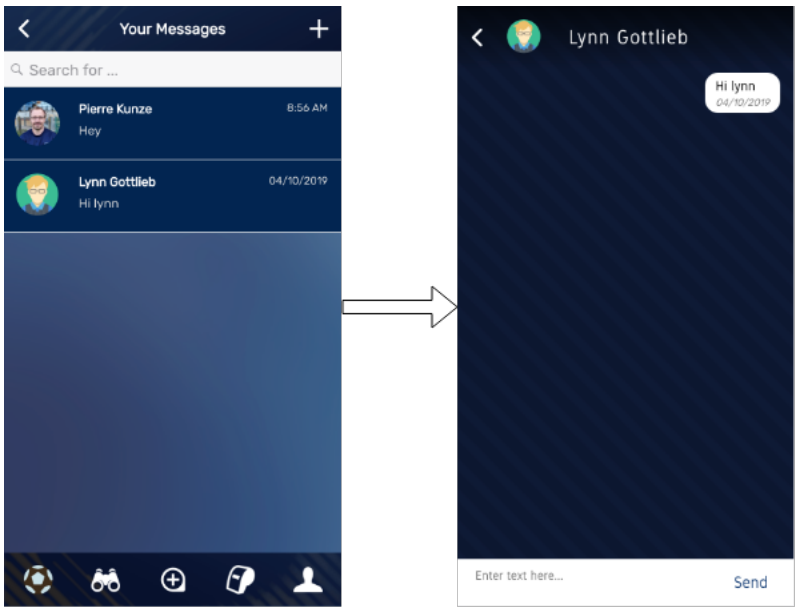


Figure 7.12: Action flow to continue an existing conversation

The top-right corner of the page features a plus button, used to create a new message. When clicking on the button, the user is presented with a screen with an empty search bar, which can be used to type in the name of the user they wish to message. The example in Figure 7.13 illustrates a user typing in the prefix “Ja”, which renders a corresponding list of results. Clicking on any one of the results will take the user to the corresponding messaging page shown in Figure 7.12, in order to initiate the conversation.

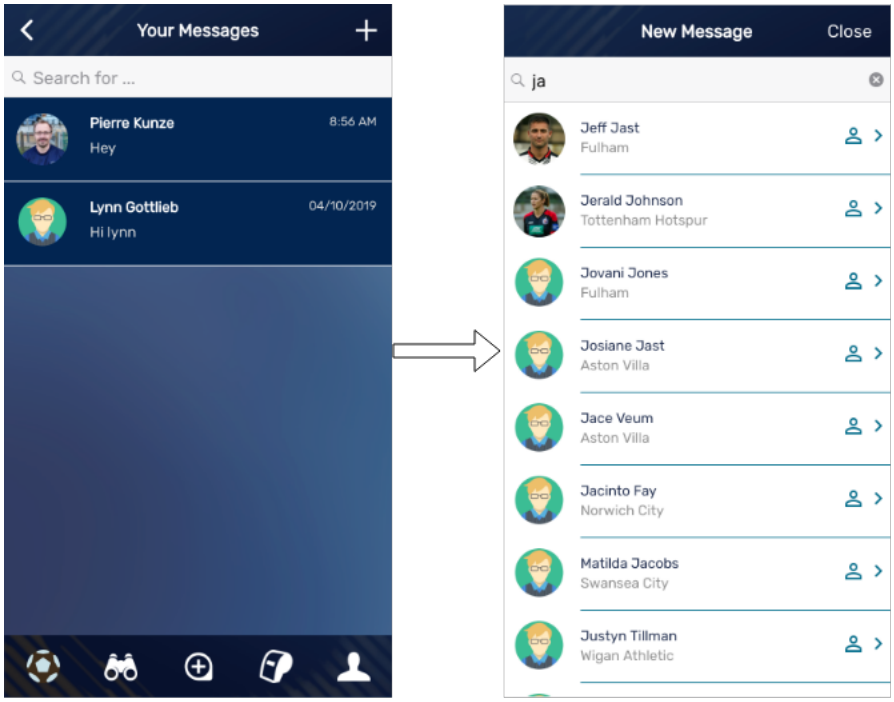


Figure 7.13: Action flow to create a new message

7.5.5 Upload Post UIs

Skouted allows users to create posts from either official matches or training sessions through the interface illustrated in Figure 7.14. Posts form a central part of the *Skouted* social network, appearing throughout user's feeds from players they follow, as well as in their own profiles. For this reason, the user interface to upload a post was designed with maximum simplicity, so that all relevant information to be entered by the user is clear.

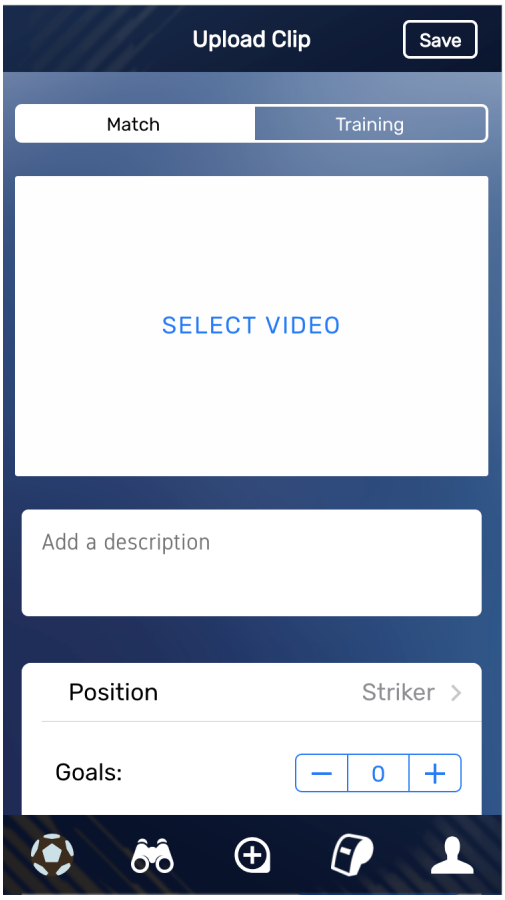


Figure 7.14: Upload Post UI Part 1

The user can use the top tabs to select between a match or training post, with Figure 7.14 showing the interface for the former. The "Select Video" area below is clickable, and when clicked will prompt the user for the associated video they wish to upload from their filesystem. During the process, the user will also be prompted to select a specific frame from the video to use as a thumbnail, later presented in posts throughout the app as seen in the feed UI section. The key match information and statistics can then be entered by the user, such as the position they played in during the match, the number of goals they scored, number of assists, and other relevant information conveyed in Figures 7.14 and 7.15.

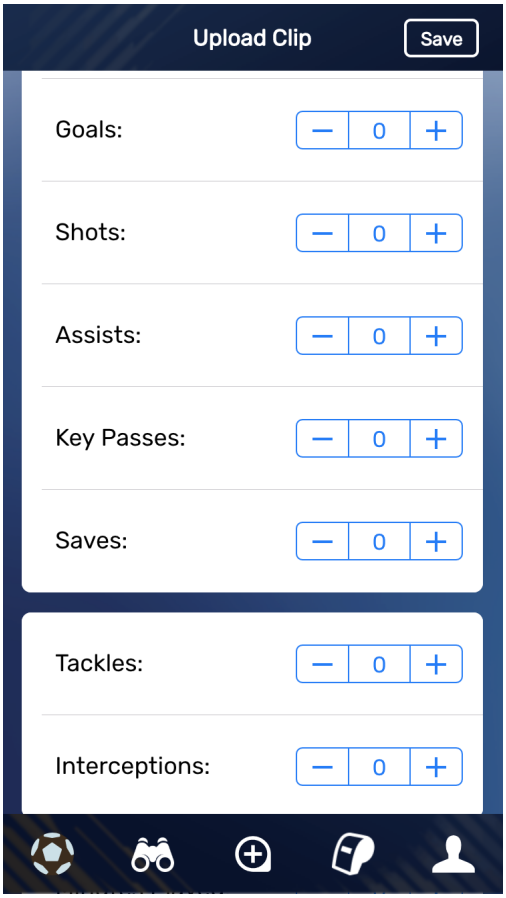


Figure 7.15: Upload Post UI Part 2

7.5.6 UI Iterations

As aforementioned, obtaining final versions of each of the page's user interfaces required actively collaborating with the customer to iterate over initial mock-ups and wireframes. For instance, two illustrative examples of early wireframes are those shown in Figures 7.16 and 7.17.

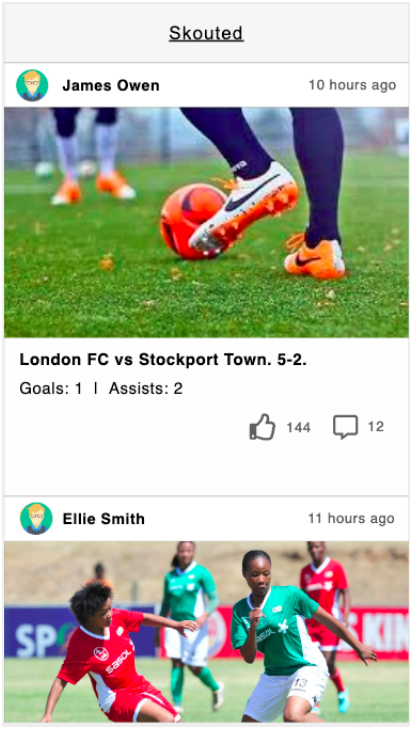


Figure 7.16: Initial Feed Wireframe

The wireframes convey an initial use of lighter tones, with an interface layout and structure similar to that used by other social networks. Using these as starting points, the customer suggested the use of darker colour tones for backgrounds, combined with lighter white-coloured fonts for the application's text. Further, the customer suggested the use of gradient tones as opposed to solid colours for any backgrounds, as well as having a carefully designed navigation bar with appropriate backgrounds. The emphasis on a consistent colour-scheme and application theme from the customer was considerable, as well as the use of football-related icons as metaphors throughout the application.

Through considering the above and continuously interacting with the customer, the team iterated over the initial wireframes over the course of the project to produce the final UIs with a common scheme shown in the UI designs section.

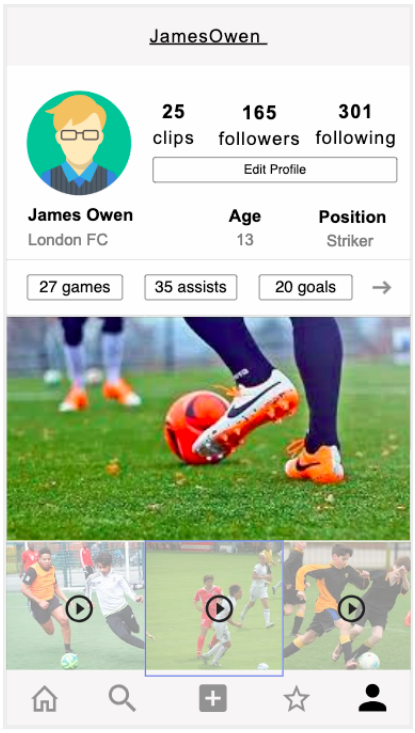


Figure 7.17: Initial Profile Wireframe

7.5.7 Sequence Diagrams

To further explain how a user interacts with the application, in addition to how each screen interacts with each other, sequence diagrams have been created for the following scenarios:

- Entering the feed page, seen in Figure 7.18
- Clicking on a video in both feed and profile page, seen in Figures 7.18 and 7.19, respectively
- Entering the profile page, seen in Figure 7.19
- Entering the edit profile page, seen in Figure 7.19
- Signing up and signing in to the application, seen in Figure 7.20
- Adding a video to the application, seen in Figure 7.21

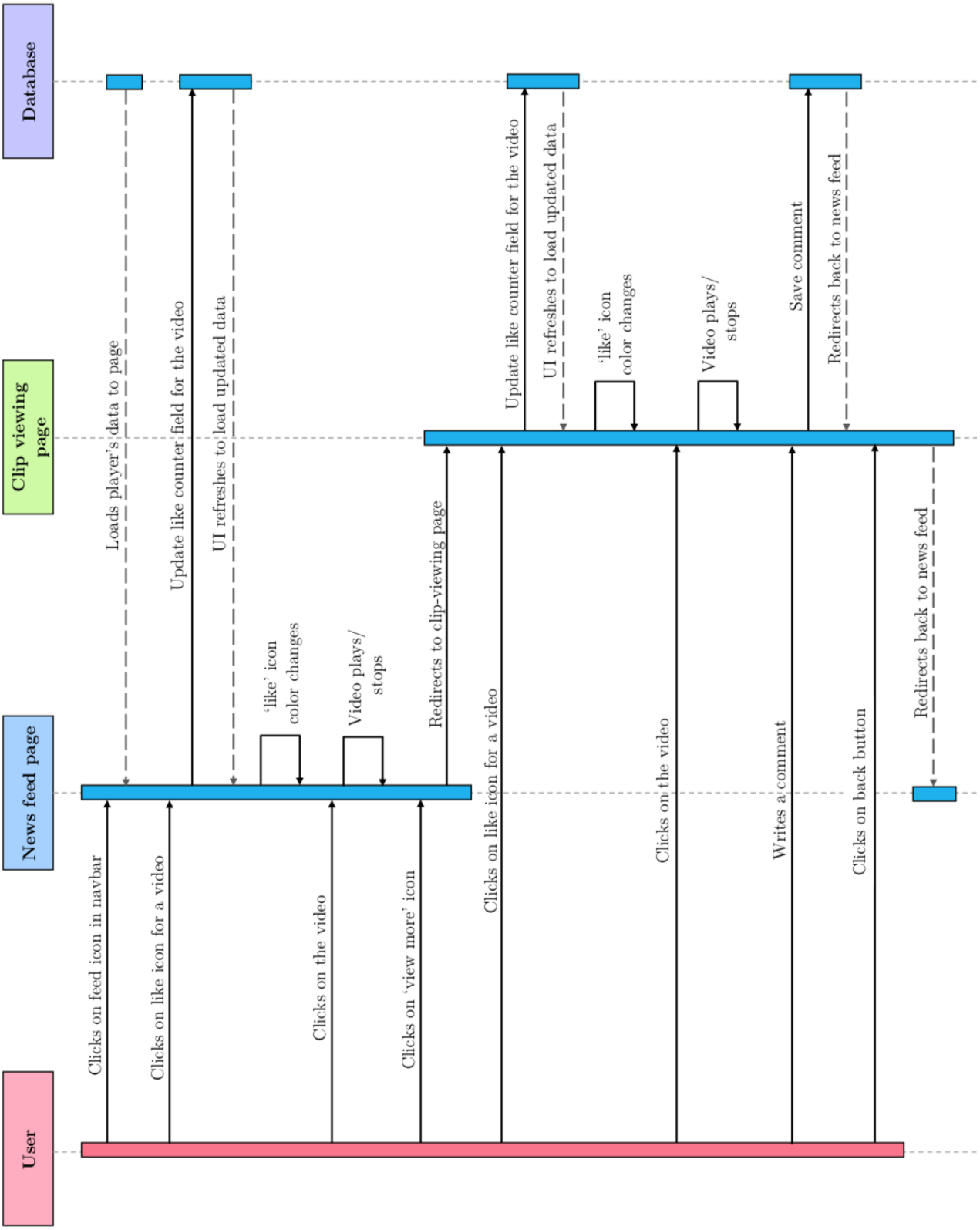


Figure 7.18: Sequence Diagram showing design of News Feed

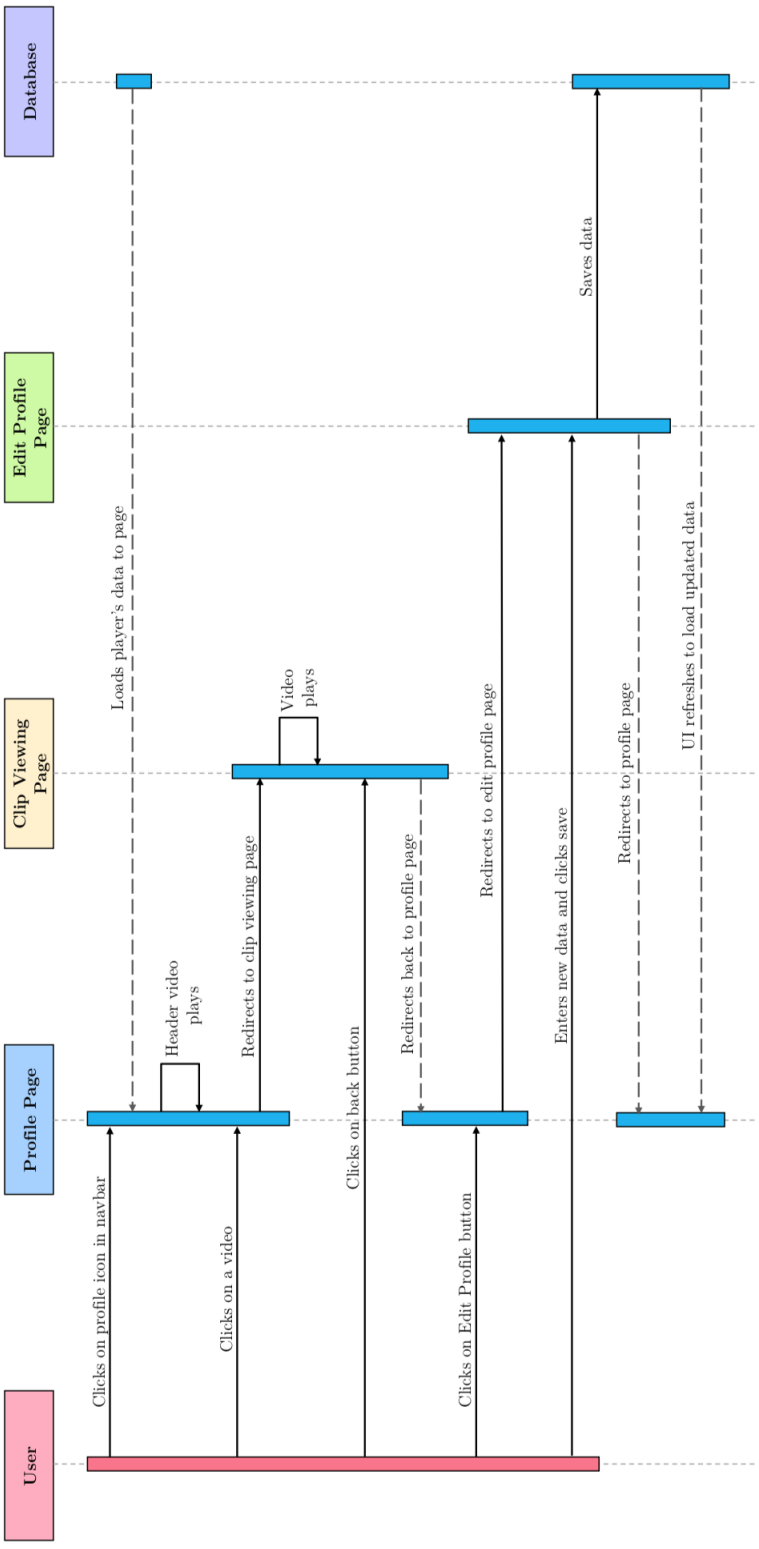


Figure 7.19: Sequence Diagram showing design of Profile Page

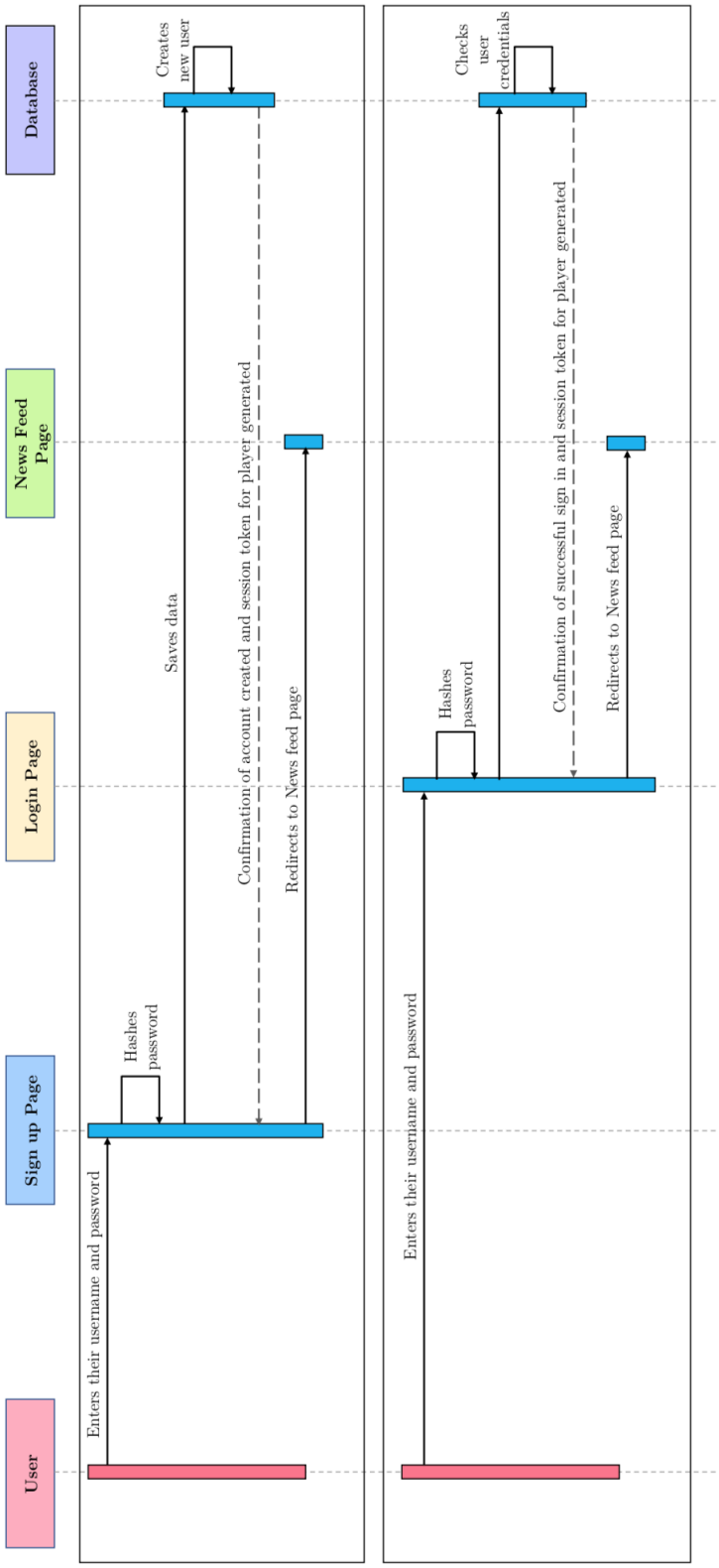


Figure 7.20: Sequence Diagram showing design of Sign Up and Login Page

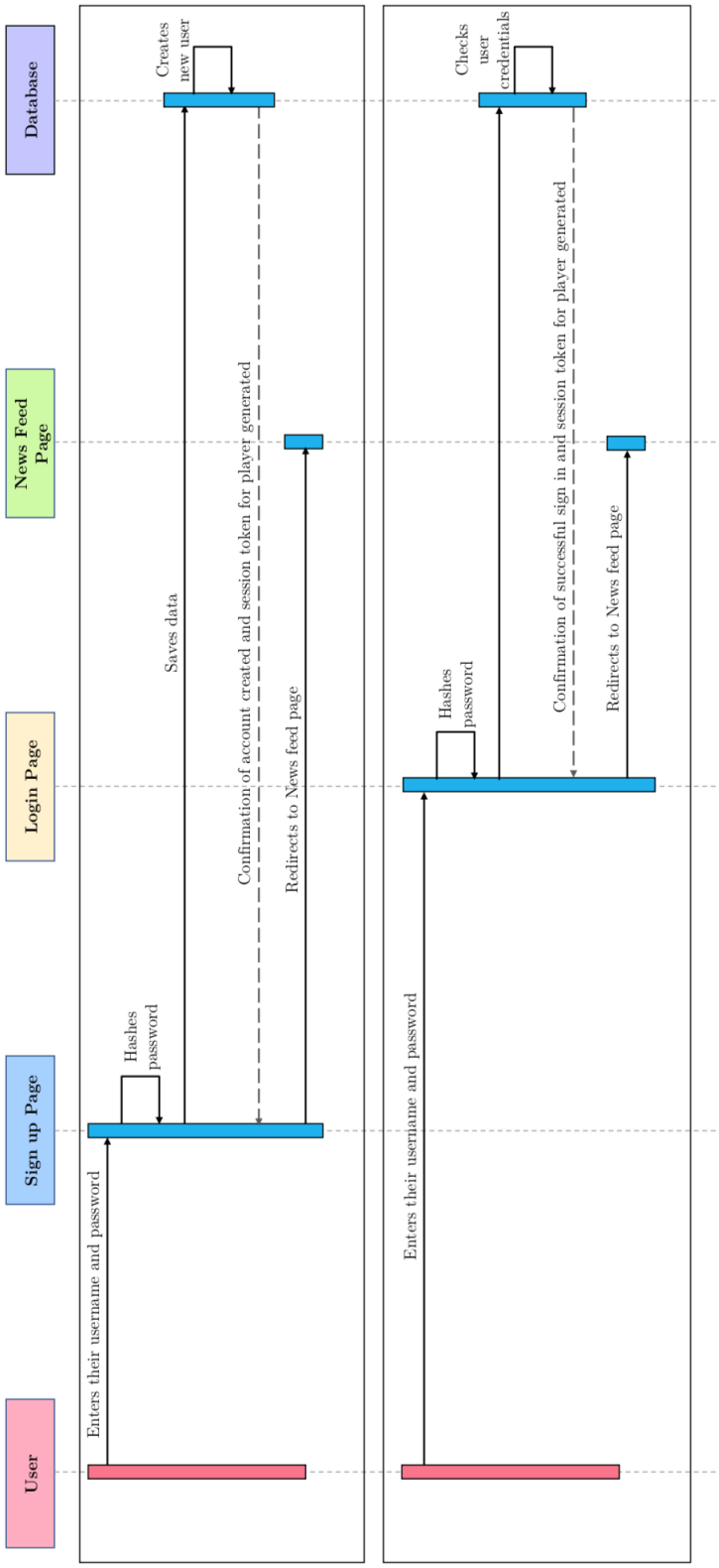


Figure 7.21: Sequence Diagram showing design of Upload Post Page

Chapter 8

Implementation

This section discusses the implementation decisions of the system, describing the mechanisms and technologies involved in implemented functions on both the front-end and the back-end, and also the configuration of the designed system infrastructure.

8.1 Implementation Overview

The system is divided into microservices, which, as per the separation of concern philosophy, define its distinct functional elements:

- mobile
- profiles
- media
- posts
- search
- messaging
- feed

These microservices are built and executed as Docker containers, supplied with environment variables and ports to configure each service. The *docker-compose* file defines these services, the ports

they run on, the mount paths of any external files, and the environment variables supplied to the service, such as authentication credentials and the URLs of other services.

```
media:
  build:
    context: media
    dockerfile: ./Dockerfile
  tty: true
  ports:
    - ${MEDIA_PORT}:${MEDIA_PORT}
  volumes:
    - ./media:${APP_MOUNT_PATH}
    - ${CREDENTIALS_LOCAL_PATH}:${CREDENTIALS_MOUNT_PATH}
  environment:
    - NODE_ENV=${NODE_ENV}
    - PORT=${MEDIA_PORT}
    - GOOGLE_APPLICATION_CREDENTIALS=${CREDENTIALS_MOUNT_PATH}
```

Figure 8.1: The definition of the *media* microservice in the *docker-compose* file

By specifying the URLs of other services as environment variables, the microservices are able to interact with each other by accessing the supplied API endpoints, which allows the system to function as a singlularly deployed system, despite being loose-coupled.

8.2 Front-End

The *mobile* microservice controls all the user's interactions with the *Skouted* app, consisting of all the front-end elements and controller libraries which integrate the front-end with the back-end. The front-end of the system is implemented using the React framework alongside Framework7, which is designed for the rapid creation of user interfaces for mobile devices.

Entry Point

The *App* root component defines the structure of the app, and is the entry point into entire front-end of the *Skouted* app. Having a single point of entry to the application decreases the complexity of the system, with all the system information being propagated down from a single source of truth. The app is initialised with the *appParams* object which define the name, theme, and routes within the app. The *routes* parameter defines all the URL paths of the app and the corresponding components that are loaded when they are accessed.

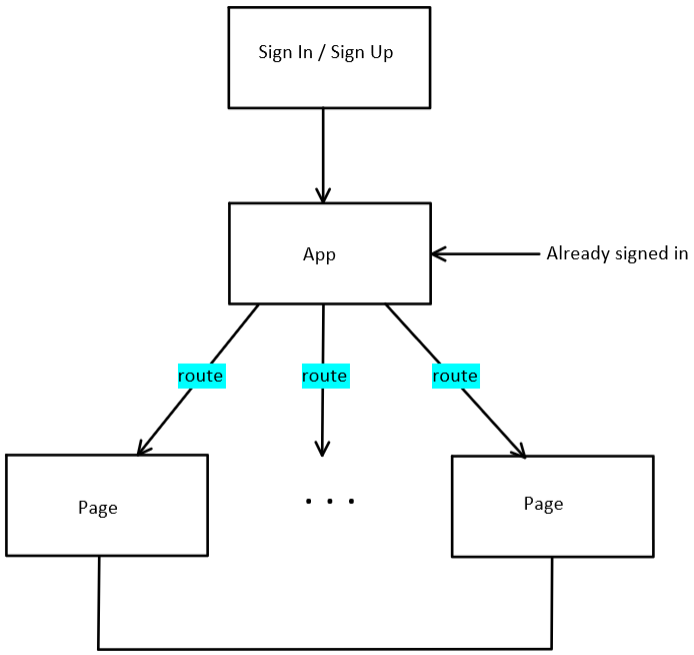


Figure 8.2: Implementation of the front-end entry point into the *Skouted* app

The app is also initialised with the data of the current user, loaded from HTML5 localstorage if the user has logged in, also provided to the *app context* which is universally accessible throughout the app. This allows all other components and pages within the app to know which user is currently logged in to the system, and obtain the user's data for other purposes.

Components

Every visual element in the *Skouted* app is wrapped as a React component. The page components consist of element components to generate the web pages of the *Skouted* app loaded by the app router, and element components such as the navigation bar are singular components that form a single visual element on the page. Each component is accompanied by a CSS definition which contains its styling information, affecting how it is displayed on the page. All CSS styles are prefixed with the root style of the component, to ensure that styling conflicts do not occur in the global namespace.

Library

The *utils.js* file within the *lib* folder contain all the shared functions that are used throughout the app by various components. The *consts.js* library file contains all the app's constants, such as the URL of each page, the app name, and the API endpoints URLs of the back-end microservices.

Controller

The *controller.js* library contains wrapper functions that send API requests to the microservices of the system, allowing the front-end of the system to communicate with the back-end to send and receive data. Functions within the controller call upon the various API endpoints defined by the back-end of the system by sending HTTP requests of different types using the *Fetch API*. The response object received by the server is then unwrapped using the user-defined *unwrapResponse* method which returns an JSON object that unpacks the success or error message returned, and the payload, which can be rendered by the front-end.

8.3 Back-End

The back-end of the system makes use of Node.js, which is a JavaScript runtime built around the Chrome V8 Engine [97]. All the application endpoints are initialised by each microservice's Node.js Express server.

8.3.1 Endpoints

The HTTP endpoints in the system are handled by the *Express* library. For each possible HTTP request, an endpoint is registed to handle how the server responds to the request sent by a user. Each microservice contains a number of endpoint definitions which are specific to their functionality.

GET

The endpoints that handle GET requests are the simplest, as they only require fetching data, typically from a Google Firestore collection. They all have the same format, returning an object or an array of objects.

PATCH

The endpoints handling PATCH requests are created to modify existing resources within the system. This typically involves updating entries in the Google Firestore NoSQL database. Their operations vary greatly, and requests may or may not return an object depending on the desired effect.

POST

POST request endpoints handle the creation of new resources in the system. They handle the submission of form data, and convert the submitted data into a format that is stored in the Firestore database. They all require the user ID of the current user to be sent as a parameter as well as any information to be stored to have the data stored in the correct collection. They may or may not return an object depending on the function.

Status Codes

There are a number of status codes used consistently throughout the system to inform the program of the results of a response sent by the server. The "2xx" status codes (e.g. 200) indicate the response to a request is successful, whilst "4xx" and "5xx" status codes mean that the server failed to give the correct response to the user. A "4xx" error indicates that there is a problem with an input the user has given to the endpoint (error 400), or they have attempted to access an invalid or unauthorised resource (error 404), and a "5xx" error (e.g. 500) indicates that there is a problem with the microservice server.

8.3.2 Media

The *media* microservice is responsible for uploading media to the Google Cloud Platform. All media files are stored in a single Google Cloud bucket with unique identifiers for object names.

Uploading Media

When the user first uploads an image or video, a number of checks are made before the media file is processed and stored. The following validation rules are applied to uploaded files:

1. Uploaded file is successfully sent to the server
2. File format of uploaded file is correct based on the mimetype and extension of the file

3. Maximum quality of a video is equal to or less than 1080p
4. Maximum width or height of an image is less than or equal to 3000 pixels
5. Dimensions of an image are square
6. Video length is less than 6 minutes long

There are two separate endpoints for handling videos and images to execute the different checks, resulting in two API defintions with differing functionality, to reduce the overall complexity of the media microservice.

After completing the validation stage, the media file is saved locally onto the server with a unique name generated using the *uuid2* package provided by the npm package manager. This package generates a *Universal Unique Identifier* which has an incredibly low collision rate. Collisions are identified by checking if a file with the generated unique identifier exists on the Google Cloud Platform, restarting the name generation process, if a collision has occurred.

Finally, the local file is uploaded to the Google Cloud bucket, automatically replicated by the Cloud Storage service. A URL is returned by the endpoint, pointing to the location of the uploaded file in the cloud bucket, to be returned by the endpoint. The temporary file is then deleted from the server asynchronously and the process is completed. Failure during the validation stage and the uploading stage results in the termination of the upload, with an error being sent to the user to notify them of the reason for failure.

Generating a thumbnail

When uploading a video, a thumbnail is automatically generated using a timestamp of the video provided in the body of the request. The *node-ffmpeg* package makes use of the *ffmpeg* libraries for manipulating the video to extract a thumbnail image. A JPEG image from the frame of the timestamp is outputted and saved as the thumbnail for the video, also returned by the endpoint.

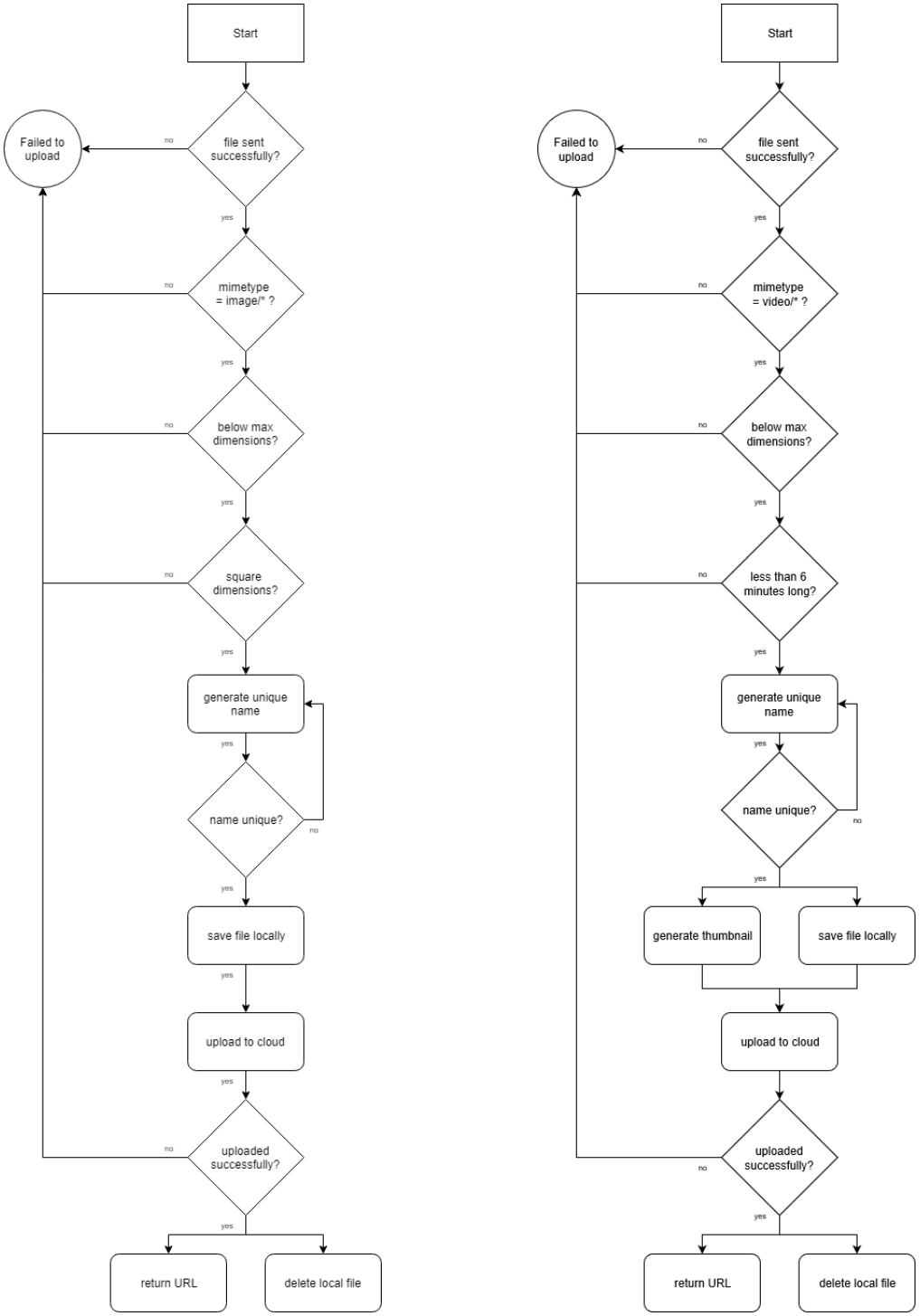


Figure 8.3: Program flow chart for uploading an image (left) and video (right)

8.3.3 Profiles

The *profiles* microservice deals with profile management, handling profile updates, and the retrieval of profile data. As well as this, it also handles the authentication in the system on a per-user basis.

The structure of the profile objects expected in the Google Firestore documents are defined in the *consts.js* library file. The profile attributes are divided into *player* attributes and *scout* attributes, with a set of *base* attributes that are common to both user types. There is also a definition for the modifiable attributes of a user, which is referenced when a request is made to update a player's profile.

Retrieving user data

There are a number of endpoints defined in this microservice to retrieve a user's profile data from their corresponding Firestore document:

- Get followers
- Get followees
- Get following status between two users
- Get profile data
- Get all posts by user

For each of these endpoints an object or value is returned to the client, which can used by the app to achieve a specific action or display some information.

Creating a user profile

The endpoint that handles the creation of a new user profile acquires input from the signup page. Firstly, a user object is generated from the passed parameters. A number of validation checks are then made to ensure that the user object is valid. It checks whether a player type first of all exists, and then whether the entries in the object correspond to the specified user type. Finally, it checks if all the user object fields are of the correct type. After passing all these checks, the user object is ready to be stored as a new document in the Firestore *users* collection. A new document is created in the *user* collection, and the data is saved into the new document. *Keycloak* is used to handle

the authentication in the system, managing identities and access to the system, and the new user is created using the *keycloak* client. Finally, the user object is duplicated in the Elasticsearch database to allow the user to be easily searched within the app. These are all completed in parallel as they are independent operations, improving the response time of the server. If any of these operations were to fail, the user creation process will terminate. The user receives a response notifying them that a new user account has been created, once all operations are complete. The flow chart showing this process is shown in Figure 8.4

Updating profile picture

The profile picture is updated by sending a request to the *media* microservice image upload endpoint, where all media processing is carried out. Once the new image URL is received as a response, the old profile picture is deleted from the Google Cloud Storage bucket.

Updating profile information

The endpoint first fetches the type of the user to determine which fields are modifiable. From this, it checks whether the supplied list of entries that have been modified are within the list of modifiable attributes for the given player type. If there exists an entry that is not found in the list then the endpoint responds with an error, else the attributes are modified in the corresponding Firestore document. The new user object is then updated in the Elasticsearch index to ensure that search results are up-to-date.

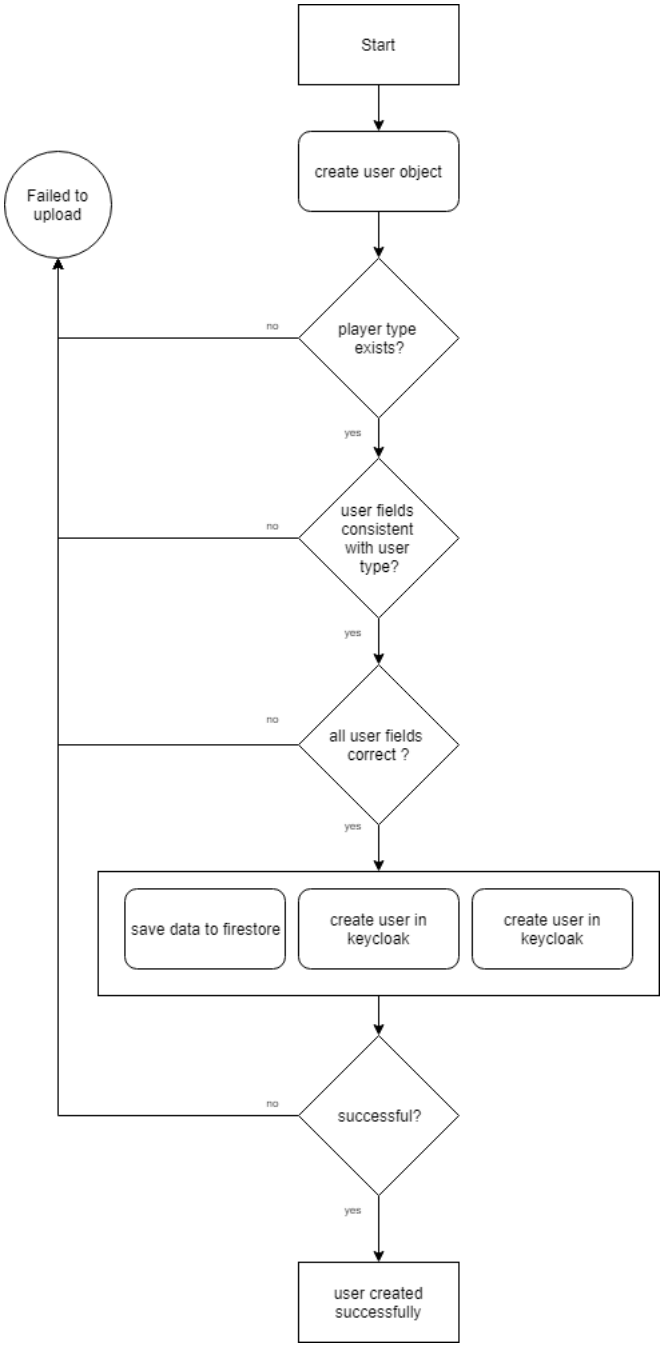


Figure 8.4: Program flow chart for creating a user

Authentication

The endpoint obtains the email and password from the user, and a JSON web token object is generated using the *Keycloak* auth client. This token object can only be successfully obtained if the credentials supplied match the user entry in the *Keycloak* client. The token object or "token

set" contains an access and refresh token which is used to manage the session of the user. Once the token set is obtained successfully, it is sent in the response of the request to the user. Using the refresh token, the user is able to obtain a new access token when accessing a protected resource or carrying out an action requiring authentication such as commenting on a post or updating profile information. Access tokens are set to expire every minute, mitigating the risk any potential account breaches, as the refresh token can be revoked. With the access token expiring, a hacker cannot refresh the this token, and their window of opportunity is limited. To authenticate with other microservices, the client must attach their access token to the header of every request made.

8.3.4 Posts

The *posts* microservice handles the creation, deletion, modification and retrieval of post information from the post collection in Cloud Firestore. This also includes managing comments, match statistics and post liking/unliking.

Post Structure

The post document structure is shown below

```
{
   userId: String,
   caption: String,
   type: String,
   clipUrl: String,
   thumbUrl: String,
   createdAt: TimestampStr,
   commentCount: Number,
   likeCount: Number,
   matchDetails: Object,
   likes: subcollection
   comments: subcollection
}
```

The *TimestampStr* type is a Date format specifically used by Firestore to encode seconds and milliseconds, and any *Date* objects are converted into this type before being stored. The match

details are stored in an object using key-value pairs to more easily accommodate potential additions of match statistics without having to change the structure of the system. The likes and comments are both subcollectons within the post document, with a post's likes and comments data stored as documents in these subcollections, rather than within the object itself. A likes counter and comments counter are stored to reduce the number of calculations carried out in the system; the count can be directly read from the post document rather than having to fetch and enumerate the total number of documents in each sub-collection on every request.

Creating a post

First of all, the endpoint calls upon the *media* API to upload the video data passed through the body of the POST request by the user. If the video fails to upload then the post creation process is terminated with an error. If it is successful, the endpoint then checks if the user ID of the user sending the request is valid. Once the user has been verified, a user document is formulated and uploaded as a new document to the post collection. This document is also duplicated in the user collection within the sub-collection of posts by the user for the purpose of updating their posts in their profile page. Finally, the post count of the user is incremented. The new post document is pushed out across the system to their followers' feeds.

Liking and Unliking

When a post is liked, a new empty document is created in the likes subcollection within the post document. The like document is uniquely identifiable by the ID which is the user ID of the user who likes the post. The like count is incremented once the post is liked. To unlike, the document is simply deleted from the likes subcollection and the like count is decremented.

Commenting

To create a new comment under a post, a new comment document is generated in the comments subcollection under the post document. First of all, it checks whether the post ID is valid, then checks if the comment string is not empty. The endpoint then creates the comment object with the structure shown below:

```
{
  commentedAt: TimestampStr,
```

```
  comment: String,
  replyTo: String,
  user: Object,
}
```

The *replyTo* field stores the ID of the comment being replied to (set to *nulll* for a top-level comment). The *user* field contains the user's profile data (avatar, name, etc.). This is for the fan-out system to easily propagate changes made in a user's profile to all comments made by the user in the system.

The endpoint checks whether the provided user ID of the commenting user and the ID of the post are valid. If so, the comment object is saved as a document in the comments sub-collection. The updated post data is then pushed out to the feeds of the followers.

The process for creating new replies to comments is essentially identical, with the only difference being that the *replyTo* field is set to the ID of the comment being replied to instead of being set to *null*.

8.3.5 Feed

The *feed* microservice is responsible for managing the feed data outputted to the user. The endpoint makes use of pagination to display feed items to the user without returning all the items in the user's feed. The endpoint initially loads 50 of the most recent feed elements for a user when the feed page is loaded. As the user scrolls, requests are sent to the user to obtain more feed items, using the timestamp of the last feed item rendered on their feed page. The endpoint then obtains 50 more items that are created earlier than the supplied timestamp and sends them in the response to the user, thereby achieving an infinite scrolling effect.

Explore

The explore feature of the application shows users new recommended posts by other users that they do not already follow. Elasticsearch is currently used to implement the selection of recommended content by taking in a set of parameters which define the scoring system to score posts by relevance to the user. The most highly ranked attribute is the geo-distance of a poster from the user's location, which prioritises players that are closer to the user, followed by the number of followers,

and the time the post was created to give the most popular players' most recent posts that are local to the user. The results are retrieved using pagination in the same manner as the feed is, with the most relevant posts being obtained initially and proceeding results loaded as the user scrolls further down the explore feed. This implementation is simpler than the recommender systems researched for the project, but provide good results to be similarly functional.

8.3.6 Messaging

The *messaging* microservice is responsible for handling messaging between users on the platform.

Structure

A conversation between two users is described as a *thread*. A thread is represented as a document within the *threads* collection under a user document, containing all message information between two users. The ID of a thread is the ID of the other party. The structure of a thread can be seen below:

```
{
  user: Object, ( { name: String, avatar: String } )
  latestSentAt: TimestampStr,
  latestMessage: Object,
  newMessageCount: Number,
  messages: subcollection
}
```

The *latestSentAt* field stores the time of the most recent message in the *messages* sub-collection, for display in the messaging home page, so this data does not need to be retrieved from the sub-collection every time the page is loaded. This also applies for the *latestMessage* field, which stores the data from the latest message in the *messages* sub-collection. The *newMessageCount* field stores the total number of new messages that have not yet been read by the user, and the user object is once again included to be pushed out.

Individual messages are stored as documents in the *messages* sub-collection under a thread with the following structure:

```
{
  sentAt: TimestampStr,
  readAt: TimestampStr,
  message: String,
  sender: String,
}
```

The *sendAt* field stores the timestamp of when the message was sent, whilst the *readAt* field stores the timestamp of when the message was read by the receiving party (initially set to *null* to indicate the other user has not yet read the message). The *message* field stores the message string and the *sender* field stores the ID of the user who sent the message, which is used to differentiate sent and received messages in the conversation page in the application.

Sending a new message

First of all the endpoint checks if the message string is empty. Then the passed IDs of both users are verified to identify if they are valid. The endpoint finally checks if a conversation currently exists between the two users by checking if the thread document exists in their messaging collections. If the threads do not exist then they are instantiated in both their collections before proceeding to add the new message document to the *messages* sub-collection. The *readAt* field in the message document is set to *null* to indicate that the message has not been read by the other user. The thread information for both users is updated accordingly, setting the new latest message and incrementing the unread message count for the receiver.

Retrieving messages from thread

When retrieving messages, it is required to update the corresponding thread document to show new messages have been read. The standard checks are made initially to determine whether the two users conversing are valid users on the system, returning an error if not the case. The endpoint then checks if the thread document for any of the users exists; a non-existent thread document indicates no messages have been sent between the two users, so an empty array is returned in the response to the request. All messages in the messages sub-collection are then retrieved in ascending order of the time sent (so the newest messages appear at the bottom of the page). For each message document, if the *readAt* field is *null* and the user viewing the message is not the sender then the message is

marked as read with the timestamp, and the number of unread messages is decremented. The array of messages is finally returned in the response. The flow diagram describing the implementation of this endpoint is shown below in Figure 8.5.

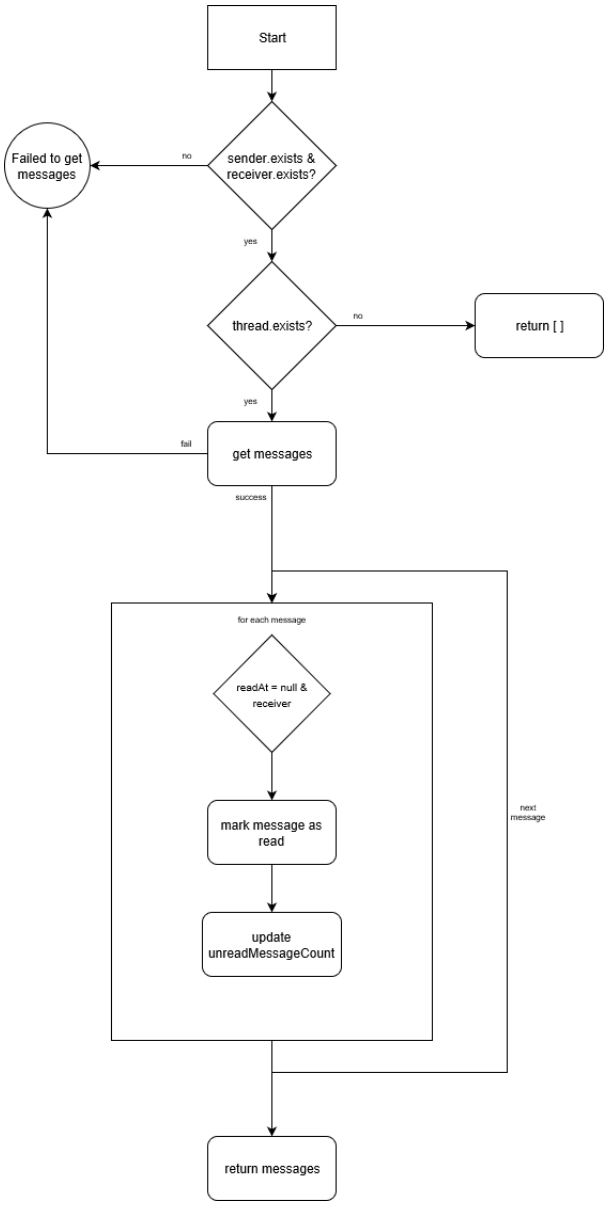


Figure 8.5: Program flow chart for retrieving messages in a thread

Listening for messages

To allow messaging to be carried out in real time, the messaging page needs to be continuously updated to render new messages. This is done by polling a listen endpoint on the messaging

microservice at an interval of 0.5 seconds. The same process for retrieving messages is used, except only messages that are sent later than the last read message are retrieved, returning newly sent messages that have not yet appeared on the messaging page.

8.3.7 Search

The *search* microservice focuses on the search functionalities in the system, which includes searching for players and scouts on the platform to view their profiles or send a message, and the *explore* features of the application to recommend new related posts to users. The microservice makes use of the Elasticsearch database for fast and elaborate querying of data.

Querying users

A series of search parameters are sent in the search request by the user, composing the filters to be used when querying users on the system. An empty query builder is initialised, and the results are filtered by geo-distance if the geolocation data is provided. The additional queries are then appended to the query builder using a reducer by joining the query parameters. The Elasticsearch client is then prompted to search the database with the query builder's query, returning an array of results which is then transformed to be able to be used by the front-end of the system.

The query builder limits the number of returned results to 20 to prevent a large number of results from being returned and affecting the system's performance. Pagination is used instead to obtain multiple results, with the query builder taking in the result index as a parameter to indicate which results onward should be returned by the Elasticsearch client. This index is changed when the user scrolls past 20 results, and a new search request is sent with the new search index to obtain the next 20 results.

Filters

The filters accept requests of the following format at the search endpoint:

```
filterName: { type, param }
```

where *filterName* is the name of the filter (name, location, etc.), the *type* indicating the query operation carried out on the filtered attribute (match, range, term) along with the value to be queried against, and the *param* indicating a sub-filter to be used for the filter specified by *filterName*

(e.g. longitude and latitude for location). Combining multiple query objects allows for very complex queries to be executed by Elasticsearch without extra development complexity.

8.3.8 Data Pushing System

The data pushing system (also known as the fan-out system) is used to propagate changes across the system. Rather than all services obtaining a user's profile information from a single document in the database, the user's information is duplicated across the system to reduce the number of requests sent to Cloud Firestore by a significant amount and reduce any processing required to retrieve a result.

Data Push Object

Documents in the database that are reliant on modifiable profile information (name, avatar, etc.) such as comments, threads and posts maintain a *user* field which stores the necessary profile information in the document. Any changes made to the profile are passed to this user object in the data pushing process.

Followers

For each user that is being followed, the following document must be updated if their name or avatar is modified. The system iterates through the *following* sub-collection, searching for the document with the ID matching that of the user, and its user object is updated. A visual representation of the process can be seen in Figure 8.6

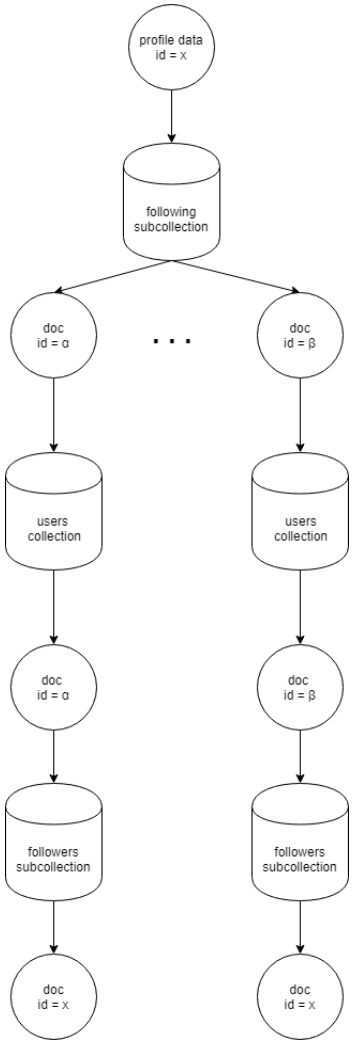


Figure 8.6: Process for updating follower data in the system

Messaging Threads

To update the messaging threads, all the documents in the *threads* sub-collection are iterated through to obtain the users who have been contacted. For each contacted user, the fan-out object in the thread document linked to the user who has updated their profile is updated within their *threads* sub-collection (see Figure 8.7).

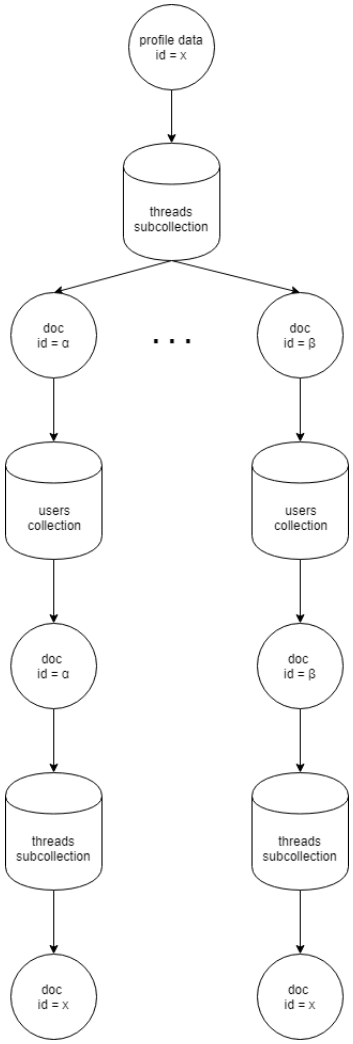


Figure 8.7: Process for updating messaging thread data in the system

Comments

The same approach is used to update the comments made by the user, with all comments in the *comments* sub-collection for the user being iterated through to obtain any affected clip IDs. The comment documents are then updated in each of the *comments* sub-collections of the clips (see Figure 8.8).

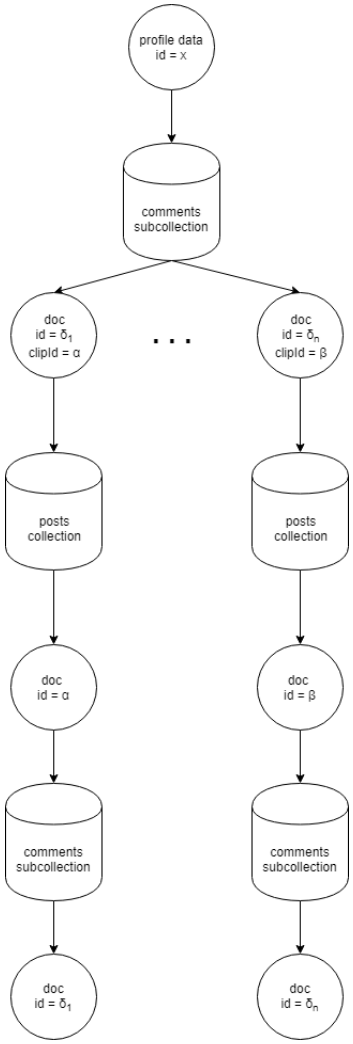


Figure 8.8: Process for updating comment data in the system

Posts

Once again, the same principles are applied to update posts after a user's profile has been modified. All posts submitted by the user are obtained from the *posts* sub-collection for the user, and then updated in the main *posts* collection (see Figure 8.9).

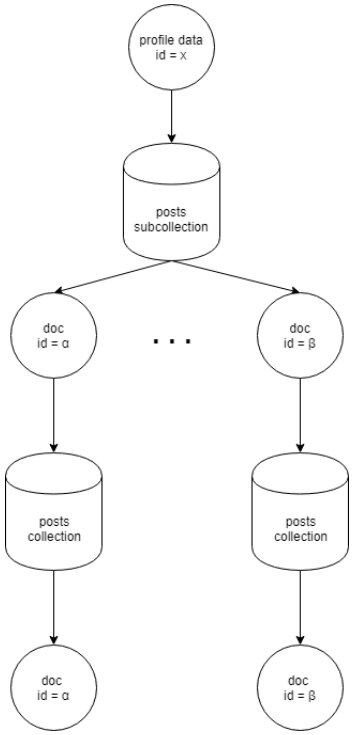


Figure 8.9: Process for updating posts profile data in the system

As well as this, any changes made to the post document must be distributed to all the feeds of users who follow the poster. This process is carried out when a change to the poster's profile information is made, the match details are modified in any way, or a new comment is submitted. It involves first updating the relevant post document within the user's collection of posts, obtaining thr list of followers from the user's *followers* collection, and updating the document in the followers' *feed* sub-collection. This is shown in Figure 8.10.

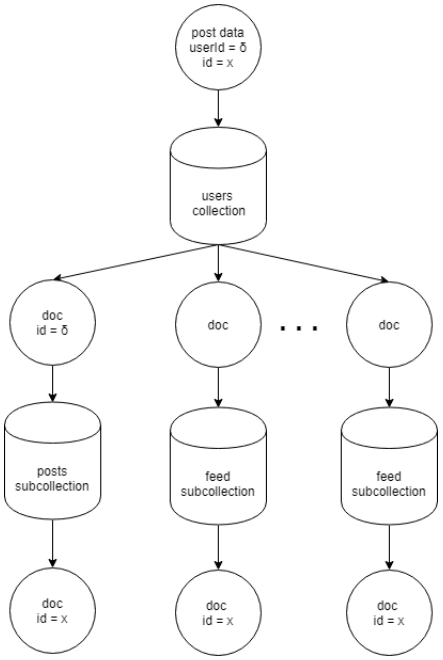


Figure 8.10: Process for updating changes made to a post in the feed

Chapter 9

Testing

Testing is vital in asserting that the system performs as expected and meets the requirements. This section outlines the distinct forms of testing that the system was subjected to, followed by a presentation the results.

9.1 Unit Testing

Unit testing involves carrying out tests on individual units of the source code to verify that they are fully functional. In the *Skouted* project, unit tests were carried out on all the individual endpoints to determine whether they gave the desired response when a request was made. A set of tests were written for each microsevice in isolation to test all the existing endpoints, using test-driven development to ensure that all endpoints in each microservice achieve the requirements by returning the desired responses. The functional requirements were broken down into specific test cases which were implemented using *chaiHTTP* [98] before implementing the endpoints to have them built to meet the requirements of the system. The tests make use of manually generated data to determine how data is read and manipulated with expectations on how the data changes during the tests (e.g. changing the name of a user in their profile using an initial name and expected new name).

9.1.1 Test Criteria

The unit tests in the system are evaluated by putting in place a set of assertions which must be met for a test to pass. The asserted status code of the test indicates the type of test, with status codes of type “2xx” indicating a successful action and “4xx” or “5xx” indicating an invalid action.

Additional assertions are placed on the response body to specify what data is expected from the response.

9.1.2 Profiles

Table 9.1: Unit tests for *profiles* microservice

Description	FRID	Assertions	Status
Creating a player	1.2, 2	status=200	COMPLETE
		response : { id : **String**, success : **String** }	COMPLETE
Creating a scout	1.1, 2	status=200	COMPLETE
		response : { id : **String**, success : **String** }	COMPLETE
Creating a player with incorrect properties	1.2, 2	status=401	COMPLETE
Fetching a player	5.5, 7.2, 7.3	status=200	COMPLETE
		response : { user : **Object** }	COMPLETE
Fetching a scout	6	status=200	COMPLETE
		response : { user : **Object** }	COMPLETE
Updating a player's details	4.4, 4.5	status=200	COMPLETE
		user.name='James Bond'	COMPLETE
Updating a player's profile photo	4.3	status=200	COMPLETE
		user.pictureURL=newURL	COMPLETE
Follow a user with an invalid ID	5.1	status=400	COMPLETE
Follow a user without having an invalid user account	5.1	status=400	COMPLETE
Follow yourself	5.1	status=400	COMPLETE

Follow a user with a valid ID and having a valid user account	5.1	status=200	COMPLETE
Get followers from invalid user ID	5.1, 5.2	status=400	COMPLETE
Get followers from valid user ID	5.1, 5.2	status=200	COMPLETE
		body : { followers : **Array** }	COMPLETE
Get following list from valid user ID	5.1, 5.2	status=200	COMPLETE
		body : { following : **Array** }	COMPLETE
Get following list from invalid user ID	5.1, 5.2	status=400	COMPLETE
Check if user follows another user with invalid user ID	5.1, 5.2	status=400	COMPLETE
Check if user follows another user with valid user ID	5.1, 5.2	status=200	COMPLETE
		body : { isfollowing : boolean }	COMPLETE
		isfollowing=true	COMPLETE
Check if user is being followed by another user with invalid user ID	5.1, 5.2	status=400	COMPLETE
Check if user is being followed by another user with valid user ID	5.1, 5.2	status=200	COMPLETE
		body : { isfollowed : boolean }	COMPLETE
		isfollowed=true	COMPLETE

9.1.3 Media

Table 9.2: Unit tests for uploading a video

Description	FRID	Assertions	Status
Uploading a valid video and thumbnail	4.1, 4.2	status=200	COMPLETE
		response : { url : **String**, thumbUrl : **String** }	COMPLETE
Uploading an image instead of a video, and valid thumbnail	4.1, 4.2	status=400	COMPLETE
Uploading a video over the maximum duration, with a valid thumbnail	4.1, 4.2	status=400	COMPLETE
Uploading a valid video with a thumbnail that's not square	4.1, 4.2	status=400	COMPLETE
Uploading a valid video with a thumbnail that exceeds the maximum dimensions	4.1, 4.2	status=400	COMPLETE
Uploading a valid video and thumbnail with incorrect file format	4.1, 4.2	status=400	COMPLETE
Uploading a valid video and a thumbnail with incorrect file format disguised as an image	4.1, 4.2	status=400	COMPLETE
No files being uploaded	4.1, 4.2	status=400	COMPLETE
No video uploaded, uploading thumbnail that is not square and too large	4.1, 4.2	status=400	COMPLETE
Uploading a valid video without a thumbnail	4.1. 4.2	status=400	COMPLETE

Uploading a video of incorrect file format in disguise with a valid thumbnail	4.1, 4.2	status=400	COMPLETE

Table 9.3: Unit tests for uploading an image

Description	FRID	Assertions	Status
Uploading a valid image	4.3	status=200	COMPLETE
		response : { pictureUrl : **String** }	COMPLETE
Uploading an image with an incorrect file format	4.3	status=400	COMPLETE
Uploading a video file disguised as an image	4.3	status=400	COMPLETE
Uploading no files	4.3	status=400	COMPLETE
Uploading an image that exceeds the maximum dimensions	4.3	status=400	COMPLETE
Uploading an image that is not square	4.3	status=400	COMPLETE

9.1.4 Posts

Table 9.4: Unit tests for commenting features

Description	FRID	Assertions	Status
Submitting a blank comment	5.3, 7.4	status=400	COMPLETE
Submitting comment to post that does not exist	5.3, 7.4	status=404	COMPLETE
Submitting comment by an invalid user	5.3, 7.4	status=404	COMPLETE

Submitting a valid comment	5.3, 7.4	status=200	COMPLETE
		response : { success : **String**, newComment : **Object** }	COMPLETE
Modifying a comment with a blank string	5.3, 7.4	status=400	COMPLETE
Modifying a comment on a post that does not exist	5.3, 7.4	status=404	COMPLETE
Modifying a comment as an invalid user	5.3, 7.4	status=404	COMPLETE
Modifying a comment that does not exist	5.3, 7.4	status=404	COMPLETE
Modifying a comment using a valid comment string	5.3, 7.4	status=200	COMPLETE
		response : { success : **String** }	COMPLETE
Replying to a comment with a blank string	5.3, 7.4	status=400	COMPLETE
Replying to a comment on a post that does not exist	5.3, 7.4	status=404	COMPLETE
Replying to a comment as an invalid user	5.3, 7.4	status=404	COMPLETE
Replying to a comment that does not exist	5.3, 7.4	status=404	COMPLETE
Successful reply to a comment	5.3, 7.4	status=200	COMPLETE
		response : { success : **String**, newComment : **Object** }	COMPLETE
Get all comments from a post	5.3, 7.4	status=200	COMPLETE

		response : { success : **String**, comments : **Array** }	COMPLETE
Get all comments from a post that does not exist	5.3, 7.4	status=404	COMPLETE
Get all comments made by a user	5.3, 7.4	status=200	COMPLETE
		response : { success : **String**, comments : **Array** }	COMPLETE
Get all comments made by an invalid user	5.3, 7.4	status=404	COMPLETE
Deleting a comment on a post that does not exist	5.3, 7.4	status=404	COMPLETE
Deleting a comment as an invalid user	5.3, 7.4	status=404	COMPLETE
Deleting a comment that does not exist	5.3, 7.4	status=404	COMPLETE
Successfully deleting a comment	5.3, 7.4	status=200	COMPLETE
		response : { success : **String** }	COMPLETE

Table 9.5: Unit tests for liking posts

Description	FRID	Assertions	Status
Liking an invalid post	5.4, 7.4	status=404	COMPLETE
Liking a post as an invalid user	5.4, 7.4	status=404	COMPLETE
Liking a valid post as a valid user	5.4, 7.4	status=200	COMPLETE
Liking a post that has been liked already	5.4, 7.4	status=200	COMPLETE

Unliking an invalid post	5.4, 7.4	status=404	COMPLETE
Unliking a post as an invalid user	5.4, 7.4	status=404	COMPLETE
Unliking a valid post as a valid user	5.4, 7.4	status=200	COMPLETE
Unliking a post that has not been liked already	5.4, 7.4	status=200	COMPLETE

Table 9.6: Unit tests for managing posts

Description	FRID	Assertions	Status
Uploading match post with valid video and match data	4	status=200	COMPLETE
Uploading training post with valid video	4	status=200	COMPLETE
Uploading training post with invalid video	4	status=400	COMPLETE
Uploading training post with invalid match stats	4	status=400	COMPLETE
Uploading training post with no match stats	4	status=200	COMPLETE
Get post data from a valid post	5.2, 5.5, 7.2, 7.4	status=200	COMPLETE
Get post data from an invalid post	5.2, 5.5, 7.2, 7.4	status=404	COMPLETE

9.1.5 Messaging

Table 9.7: Unit tests for messaging features

Description	FRID	Assertions	Status
Send message to invalid user	7.1, 9.1, 9.2	status=404	COMPLETE
Send message to valid user with empty message body	7.1, 9.1, 9.2	status=400	COMPLETE
Send valid message to a valid user	7.1, 9.1, 9.2	status=200	COMPLETE
		response : { success : **String** }	COMPLETE
Get list of all conversations as a valid user	7.1, 9.1, 9.2	status=200	COMPLETE
		response : { success : **String**, threads : **Array** }	COMPLETE
Get list of all conversations as an invalid use	7.1, 9.1, 9.2	status=404	COMPLETE
Get all messages sent and received from a valid user	7.1, 9.1, 9.2	status=200	COMPLETE
		response : { success : **String**, messages : **Array** }	COMPLETE
Get all messages sent and received from an invalid user	7.1, 9.1, 9.2	status=404	COMPLETE

9.1.6 Search

Table 9.8: Unit test for search features

Description	FRID	Assertions	Status
Get list of results for players starting with 'Sid'	8	status=200	COMPLETE
		response : { results : **Array** }	COMPLETE
		results.size > 0	COMPLETE
		results.includes(x) & x.name = 'Sidney Zieman'	COMPLETE

9.1.7 Feed and Explore

Table 9.9: Unit test for feed features

Description	FRID	Assertions	Status
Fetch feed items for user	8	status=200	COMPLETE
		response : { results : **Array** }	COMPLETE
		results.size > 0	COMPLETE
Fetch explore feed recommendations for user	10.1	status=200	COMPLETE
		response : { results : **Array** }	COMPLETE
		results.size > 0	COMPLETE
		results[0].user.name = "Patrick Brownlee"	COMPLETE

9.2 Integration Testing

Integration testing is the process of testing the interactions between different components of the system, building upon unit tests, to assert that groups of components work together, even if the

individual components pass all unit tests [99].

In this scenario, integration testing was carried out by emulating the usage of the system's back-end microservices. Specifically, a data generator was implemented to populate the system with realistic user data, using a combination of video content and thumbnails acquired online and randomised data supplied by the *faker.js* npm package [100]. This was composed as a script, found in the *common/lib/seed.js* file.

To begin with, the script generates 2000 players and 1000 scouts and uploads profile pictures corresponding to the age and gender of the profile to at least 50% of the profiles. The script then proceeds to initiate the follow action between a random mapping of profiles. This tests the *profiles* microservice.

The script then uploads posts, selecting a random video from the provided sample, and generating a random caption. Following this, the post is then liked and commented upon by a sample of users, both followers and non-followers. This tests the *posts* microservice.

Finally, the script instantiates conversations between a random sample of users, using randomly generated content to send messages between the profiles. This tests the *messaging* microservice.

The integration testing carried out ensures that the microservices work together when uploading data. The consumption of data for all other microservices was tested successfully with unit tests.

9.3 System Testing

System testing is defined as the process of testing an integrated system to verify that it meets the specified requirements [44]. Throughout this project, both white box and black box testing were performed.

9.4 White Box System Testing

White box testing is based on the analysis of the internal structure of a component or system [44], where the tester has a significant level of knowledge of the internal functioning of the software. Due to system testing being performed mostly by the project manager, a large part of it was in fact white box testing, as the PM aided in the design of the software and actively worked on it as

a front and back-end developer.

The testing was organised through epics and user stories from the requirements. For example, one of the epics tested was "Post creation", which allows a user to upload a post including a video from their match and relevant statistics. Creating a post relies internally on back-end services to upload images and videos, therefore the system testing also ensures that these individual services are functioning correctly. The expected output of the test is a success message upon the user's input of a video and relevant metadata. The test was performed, and the input matched the expected output.

Similarly, user stories such as those relating to followers were also subject to system testing. For instance, a high priority story states "Players and scouts should be able to interact with other players by following or unfollowing them". The user's input in this case would be the name of a user in the search bar, then entering their profile, and clicking the follow button shown in Figure 7.10. The expected output would be a change of colour from white to green of the button, and a change in text from "Follow" to "Following". Further, since white box testing is being applied, the tester would also verify that the searched user was added to the list of followees of the user. Finally, another expected output is that the user appears in the list of followers of the application in the user's own profile. When this test was performed, the expected output successfully matched the input.

The tests described above were carried out in a similar fashion for all the user stories and epics contained in the functional requirements table 4.1.

9.5 Black Box System Testing

Black box system testing occurs when the internal structure and implementation of the software is unknown to the tester [44]. The team also performed black box system testing with a variety of external subjects, separate to the user acceptance testing. The tests typically consisted in giving the subjects instructions to perform certain actions, and then verifying that the outputs for a given input matched the expected output. For example, several students were given the following set of instructions to test the commenting functionality:

1. Access a specific post from a feed
2. Enter a comment in the "clip-viewing" page which should now be visible

3. Check that the entered comment is rendered and viewed appropriately by the user

The above test was performed successfully, with users being able to access posts from the main feed and enter comments. Other similar tests were carried out with successful outcomes for different parts of the application.

9.6 Test Cases

The testing team have conducted several test case scenarios to ensure that the application works as it is intended to. Given below is an example of one such test case. Appendix E shows all test cases.

This table provides details about the name of both the project and module the test cases belong to, in addition to who the test cases were created, reviewed and executed by and their corresponding dates. The test cases which come under the module are then given in the table below.

Table 9.10: Information on test Cases for Signing Up

Project Name	Skouted
Module Name	Sign Up
Created by	Daniel Alarms
Date of creation	22-03-2019
Reviewed by	Akriti Pathania
Date of review	24-03-2019
Executed by	Daniel Alarms
Date of execution	24-03-2019

Table 9.11: Test Cases for signing up

TID	2
Test Name	Create an account
Test Case	Enter invalid username
Pre-Condition	User does not have an account

Test Steps	Click on sign up button; Enter invalid username; Click sign up button
Test Data	ak!i
Expected Results	Notify user of invalid username
Post Condition	Sign up screen refreshes
Actual Result	User notified of invalid username
Status	COMPLETE

9.7 User Acceptance Testing

After the application had passed all the test cases, it then went through User Acceptance Testing (UAT). The application has been tested by 12 students from the University of Warwick. Out of the 12 students, 10 had android based phones and two had iOS based phones. The students were supervised by three group members, taking notes throughout the process. The task for this testing phase was to create an account and explore the application, without any assistance from members of the group. After playing with the application, the students were given a questionnaire to fill in (shown in appendix F). Additionally, some questions about their experience were answered verbally, the results of which can be seen in Section 10. After the testing phase was completed, the accounts of the students were deleted, to ensure that *Skouted* did not retain any personal details about the individuals.

Chapter 10

Evaluation

This chapter will evaluate how the project fared in comparison to the Functional and Non-Functional Requirements stated in Sections 4.2.1 and 4.2.2, respectively. The chapter will then measure the project's level of compliance with the ISO/IEC in 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models document [101]. Finally, an overview of the feedback received from the customer and the students who participated in the User-Acceptance Test will be provided.

10.1 Evaluation of Functional Requirements

The table below provides an evaluation of the functional requirements. All high priority functional requirements were successfully completed. Any incomplete DFIDs were medium or low priority requirements. They will be completed prior to application release and have not had any impact on the success of the project.

Table 10.1: An evaluation of the functional requirements

FRID	Priority	DFID	Status	Comment
1	High	1.1	COMPLETE	User can create an account as a scout.
		1.2	COMPLETE	User can create an account as a player.
2	High	2.1	COMPLETE	Players can successfully enter their information when signing up to the platform.

		2.2	COMPLETE	When registering players under the age of 13, they are asked to provide the platform with their guardian's contact details.
		2.3	COMPLETE	When scouts sign up to the platform they are able to provide the platform with certain preferences they have for players. From this information the application will display relevant players to the scout.
		2.4	INCOMPLETE	This feature has not yet been implemented and will be implemented as a part of future work for the application.
		2.5	COMPLETE	Users are able to input their current location so correct age restriction rule can then be applied from this data.
3	High	3.1	COMPLETE	Both Scouts and players can successfully be authenticated into the system.
4	High	4.1	COMPLETE	Players can successfully upload video clips of their matches and training sessions by using the upload media feature.
		4.2	COMPLETE	Players can successfully upload a header video by using the upload media feature.
		4.3	COMPLETE	Players can successfully upload a new profile picture using the edit profile page.
		4.4	COMPLETE	Players can successfully modify their player attributes using the edit profile page.
		4.5	COMPLETE	Players can successfully enter their match statistics in addition to uploading a new video clip using the upload post page.
5	High	5.1	COMPLETE	Both scouts and players can successfully follow or unfollow other players.
		5.2	COMPLETE	Both scouts and players can successfully view posts by players they follow, via their home feed.
		5.3	COMPLETE	Both scouts and players can successfully comment on video clips posted by other players.

		5.4	COMPLETE	Both scouts and players can successfully like or unlike video clips posted by other players.
		5.5	COMPLETE	Both scouts and players can successfully visit another player's profile and view their posts and statistics.
6	High	6.1	COMPLETE	Players can successfully visit a scouts' profile and view their details.
		6.2	INCOMPLETE	The verification feature has not yet been implemented and so this requirement is on hold as of now and will be implemented as a part of future work for the application.
7	High	7.1	COMPLETE	Scouts can successfully direct message a player who is of interest to them via the platforms messaging hub.
		7.2	COMPLETE	Scouts can successfully view a player's video by tapping on the video.
		7.3	COMPLETE	Scouts can successfully view a player's profile by visiting their profile page.
		7.4	COMPLETE	Scouts can successfully comment and like a player's video which they have seen via their news feed or directly from the player's profile page.
		7.5	INCOMPLETE	Scouts being able to contact a guardian for players under the age of 13 is a feature yet to be implemented and is considered to be future work for the application.
8	High	8.1	COMPLETE	Scouts are able to specify a search criteria which is then applied to players. The players who match that criteria are then displayed to the scout.
9	High	9.1	COMPLETE	Players and scouts can successfully communicate with one another via the messaging hub.
		9.2	COMPLETE	Players can successfully communicate with one another via the messaging hub.
		9.3	COMPLETE	The platform itself can successfully communicate with players and scouts via the messaging hub.

		9.4	INCOMPLETE	The feature to relay any communication with players under the age of 13 to their guardian has not been implemented as of yet, however is considered to be of high importance so is part of the future work for this platform.
		9.5	INCOMPLETE	The system must detect spam
		9.6	INCOMPLETE	The system must prevent users suspected of sending spam from using the message feature
10	Medium	10.1	COMPLETE	On first log in, the platform allows players to successfully follow popular players in their contacts or located nearby.
		10.2	INCOMPLETE	On first log in for a scout, the platform successfully shows players that are similar to the scouts general sign up criteria.
11	Medium	11.1	COMPLETE	Scouts can successfully upload a new profile picture via the edit profile page.
		11.2	INCOMPLETE	As of right now there is no verification feature implemented. This is an area for future work for the application.
		11.3	COMPLETE	Scouts can successfully update their scouting preferences via the search page. Search results are then filtered to meet the new requirements set by the scout.
12	Medium	12.1	INCOMPLETE	As of right now, the system does not have an administrative platform, so this requirement is a part of future work for the application.
		12.2	INCOMPLETE	As of right now, the system does not have an administrative platform, so this requirement is a part of future work for the application.

13	Low	13.1	INCOMPLETE	As of right now, there is no notification system in place for when a scout views a player's profile, however is an important feature and so is a part of future works for the application.
14	Low	14.1	COMPLETE	Players can see their performance statistics posted with their video clips.
15	Medium	15.1	COMPLETE	The system allows users to find content from users they do not follow
	Medium	15.2	INCOMPLETE	This requirement has not yet been implemented but research into how this can be implemented has been conducted.
16 - 23	Low	16 - 23	INCOMPLETE	The system does not yet allow users to fully exercise their rights, nor does it implement safeguarding procedures. These requirements must be implemented before the application is released.

10.2 Evaluation of Non-Functional Requirements

Table 10.2 shows the status of the non-functional requirements. The development team successfully completed all non-functional requirements.

Table 10.2: An evaluation of the non-functional requirements

NFID	Priority	Status	Comment
1	High	COMPLETE	The system successfully provides a mobile-first interface, supporting both iOS and Android, as can be confirmed by testing the application on both platorms.
2	High	COMPLETE	The system successfully provides a discoverable user-interface and a familiar mobile user experience, as can be confirmed by the overall consensus of the User-Acceptance Testing phase.
3	Medium	COMPLETE	As the system was developed using a micro-services architecture the system is extensible and maintainable.

4	Medium	COMPLETE	After rigorously testing the system, it seems that it is robust to failure.
5	Medium	COMPLETE	As the system was developed using a micro-services architecture the system can be scaled to incorporate more features. In addition, the data structures chosen ensured that the application easily scales whether it has a hundred users or a hundred-thousand users.
6	Medium	COMPLETE	The development team has an Ethics Manager to ensure that the system is complying with the 'General Data Protection Regulation' and the 'Data Protection Act' at all times.

10.3 Usability

System usability is described as the 'degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use' by the ISO/IEC in 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models [101].

To evaluate the usability of the *Skouted* platform, the following characteristics will be assessed and presented Tables 10.3 to 10.7, using the template provided by the ISO/IEC:

- Understandability
- Documentation
- Buildability
- Installability
- Learnability

Table 10.3: An evaluation of understandability

Understandability	
PASS	
Metric	Supporting Comment

How straightforward is it to understand: • What the software does and its purpose? • The intended market and users of the software? • The mandatory functions the system must provide by the initial release? • The advanced functions the system shall provide following the initial release?	Section 1 of this document outlines the project's main objectives, purpose, intended market and stakeholders. Chapter 4 presents the mandatory functional and non-functional requirements that must be provided prior to the closure of this project and the advanced functionality required prior to the release of the application.
Is a high-level description of what and who the software is for, provided?	Yes, see the Skouted landing page (skouted.co.uk)
Is a high-level description of how the software works provided?	Yes, see the user manual in appendix H
Is an architectural overview, with diagrams provided?	Yes, see chapter 7.

Table 10.4: An evaluation of Documentation

Documentation	
PASS	
Metric	Supporting Comment
Looking at the user documentation, what is its: • Quality? • Completeness? • Accuracy? • Appropriateness? • Clarity?	The user manual provides a description of the application's functions and how to use them. The developer documentation provides installation guidelines and API information. Chapter 7 also provides a design description for the application.

Does the document consist of clear, step-by-step instructions?	The document outlines each step for doing every small task in the application in a rational order to make things clear for the user.
Does the document give examples of what the user can see at each step?	Yes, the user manual provides screenshots of what the user should see.
Is the documentation partitioned into sections for users, user-developers and developers?	Yes, the user manual provides a section for users. The API documentation provides information for developers.

Table 10.5: An evaluation of Buildability

Buildability	
PASS	
Metric	Supporting Comment
How straightforward is it to: Tests are provided to verify the build has succeeded.	Automatic tests have been implemented in order to verify the build of the system.
Is an automated build system used to build the software?	Yes, Docker handles the build process
Are all third party dependencies listed?	Yes, the *package.json* file in the source code provides all dependencies. Dependencies can be installed with *npm install* in the terminal.
Does the documentation provide instructions for building the software?	Yes, see the user manual.

Table 10.6: An evaluation of Installability

Installability	
PASS	
Metric	Supporting Comment

When software is installed, its contents are organised into sub-directories (e.g. docs for documentation, libs for dependent libraries) as appropriate	Yes.
Are tests provided to verify the install succeeded?	Yes
Is Dependency management used to automatically install all dependencies?	Yes, *npm install* can be used to install dependencies.
Is documentation provided for users and developers?	Yes.

Table 10.7: An evaluation of Learnability

Learnability	
PASS	
Metric	Supporting Comment
How straightforward is it to learn how to achieve: • Basic functional tasks? • Advanced functional tasks?	The application builds on the designs and intuitions of other well-known social media applications. It should therefore be simple for a user to learn and perform basic and more advanced tasks.
Is a getting started guide provided outlining a basic example of using the software?	Yes, see appendix H.

10.4 Sustainability and Maintainability

System maintainability is described as the 'degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers' by the ISO/IEC in 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models [101].

To evaluate the sustainability and maintainability of the *Skouted* platform, the following charac-

teristics will be assessed using the template provided by the ISO/IEC:

- Identity
- Governance
- Community
- Accessibility
- Testability
- Portability
- Analysability
- Changeability

Table 10.8: An evaluation of identity

Identity	
PASS	
Metric	Supporting Comment
To what extent is the identity of the project/software clear and unique both within its application domain and generally?	The application is identifiable by its unique colour scheme and style.
Does the project/software have its own domain name?	Yes.
Does the project/software have a logo?	Yes. And this logo is presented within the application and the website.
Does the project/software have a distinct name within its application area.	The project has a distinct name within its application area and beyond.
Does a search by Google on the name plus keywords from the application area throw up the project web site in the first page of matches?	A search on Google with input *Skouted* returns skouted.co.uk as the first hit.

Table 10.9: An evaluation of governance

Governance	
PASS	
Metric	Supporting Comment
To what extent does the project make its management, or how its software development is managed, transparent?	See section 5.
The project has defined a governance policy	Yes. Project Management is described in chapter 5. A description of the software development methodologies is provided in chapter 5.5.

Table 10.10: An evaluation of community

Community	
PASS	
Metric	Supporting Comment
To what extent does/will an active user community exist for this product?	The user community for this product will be anyone in the UK who is looking to be scouted over their footballing talents. Based on this, it is predicted that there will be a large active user community for this product.

Table 10.11: An evaluation of accessibility

Accessibility	
PASS	
Metric	Supporting Comment
Who, if anyone, has the ability to browse source code repository online?	Only the development team have access to the source code, in addition to the University of Warwick, for marking purposes only.

To what extent is the software accessible?	As of now, only the development team have access to the source code, in addition to the University of Warwick for marking purposes only. Once the application has been marketed, the application will be made available to both Android and iOS users alike.

Table 10.12: An evaluation of testability

Testability	
PASS	
Metric	Supporting Comment
How straightforward is it to test the software to verify modifications?	Unit tests automatically verify modifications to the codebase before being deployed. The application can be tested after modification by signing up as a new user and using the application as one normally would.
Does the project have unit tests?	Yes.
Does the project have integration tests?	Yes.
Does the project have automated tests to check conformance to coding standards?	The project follow the coding standards set up on ESLint.
Are test results visible to all developers/members?	Yes.
Does the project have scripts for testing scenarios that have not been automated (e.g. for testing GUIs).	Yes, for back-end development.

Table 10.13: An evaluation of portability

Portability
PASS

Metric	Supporting Comment
To what extent can the software be used on other platforms?	Users can run the application on any browser (mobile or otherwise). It is intended for use on mobile devices.
Can the application be built on and run under Android?	Yes.
Can the application be built on and run under iOS?	Yes.
Can the application be easily deployed?	Yes, the application comes in the form of a Docker container which can easily be deployed on most operating systems with few simple commands.

Table 10.14: An evaluation of analysability

Analysability	
PASS	
Metric	Supporting Comment
How straightforward is it to analyse the software's source release to: • Understand its implementation architecture? • Understand individual source code files and how they fit into the implementation architecture?	The application is created using a microservices architecture which is reflected in how the software's source code is structured. This is described in chapter 7
Is the source code structured into modules or packages?	Yes.
Does the source code structure relate clearly to the architecture or design?	Yes, it reflects the structure of using a microservices architecture and the design ideas presented in chapter 7.

Is the source code repository a version control system.	Yes, the source code is saved on GitLab, which itself has version control.
Is the source code commented?	Yes, comments have been added to describe what is being implemented and an overview of how it has been done.
Has the source code been laid out and indented well?	Yes.
Does the source code use sensible classes, packages and variable names?	Yes.
Are there any old source code files that should be handled by version control e.g. "SomeComponentOld.java"?	No.
Is there any commented out code?	No.
Are coding standards recommended by the project?	Yes, the project uses ESLint which recommends how to change the code so that it matches the set coding standard.
Are the Project-specific coding standards consistent with community or generic coding standards (e.g. for C, Java, FORTRAN etc.)?	Yes.

Table 10.15: An evaluation of changeability

Changeability	
PASS	
Metric	Supporting Comment

How straightforward is it to modify the software to: • Address issues? • Modify functionality? • Add new functionality?	It is easy to modify the software because of the microservices architecture. If there is new functionality to be added, a new microservice will be created to address that functionality, without altering any other part of the application. To modify a functionality, the code can be easily found in the microservice it relates to.

10.5 Feedback from customer

In the middle of January the group was tasked with showcasing the product built so far to the customer as a sort of checkpoint. During that period, the application had its profile, news feed, clip viewing, messaging and uploading videos page in a 'ready to show' phase. After having displayed the applications functionality, the customer was satisfied with how the development was going and the overall look of the application so far. This indicates that the application is turning out how the company had envisioned it to be and as a result the group is on-track with the project.

10.6 Feedback from potential adult users

As mentioned in the Testing section of this report, the application had gone through User-Acceptance Testing, the result of which will be detailed below.

A summary of the verbal responses received by the students when asked about their experience using the application is as follows.

- Every student found it easy and straightforward to sign up to the application.
- Every student found it easy to navigate around the application.
- Every student found it easy to upload a video and enter statistics the corresponding statistics for it.
- 10 students found the player card statistics visualisation to be "cool".
- Every student found it easy to follow another player.

- Every student found it easy to like and comment on a video.
- Every student thought that the size of widgets, such as buttons an text boxes, were large enough.
- Every student though that the general user interface of the application felt similar to other social media applications and therefore was intuitive as to what they were supposed to do.
- All the students thought that the design of the application had been maintained consistently throughout every screen.

In addition to this, the students had also been given a questionnaire to fill out, similar to what would have been given to potential users under the age of 13. The results of the questionnaire are given in the tables below.

Table 10.16: Device Application has been tested on

Which device have you tested the application on?	
Android	9
iOS	3

Table 10.17: User reaction to the product

What is your first reaction to the product?	
Amazing	7
Good	5
Neither good or Bad	0
Bad	0
Terrible	0

Table 10.18: User reaction to the application design

How visually appealing is our application design?	
Extremely appealing	3
Very appealing	9

Somewhat appealing	0
Not very appealing	0
Not at all appealing	0

Table 10.19: User reaction to ease of navigation in application

How easy is it to navigate through the application?	
Very easy	8
Easy	4
Neither easy nor difficult	0
Difficult	0
Very difficult	0

Table 10.20: User's take on mpost useful feature

Which feature(s) of the application is/ are most valuable to you?	
Being able to upload videos	6
Being able to upload match statistics	6
Being able to view other player's profiles	5
Being able to communicate with scouts	7
Being able to communicate with other players	6
All the above	5

A summary of the more descriptive questions asked is given below:

- When asked if the students themselves would use the application and why, the fair majority said yes and that they were already into playing football. Those that said no gave the reason that they were not interested in the sport but when asked if they would recommend the application to their friends or family, all of them said yes.
- When asking the students who were interested in the application, if they could see them using it frequently, almost everyone said yes, mentioning that they were excited at the opportunity

to potentially get recognised by scouts.

- When asked about what features of the application was the most useful to them, the overall consensus was to be able to communicate with scouts and that "it's made easier by this platform".
- When asking for students to describe the application design, many said that the overall design of the application looked "cool" and that the application had its own identity. Furthermore, some students described the colour scheme of the application as "eye-catching" and "something different".

10.7 Test Cases

As can be seen in appendix E, every test case has passed the scenarios created for it. This indicates that the application's features are working as they were intended to. All the test cases have been performed on both the Android and iOS operating systems. The results in appendix E therefore merge both platform results into one test case.

10.8 Challenges

There were various challenges encountered over the course of the project, which can be separated into project management challenges and technical challenges.

10.8.1 Technical Challenges

Technical challenges involved difficulties encountered in the implementation process of the project from a technological standpoint.

Defining media standards

It was necessary to have the media uploaded in the system to have a standardised set of properties, as it is certain that users would be uploading videos and images of various sizes. This is to make sure that all elements in the system appear consistent across all devices for a uniform experience. As well as this, the system has to use as little of the limited available resources as possible (storage) to increase efficiency. The system has to ensure that the users conform to a set of reasonable standards when uploading media whilst making it as easy and convenient as possible to use the

platform. This includes restricting the maximum and minimum dimensions of images and videos, only allowing square images and limiting the length of a video.

Resource Management

As mentioned above, the amount of resources available to the system such as storage are limited, meaning that it has to be as efficient as possible in the consumption of these resources to reduce the risk of failure. Limiting the size of a video or image a user could upload was not the best method as this property cannot be directly altered by the user, only indirectly by changing the dimensions. Therefore, reduction of file size was done on the system using compression. There was also the issue of having the compression executed on the server or the client machine. Having compression carried out on the server when uploading an image or video would have been more expensive as the number of upload requests increased, so it was decided that the compression would be done on the client's machine.

Search Features

The system came across a limitation when attempting to implement the search functions to find other players and scouts due to the Google Cloud Platform being unable to execute queries which filter multiple fields of different types. To overcome this, there had to be an immediate change in the architecture of the system. Elasticsearch was introduced to carry out search queries over its own database which was a duplicate of the Google Firestore database.

10.8.2 Project Management Challenges

The timeline of the project was created taking into account various obstacles such as courseworks and periods of unavailability of the team, as well as the risks assessed for the project. Estimates were made on these events to determine the completion time of each phase within the project and the overall slack time. However, it was found that some of our estimates for consumed time were optimistic, causing delays in the progression of the project. In order to avoid further delays and increase productivity, as aforementioned, the team changed from a sprint-based approach to a kanban workflow. The change loosened the bi-weekly completion deadlines of features previously had with sprints, giving the team greater flexibility to work around external deadlines.

Another major challenge was managing the relationship with the customer. As described in the

"Managing Customer Relationships" section, the customer often required iterating over specific application segments to suit the target audience's needs, especially certain user interfaces. As mentioned, the challenge often lied in compromising with the customer after several iterations, due to time constraints and external deadlines. The team managed this communication aspect well, reaching compromises over the different communication channels.

Chapter 11

Future Work

Significant progress has been made through the project with only a small number of user stories not being implemented due to scope creep where the client requested new features as well as changes to already completed work. This section outlines additional features that may be included in the product in future to create a more useful platform to both players and scouts.

11.1 Additional Sports

One way to enable the *Skouted* app to bring in a larger userbase is to open the platform to multiple sports, making it a platform for sport scouting in general. For example, sports such as basketball and other team-based sports could find as much use from the platform as football. This would take a large amount of work hence why it was not included in this project with its relatively short time frame. Opening the platform to additional sports would require experts from each sport to coach us in how to make the platform useful to scouts and players within a given sport which introduces large complexity with potentially large financial and time requirements.

11.2 Advanced Analysis

Competitors such as Wyscout provide advanced data analysis from video data from football competitions. Given that *Skouted* is a media platform, such features could also be implemented giving scouts a much quicker way of evaluating a player's skill. Features might include tracking players of interest within football match videos and detecting and counting certain actions executed by a

player. By building an advanced database of valuable data, it allows *Skouted* to be used by the wider sport community and not just players and scouts. Such data could be useful for anybody with an interest in a sport such as journalists for writing detailed articles or coaches for tracking player performance. These kinds of data driven features come with great responsibility and should only be implemented when agreed upon by the platform users and their guardians if under a certain age.

Another possibility is to provide live stream sports analysis providing real time data to interested parties. Such a feature would be useful for higher levels in sport where games are professionally filmed.

11.3 Teams

The ability to manage teams within the app was requested by the customer multiple times. After eventually discussing already requested features and prioritising with the customer it was determined that implementing the 'Teams' feature would not be feasible given the time constraints. However, it is a useful feature which would fit in well with what the app is trying to achieve. Having teams allows players and potentially coaches to scout players for their local teams if they are short on players and provide an easy means of communication.

11.4 Premium Options

The *Skouted* app will always have a free option to enable the youngest and newest players to get involved with the community. A premium option would allow the most serious players additional features to help them get noticed by scouts. Features might include:

1. Allowing a player to get pushed up the list of recommended players for a scout (if deemed a good fit for a given scout)
2. Access to exclusive events outside of the app such as scouting days
3. Access to coaching

11.5 Data Compliance

A lot of research has been conducted into the data compliance and ethical issues associated with social media platforms of this kind. Currently, most of this research has not been integrated into the platform as the focus was on getting a working version with the most essential features for testing with potential users. In future features such as parental permissions and vulnerable person protection measures mentioned in Section 3, as well as GDPR compliance would need to be put into action.

11.6 Moderation Platform

The *Skouted* app allows users to upload videos whenever they please, which needs to satisfy content guidelines provided in the terms and condition of the app. The ability to upload as they please opens the platform to abusive and inappropriate content which is not suited to the young audience this app is intended for. A moderation platform would allow *Skouted* staff to moderate the content that is allowed on the platform. This could be achieved by providing a 'report' button within the user interface allowing users to flag a post as inappropriate. From within the moderation platform, such reports could be investigated, and posts deleted if deemed to not fit the platforms content guidelines. As mentioned in Section 2, the report button could be used alongside machine learning algorithms which can analyse videos and classify them as inappropriate. For example, by classifying whether the video is sport related or not, this would mean less human resources are required to filter through the content uploaded.

11.7 Recommendation Systems

The *Skouted* platform could benefit from having the recommendation systems mentioned in Section 2 being implemented. This would allow the app to adapt the content given to users to make it better suited to them and improve the user experience. It will also allow scouts to find relevant players much easier by learning over time what they are interested in and providing similar players.

11.8 Web Platform

Scouts will generally be much older than the intended audience for this app and the app has been designed to be attractive to a younger audience. Although the app has been designed to be professional looking, it does cater to this younger audience. Therefore, scouts using the platform to find talent may find the mobile app does not enable them to efficiently find relevant players and get a good overview of a player's performance quickly. Scouts may find it much easier to use a web platform to be able to see more information about a given player at once and easily compare multiple different players on one screen. Such a feature will be necessary in future to ensure scouts can perform their jobs effectively and keep them interested in the platform.

Chapter 12

Conclusions

The following section presents an overall summary of the development of the project as well as the innovations discovered throughout. Finally, the document concludes by providing a group assessment of the project.

12.1 Project Summary

The *Skouted* project commenced by exploring the relevant football background concerning scouts, agents and the youth football market. Through conversations with the customer and their domain-specific expertise, the team obtained the necessary grounding to perform the relevant market research and investigate the scouting software available as well as any football social networks that might be considered similar to *Skouted*. Once a gap was identified, namely that of connecting football players with scouts directly via digital means, the specification for the software platform was agreed with the customer as well as the functional and non-functional requirements. Similarly, from the outset of the project, the relevant risks and surrounding ethical issues were considered and noted for further exploration and compliance.

The initial research, specification and requirements were followed by the design of the relevant application features and architecture. A choice of technologies was agreed, and a microservices back-end architecture designed to support the application's front-end. The designs for the front-end were collectively iterated over, as explored in detail in UI design section, with the final set of user interfaces achieving both a young and professional look-and-feel in order to suit young players

and scouts. The back-end was continuously tested through unit tests for each individual endpoint of the microservices as well as being subject to continuous integration testing, resulting in a successful implementation to support all of the application's front-end features.

Finally, the project explored the surrounding legal, ethical and social issues, using as a basis current legislation, existing literature and research on social networks. Chapter 3 provides an extensive list of requirements that will be implemented prior to the systems release to the public. All necessary steps and precautions must be taken to ensure the *Skouted* application is GDPR compliant and is effective at safeguarding children and vulnerable people.

As a whole, the project successfully met all the high priority requirements established in the initial specification, working effectively as a team to produce a fully functional software platform connecting young players and scouts. Both academic and customer deadlines were satisfied throughout the project's duration, and the team delivered a final software platform which the customer was satisfied with. There were challenges throughout that the team overcame collaboratively, as well as possible innovations explored. The final *Skouted* platform and the experience and knowledge gained by each of the team members have resulted in a successful project.

12.2 Innovations

The architecture used for the *Skouted* project is designed to be highly scalable. This means that the performance of the system does not suffer as the number of users in the system increases due to its ability to handle growing amounts of work, and is always prepared to enlarge as necessary. This is essential for the success of *Skouted* after development if a large and ever-growing userbase is achieved.

Microservices

The use of microservices promotes scalability in the system, as they make use of containers which are designed to self-replicate and expand as demand increases [102]. They are built to utilise virtualisation and cloud computing as core functionalitiess, both being concepts that are centred around scalability. The microservices created are deployed onto Google Kubernetes clusters on the Google Kubernetes Engine (GKE). This service manages containers according to the demands of the system, automatically scaling when necessary.

Database

The system makes use of NoSQL to achieve high data scalability due to its ability to scale horizontally across multiple servers. The data is distributed across multiple nodes, meaning that increasing capacity only requires an increase in the number of nodes rather than an increase in the capacity of a single node. This makes it much easier to scale the system on demand whilst minimising downtime and impact on performance at higher loads.

12.3 Group Assessment of the Project

In summary, the project achieved its high priority objectives and demonstrated the benefits of using a number of methodologies, approaches and best practices. The groups hard work and dedication resulted in a professional application, which satisfied the customers expectations. From the group's personal perspective, the opportunity to put into practice and implement knowledge gained over the past 4 years was invaluable and enjoyable, but not without its challenges. Each individual has had a unique experience, and will walk away with enhanced social and technical abilities.

There are still a number of requirements that need to be implemented before the application is ready for release, most importantly the legal requirements and safeguarding measures. The team will continue to work on the remaining requirements and are scheduled to release the application by August 2019.

Bibliography

[1] M. Whitehouse, *The way forward, Solutions to England's Football Failings.* Bennion Kearny Limited, 2013.

[2] A. Lawrence, "Ramos was the wrong man from sevilla," 2008. [Online]. Available: https://www.theguardian.com/sport/blog/2008/oct/25/premier-league-tottenham-hotspur

[3] J. Burt, "Diego costa: I take things to limit but i did nothing wrong," *The Telegraph*, 2015.

[4] "Wyscout." [Online]. Available: https://wyscout.com/

[5] "Scoutingsystem." [Online]. Available: https://scoutingsystem.com/

[6] "Scout7." [Online]. Available: http://info.scout7.com/en/

[7] "Eye 4 talent." [Online]. Available: https://eye4talent.com/

[8] "Sportifico." [Online]. Available: https://www.sportifico.com/

[9] "Tonsser." [Online]. Available: https://tonsser.com/

[10] Securelist.com, "Kaspersky security bulletin. spam evolution 2013," 2014.

[11] W. U. in St Louis, "Email spam detection using machine learning," 2014.

[12] S. A. et al, "Spam filtering using k – nn," 2009.

[13] A. K. et al, "Large-scale video classification with convolutional neural networks," 2013.

[14] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," 2005.

[15] E. Denham, "Beyond 2018 – data protection laws built to last," 2018. [Online]. Available: https://ico.org.uk/about-the-ico/news-and-events/news-and-blogs/2018/05/beyond-2018-data-protection-laws-built-to-last/

[16] "Children's online privacy protection rule ("coppa")," 2013. [Online]. Available: https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-online-privacy-protection-rule

[17] THE EUROPEAN PARLIAMENT AND THE COUNCIL OF THE EUROPEAN UNION, "General data protection regulation," *Official Journal of the European Union*, 2016.

[18] UK Council for Child Internet Safety, "Good practice guidance for the providers of social networking and other user-interactive services," 2010. [Online]. Available: https://www.gov.uk/government/groups/uk-council-for-child-internet-safety-ukccis

[19] D. Archard, "Child abuse: parental rights and the interests of the child." *Journal of Applied Philosophy*, vol. 7, no. 2, pp. 183–194, 10 1990.

[20] A. Westin, *Privacy and freedom.* New York: Atheneum, 1967.

[21] "Children's online privacy and freedom of expression: Industry toolkit." 2018.

[22] "Digital citizenship." [Online]. Available: https://www.tes.com/teaching-resources/digital-citizenship

[23] Livingstone, S., Stoilova, M., and Nandagiri, R, "Children's data and privacy online: Growing up in a digital age. an evidence review," London: London School of Economics and Political Science, 2019.

[24] Cassidy, W., Faucher, C., and Jackson, M., "Cyberbullying among youth: A comprehensive review of current international research and its implications and application to policy and practice," 2013.

[25] Pendergrass, W. and Wright, M., "Cyberbullied to death: An analysis of victims from recent events," *Issues in Information Systems*, vol. 15, pp. 132–140, 2014.

[26] "Children and parents: Media use and attitudes report," 2017.

[27] Livingstone, S., "EU Kids Online," 2014.

[28] "Social networking services & social media: Guidelines for safeguarding children, young people and vulnerable adults 2011," Online, August 2011. [Online]. Available: http://safeguarding.dudley.gov.uk/child/work-with-children-young-people/e-safety-and-use-of-images/

[29] UKCCIS, "Child safety online: A practical guide for providers of social media and interactive services," 2015. [Online]. Available: https://www.gov.uk/government/groups/uk-council-for-child-internet-safety-ukccis

[30] R. S. for Public Health, "#StatusOfMind: Social media and young people's mental health and wellbeing," Online, May 2017.

[31] B. Lamb, "Human diversity: Its nature, extent, causes and effects on people," 2015.

[32] S. funding club, 2000, https://www.startupfundingclub.com.

[33] Hiyacar, 2000, https://www.hiyacar.com.

[34] C. Drumond, 2000, https://www.atlassian.com/agile/scrum.

[35] S. F. Conservancy, "Git," 2019. [Online]. Available: https://git-scm.com/

[36] G. Inc, "Gitlab," 2019. [Online]. Available: https://about.gitlab.com/

[37] D. Radigan, 2000, https://www.atlassian.com/agile/kanban.

[38] M. Rehkopf, 2000, https://www.atlassian.com/agile/project-management/epics-stories-themes.

[39] W. Inc., 2000, https://www.whatsapp.com/.

[40] Slack, 2000, https://slack.com/intl/es-es/.

[41] Skype, 2000, https://www.skype.com/en/.

[42] J. Foundation, "Eslint - the pluggable linting utility for javascript and jsx," 2013. [Online]. Available: https://eslint.org/

[43] L. DeNA Co., "JSX - a faster, safer, easier JavaScript," 2013. [Online]. Available: https://jsx.github.io/

[44] I. Sommerville, *Software Engineering.* Addison-Wesley, 2011.

[45] DOE, 2000, https://energy.gov/sites/prod/files/cioprod/documents/Risk_Management.pdf.

[46] V. Singh, "Native app development vs. hybrid app development," Sep 2018.

[47] Facebook, "React native · a framework for building native apps using react." [Online]. Available: https://facebook.github.io/react-native/

[48] Google, "Flutter - beautiful native apps in record time." [Online]. Available: https://flutter.dev/

[49] B. Armour, "5 key benefits of native mobile app development," Jan 2019. [Online]. Available: https://clearbridgemobile.com/benefits-of-native-mobile-app-development/

[50] Szymon and Ula, "React native vs. native app development-pros and cons for business." [Online]. Available: https://www.polidea.com/blog/react-native-vs-native-app-developmentpros-and-cons-for-business/

[51] A. Nalwaya, "React native bridge for ios and android," Jan 2019. [Online]. Available: https://hackernoon.com/react-native-bridge-for-ios-and-android-43feb9712fcb

[52] P. Ngo.T, "Introducing dart programming language & special features," Apr 2018. [Online]. Available: https://medium.com/sk-geek/introducing-dart-programming-language-special-features-c79c1a70645e

[53] "Get started building mobile apps with javascript, angular and vue.js — nativescript." [Online]. Available: https://www.nativescript.org/getting-started-with-nativescript

[54] "How to get nativescript support." [Online]. Available: https://www.nativescript.org/enterprise

[55] "Progressive web apps," Mar 2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

[56] "Template syntax." [Online]. Available: https://angular.io/guide/template-syntax

[57] C. Menezes, "Angular 2 - a quick intro about template syntax," Apr 2016. [Online]. Available: https://medium.com/front-end-weekly/angular-2-a-quick-intro-about-template-syntax-121f9b160a64

[58] B. K. Dubey, "Difference between typescript and javascript," Feb 2019. [Online]. Available: https://www.geeksforgeeks.org/difference-between-typescript-and-javascript/

[59] E. Simons, "What exactly is react?" Apr 2019. [Online]. Available: https://thinkster.io/tutorials/what-exactly-is-react

[60] J. Neuhaus, "Angular vs. react vs. vue: A 2017 comparison," Aug 2017. [Online]. Available: https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176

[61] "What is ionic." [Online]. Available: https://ionicframework.com/what-is-ionic

[62] V. Kharlampidi, "Framework7." [Online]. Available: https://framework7.io/

[63] J. Cowart, "What is a hybrid mobile app?" Jun 2012. [Online]. Available: https://www.telerik.com/blogs/what-is-a-hybrid-mobile-app-

[64] "Get started fast." [Online]. Available: https://cordova.apache.org/

[65] A. Nadalin, "On monoliths, service-oriented architectures and microservices," Feb 2015. [Online]. Available: https://odino.org/on-monoliths-service-oriented-architectures-and-microservices/

[66] K. Team, "Microservices vs monolith vs serverless: Best architectural strategy — k&c consulting," Apr 2018. [Online]. Available: https://kruschecompany.com/blog/post/microservices-vs-monolith-best-architectural-strategy

[67] C. Richardson, "Microservices pattern: Monolithic architecture pattern." [Online]. Available: https://microservices.io/patterns/monolithic.html

[68] ——, "Microservices pattern: Microservices architecture pattern." [Online]. Available: https://microservices.io/patterns/microservices.html

[69] S. F. III, "What serverless computing really means, and everything else you need to know," Apr 2019. [Online]. Available: https://www.zdnet.com/article/what-serverless-computing-really-means-and-everything-else-you-need-to-know/

[70] "Express/node introduction." [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

[71] N. Foundation, "The node.js event loop, timers, and process.nexttick()." [Online]. Available: https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/

[72] "What is python? executive summary." [Online]. Available: https://www.python.org/doc/essays/blurb/

[73] "Flask." [Online]. Available: http://flask.pocoo.org/

[74] L. Gupta, "What is java programming language?" Dec 2018. [Online]. Available: https://howtodoinjava.com/java/basics/what-is-java-programming-language/

[75] Mar 2019. [Online]. Available: https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html

[76] L. Olivera, "Everything you need to know about (relational) databases," January 2019. [Online]. Available: https://dev.to/lmolivera/everything-you-need-to-know-about-relational-databases-3ejl

[77] "what is mysql," edureka!

[78] 2ndQuadrant, "postgresql vs mysql."

[79] "About sqlite." [Online]. Available: https://www.sqlite.org/about.html

[80] "Cloud firestore documentation — cloud firestore — google cloud." [Online]. Available: https://cloud.google.com/firestore/docs/

[81] "Cloud storage documentation." [Online]. Available: https://cloud.google.com/storage/docs/

[82] S. J. Vaughan-Nichols, "What is docker and why is it so darn popular?" Mar 2018. [Online]. Available: https://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/

[83] S. Yegulalp, "What is kubernetes? container orchestration explained," Apr 2019. [Online]. Available: https://www.infoworld.com/article/3268073/what-is-kubernetes-container-orchestration-explained.html

[84] "gke overview — kubernetes engine documentation — google cloud."

[85] "Gitlab continuous integration & delivery." [Online]. Available: https://about.gitlab.com/product/continuous-integration/

[86] "Full text search — cloud firestore — google cloud." [Online]. Available: https://cloud.google.com/firestore/docs/solutions/search

[87] R. Heck, "How to improve your full-text search in elasticsearch with ngram tokenizer," May 2017. [Online]. Available: https://devticks.com/how-to-improve-your-full-text-search-in-elasticsearch-with-ngram-tokenizer-e346f29f8ddb

[88] K. Team, "About." [Online]. Available: https://www.keycloak.org/about.html

[89] A. E. Amri, "Google kubernetes engine; explain like i'm five!" Feb 2019. [Online]. Available: https://medium.com/devopslinks/google-kubernetes-engine-explain-like-im-five-1890e550c099

[90] "Screen sizes." [Online]. Available: http://screensiz.es/iphone-6-2

[91] M. Casserly, "Iphone vs android market share," 2019. [Online]. Available: https://www.macworld.co.uk/feature/iphone/iphone-vs-android-market-share-3691861/

[92] V. Peterson, "Design considerations for mobile," 2012. [Online]. Available: https://www.symantec.com/connect/blogs/design-considerations-mobile

[93] S. Kristoffersen, "Learnability and robustness of user interfaces," 2008.

[94] R. Picking, V. Grout, J. McGinn, J. Crisp, and H. Grout, "Simplicity, consistency, universality, flexibility and familiarity: The scuff principles for developing user interfaces for ambient computer systems." *IJACI*, vol. 2, pp. 40–49, 07 2010.

[95] A. P. Conn, "Time affordances: The time factor in diagnostic usability heuristics," in *CHI*, 1995.

[96] J. Nielson, *Usability Engineering*. Academic Press, 1993.

[97] V8, "V8 javascript engine." [Online]. Available: https://v8.dev/

[98] J. Luer, "Chai http - http integration testing with chai assertions." [Online]. Available: https://www.chaijs.com/plugins/chai-http/

[99] "Integration testing," Mar 2018. [Online]. Available: http://softwaretestingfundamentals.com/integration-testing/

[100] Marak, "Marak/faker.js," Feb 2018. [Online]. Available: https://github.com/marak/Faker.js/

[101] ISO, "Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models," International Organization for Standardization, ISO/IEC 25010:2011, 2011.

[102] T. Bradley, "The challenges of scaling microservices," 2017. [Online]. Available: https://techbeacon.com/app-dev-testing/challenges-scaling-microservices

Appendix A

User Stories

User Persona	Description
Player	An individual football player that wishes upload content and gain exposure to potential scouts
Underage Player	A player that is aged 13 or below
Guardian	A parent/carer of a player aged 13 or below
Scout	An individual that wishes to discover players relevant to a desired search criteria
User	A player or scout, as described above
Unauthenticated User	Any individual that is not logged into the system
Administrator	An individual that wishes to monitor the status of the system

Table A.1: User Personas and Descriptions

UID	User Story	Requirement(s) ID
1	As a player, I want to sign up to the platform, so that I can use the player-oriented services of the system to attract a scout.	1.2, 2
2	As a scout, I want to sign up to the platform, so that I can use the scout-oriented services of the system to discover player talent.	1.1, 2
3	As a player, I want to provide information about my skills when I sign up, so that scouts can more easily target my profile	2.1

4	As a scout, I want to provide information about my player preferences when I sign up, so that I can more easily target relevant players	2.2
5	As an underage player, I want to provide contact details of a guardian, so that I am able to use the messaging hub and my guardian is aware of my communication	2.3
6	As a scout, I want to upload identity documents that verify my membership to a scouting organisation, so that players know that I am trustworthy	2.4, 11.2
7	As a user, I want to sign into the platform, so that I can access the platform's functionality with my identity	3
8	As an unauthenticated user, I want to browse player and scout profiles, so that I that can view information without having an account on the platform	3.1
9	As a player, I want to upload video clips to my profile, so that scouts and other players may view my football gameplay footage	4.1
10	As a player, I want to upload a header video, so that scouts and other players may easily see a self-selected highlight of my gameplay footage	4.2
11	As a user, I want to upload a profile picture, so that other users may identify my profile visually	4.3, 11.1
12	As a player, I want to modify my player attributes, so that scouts can easily view my strengths	4.4
13	As a player, I want to upload my match statistics, so that scouts can view my historical performance	4.5, 14
14	As a user, I want to follow and unfollow other players, so that their latest posts appear on my home feed	5.1
15	As a user, I want to view the latest posts by players I follow in my home feed, so that I can view their latest footage clips	5.2
16	As a user, I want to comment on posts by players, so that I can provide feedback or my thoughts	5.3

17	As a user, I want to like or unlike posts by players, so that I can indicate a positive sentiment to the player's video clip	5.4
18	As a user, I want to visit a player's profile, so that I can view more information about the player and a full listing of their uploaded video clips	5.5
19	As a player, I want to use the message hub to contact users directly, so that we can have a discussion on any matters	6.1, 9.1, 9.2
20	As a scout, I want to use the message hub to contact players directly, so that we can have a discussion on any matters	6.1, 7.1, 9.1
21	As a player, I want to view a scout's details found on their profile, so that I can see if I fit under their preferences or criteria	6.2
22	As a scout, I want to be able to view a player's profile which contains their information and videos, so that I can see for myself if they perform to my preferences	7.2, 7.3
23	As a scout, I want to be able to view insights into a players performance, so that I can better identify talented players	14
24	As a scout, I want to comment and react to a player's video, so that I can express what my opinion is of their content	7.4
25	As a scout, I want to be able to contact an underage player's guardian via email, so that I can express my interest in the player	7.5
26	As a player, I want to receive messages from the platform directly, so that I can be aware of any changes, updates and opportunities provided by the system	9.3
27	As a scout, I want to receive messages from the platform directly, so that I can be aware of any changes, updates and opportunities provided by the system	9.3
28	As a scout, I want communications with underage players to be forwarded to their guardian, so that they are automatically updated with anything I discuss with their guardian	9.4

29	As a guardian of a player on the platform, I want any communications my ward has with other players to be relayed to me, so that I can monitor their activity on the system and ensure there are no concerns.	9.4
30	As a player, I want to be shown popular players, players in my contacts, or local players the first time I login to the system, so that I have relevant content on my feed that I can interact with when I first use the platform.	10.1
31	As a scout, I want to be shown players that are similar to my general search criteria, so that it is easier for me to find players that match my preferences	10.2
32	As a user, I want to be able to update my profile picture, so that other users can identify me visually	11.1
33	As a scout, I want to be able to update my scouting preferences on my profile, so that I can ensure my player suggestions stay relevant in the event that my criteria for players changes.	11.3
34	As an administrator, I want to be able to moderate sign-ups, flag inappropriate messages and flag malicious accounts on the administrative platform to regulate the system for users	12.1
35	As an administrator, I want to view the usage metrics of the system using the administrative platform, so that they can be used to further improve the system	12.2
36	As a player, I want to be notified when a scout views my profile, so that I am aware that I am gaining exposure	13

Table A.2: User Stories

Appendix B

Skouted Child-Friendly Terms and Conditions

Our Rules and Content Guidelines:

1. You must be **13** or over to use *Skouted* without parental consent
2. Don't lie about yourself when you set up your *Skouted* account, this includes your age.
3. You must gain parental consent from your parent or guardian who is over 18 years old.
4. Don't post anything non-football related.
5. Don't post anything showing violence, or that might make other people scared or hurt their feelings, or images showing nudity.
6. Don't use anybody else's account without their permission or try to find out their login details.
7. Keep your password secret and don't let anyone else use your *Skouted* account.
8. Don't bully anyone or post anything horrible about people.
9. Don't post other peoples' private or personal information.
10. Don't use *Skouted* to do anything illegal or that we tell you not to do.
11. Don't change anything about our website or applications, upload any type of virus or do anything that might interfere with the way *Skouted* works.

12. Don't do anything that might affect how other people use and enjoy *Skouted* .
13. Don't encourage anyone to break these rules.

Your Rights:

1. You have the right to feel safe using *Skouted*
2. If we have deleted something of yours and you think it's unfair, you have the right to complain.
3. We will let you know if we change our terms and conditions, including how we use your data. If you keep using *Skouted* after we've told you about the changes, we'll assume that you are okay with them.
4. If you break these rules, you are responsible.
5. You have the right to ask for the data held about you, and ask us to edit or delete your data.
6. You can close your *Skouted* at any time and all your data will be deleted.
7. You can opt-in and out of parental controls with the consent of your parents if you are underage.
8. You have the right to report content or users which make you scared or sad.
9. You will be able to block content or users that you don't wish to see.

Our Rights:

1. *Skouted* can collect information about you including:
    - Everything you tell us when you set up your account
    - Details about what you 'like' or post
2. We use your information to improve safety on *Skouted* . If we think something you post looks suspicious or might be breaking our rules, we will investigate it.
3. If one of your posts has been reported, we may temporarily remove your post while we investigate it.
4. If you are found to be breaking these rules, we may delete your *Skouted* account and you will not be able to make another one.

Appendix C

Skouted Terms and Conditions

By downloading or using the Skouted app, these terms will automatically apply to you – you should make sure therefore that you read them carefully before using the app. You're not allowed to copy, or modify the app, any part of the app, or our trademarks in any way. You're not allowed to attempt to extract the source code of the app, and you also shouldn't try to translate the app into other languages, or make derivative versions. The app itself, and all the trade marks, copyright, database rights and other intellectual property rights related to it, still belong to Skouted.

Skouted is committed to ensuring that the app is as useful and efficient as possible. For that reason, we reserve the right to make changes to the app or to charge for its services, at any time and for any reason. We will never charge you for the app or its services without making it very clear to you exactly what you're paying for.

The Skouted app stores and processes personal data that you have provided to us, in order to provide our Service. It is your responsibility to keep your phone and access to the app secure. We therefore recommend that you do not jailbreak or root your phone, which is the process of removing software restrictions and limitations imposed by the official operating system of your device. It could make your phone vulnerable to malware/viruses/malicious programs, compromise your phone's security features and it could mean that the Skouted app won't work properly or at all.

You should be aware that there are certain things that Skouted will not take responsibility for.

Certain functions of the app will require the app to have an active internet connection. The connection can be Wi-Fi, or provided by your mobile network provider, but Skouted cannot take responsibility for the app not working at full functionality if you don't have access to Wi-Fi, and you don't have any of your data allowance left.

If you're using the app outside of an area with Wi-Fi, you should remember that your terms of the agreement with your mobile network provider will still apply. As a result, you may be charged by your mobile provider for the cost of data for the duration of the connection while accessing the app, or other third party charges. In using the app, you're accepting responsibility for any such charges, including roaming data charges if you use the app outside of your home territory (i.e. region or country) without turning off data roaming. If you are not the bill payer for the device on which you're using the app, please be aware that we assume that you have received permission from the bill payer for using the app.

Along the same lines, Skouted cannot always take responsibility for the way you use the app i.e. You need to make sure that your device stays charged – if it runs out of battery and you can't turn it on to avail the Service, Skouted cannot accept responsibility.

With respect to Skouted's responsibility for your use of the app, when you're using the app, it's important to bear in mind that although we endeavour to ensure that it is updated and correct at all times, we do rely on third parties to provide information to us so that we can make it available to you. Skouted accepts no liability for any loss, direct or indirect, you experience as a result of relying wholly on this functionality of the app.

At some point, we may wish to update the app. The app is currently available on – the requirements for system (and for any additional systems we decide to extend the availability of the app to) may change, and you'll need to download the updates if you want to keep using the app. Skouted does not promise that it will always update the app so that it is relevant to you and/or works with the version that you have installed on your device. However, you promise to always accept updates to the application when offered to you, we may also wish to stop providing the app, and may terminate use of it at any time without giving notice of termination to you. Unless we tell you otherwise, upon any termination, (a) the rights and licenses granted to you in these terms

will end; (b) you must stop using the app, and (if needed) delete it from your device.

Changes to This Terms and Conditions

We may update our Terms and Conditions from time to time. Thus, you are advised to review this page periodically for any changes. We will notify you of any changes by posting the new Terms and Conditions on the application. These changes are effective immediately after they are posted on the application.

Contact Us

If you have any questions or suggestions about our Terms and Conditions, do not hesitate to contact us at info@skouted.co.uk.

This Terms and Conditions page was generated by App Privacy Policy Generator

Appendix D

Risk Mitigation Forms

Statement of Risk: ID 1: Project overruns deadline		
S	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 12
FT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Plan practically about how to manage the project throughout the time period available Schedule more time on parts which seem more challenging		
Post Mitigation Risk Score: 5 (Rare, Catastrophic)		
Contingency Trigger and Action Trigger: Underestimating work load of project Action: Ensure that group is meeting the ckeckpoints along the way		

Table D.1: Risk Management Form: Risk ID 1

Statement of Risk: ID 2: Project running behind schedule		
S	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 12
FT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Plan practically about how to manage the project throughout the time period available Schedule more time on parts which seem more challenging		
Post Mitigation Risk Score: 9 (Possible, Moderate)		
Contingency Trigger and Action Trigger: Underestimating work load to project load balance Action: Help any group member who may be falling behind		

Table D.2: Risk Management Form: Risk ID 2

Statement of Risk: ID 3: Implementing beyond skills set of developers		
Q	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 9
NT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Developers should make time early on to learn the skills required for this project Developers should learn in the job		
Post Mitigation Risk Score: 4 (Rare, Significant)		
Contingency Trigger and Action Trigger: Developers have never used the required programming languages before Action: Developer should find time to practise using that language		

Table D.3: Risk Management Form: Risk ID 3

Statement of Risk: ID 4: Poor communication with stakeholders		
Q	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 8
NT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Schedule meeting with the stakeholders Communicate with client on Whatsapp to keep them informed of the development		
Post Mitigation Risk Score: 3 (Rare, Moderate)		
Contingency Trigger and Action Trigger: Group get busy of the implementation side and leaves out stakeholders Action: Business Analyst role is to always keep the client informed of what the group is doing		

Table D.4: Risk Management Form: Risk ID 4

Statement of Risk: ID 5: Hardware malfunction which causes loss of work		
S	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 5
NT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Have backups of project work on a drive Sync project on GitLab continuously		
Post Mitigation Risk Score: 2 (Unlikely, Negligible)		
Contingency Trigger and Action Trigger: Virus infiltrates a machine Action: Restore project from GitLab/ drive		

Table D.5: Risk Management Form: Risk ID 5

Statement of Risk: ID 6: Key staff are ill or unavailable at critical times in the project schedule		
P	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 5
NT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Ensure no one is under too much pressure that it affects their health Spot symptoms earlier on and allow the team member to rest		
Post Mitigation Risk Score: 5 (Almost Certain, Negligible)		
Contingency Trigger and Action Trigger: Any health issue Action: Equally distribute members work to rest of the team until they have recovered		

Table D.6: Risk Management Form: Risk ID 6

Statement of Risk: ID 7: A competitive product is marketed before the system is read for release		
S	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 4
NT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Deliver the application to the client as soon as possible Make the application as unique as possible		
Post Mitigation Risk Score: 4 (Likely, Moderate)		
Contingency Trigger and Action Trigger: A competitor company releases a similar application before the client does Action: Make the client's application stand out from its competitor		

Table D.7: Risk Management Form: Risk ID 7

Statement of Risk: ID 8: Underestimation of system size and complexity		
P	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 9
NT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Take time to plan the architecture of the system Use data structures that allow scalability		
Post Mitigation Risk Score: 6 (Unlikely, Moderate)		
Contingency Trigger and Action Trigger: Poor architectural planning early on in the project Action: Reconstruct the architectural design to meet the requirements of system		

Table D.8: Risk Management Form: Risk ID 8

Statement of Risk: ID 9: Client rejection upon reveal of the finalised platform		
C	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 4
FT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Communicate with client throughout the development of the project Mutually agree upon a number of requirements for the final system to have accomplished		
Post Mitigation Risk Score: 2 (Rare, Low)		
Contingency Trigger and Action Trigger: No communication with clients on the way the project is going Action: Schedule meeting with clients every so often		

Table D.9: Risk Management Form: Risk ID 9

Statement of Risk: ID 10: Imbalance in workload distribution and/ or contribution between developers		
S	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 4
NT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Plan out how to delegate tasks early on Schedule more time on parts which seem more challenging		
Post Mitigation Risk Score: 2 (Rare, Low)		
Contingency Trigger and Action Trigger: A team member becomes lazy and does not do their designated work Action: Project Manager should speak to team member to motivate them		

Table D.10: Risk Management Form: Risk ID 10

Statement of Risk: ID 11: Disagreements surrounding product development strategies between developers		
Q	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 12
NT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Members of group should discuss and finalise strategies earlier on in the project Vote on the possible options available		
Post Mitigation Risk Score: 6 (Unlikely, Moderate)		
Contingency Trigger and Action Trigger: Team members have opposing beliefs as to which development strategy is the best Action: Discuss as a team and then vote on the best strategy		

Table D.11: Risk Management Form: Risk ID 11

Statement of Risk: ID 12: The software defect rate is excessive and repair time is underestimated		
SP	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 10
NT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Research about the software being used and if there are any compatibility issues identified Used software which has been used before and is known to not have bugs		
Post Mitigation Risk Score: 9 (Possible, Moderate)		
Contingency Trigger and Action Trigger: Using a new combination of software Action: Research into how to solve the problem faced		

Table D.12: Risk Management Form: Risk ID 12

Statement of Risk: ID 13: Changes in resources such as APIs causes unanticipated problems within the program		
P	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 8
NT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Plan what resources are required at the start of the project Schedule more time on parts which seem more challenging		
Post Mitigation Risk Score: 6 (Unlikely, Moderate)		
Contingency Trigger and Action Trigger: Old resources may not be compatible with the new ones used Action: Find alternative to the new resources used that are know to be compatible		

Table D.13: Risk Management Form: Risk ID 13

Statement of Risk: ID 14: Underestimation of system storage requirements		
PC	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 12
FT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Use data structure which are easily scalable		
Post Mitigation Risk Score: 9 (Possible, Moderate)		
Contingency Trigger and Action Trigger: System can only handle so much data before it crashes Action: Change data structure used to a more scalable one		

Table D.14: Risk Management Form: Risk ID 14

Statement of Risk: ID 15: Project Management approach hinders progression		
S	Consequence: (**C**ost, **S**chedule, **P**erformance, **Q**uality)	Risk Score: 12
FT	Time frame of risk: (**N**ear-**T**erm, **F**ar-**T**erm)	
Mitigation Strategy: **Accept** Decide which software methodology would be best suited to the project Take into account of possible situations that could come up and plan in advance for them		
Post Mitigation Risk Score: 6 (Unlikely, Moderate)		
Contingency Trigger and Action Trigger: Project has been poorly planned early on Action: Properly re-plan the direction of the project and proceed from there on out		

Table D.15: Risk Management Form: Risk ID 15

Appendix E

Test Cases

Table E.1: Information on test Cases for Signing Up

Project Name	Skouted
Module Name	Sign Up
Created by	Daniel Alarms
Date of creation	21-03-2019
Reviewed by	Akriti Pathania
Date of review	24-03-2019
Executed by	Daniel Alarms
Date of execution	24-03-2019

Table E.2: Test Cases for signing up

TID	1
Test Name	Create an account
Test Case	Enter valid password and username
Pre-Condition	User does not have an account
Test Steps	Click on sign up button; Enter valid username, password and confirm password; Click sign up button

Test Data	username: John Doe; password: 123456; confirm password: 123456
Expected Results	User is allowed access into the application
Post Condition	Screen changes to Feed
Actual Result	User is allowed access into the application
Status	COMPLETE

Table E.3: Test Cases for signing up

TID	2
Test Name	Create an account
Test Case	Enter invalid username
Pre-Condition	User does not have an account
Test Steps	Click on sign up button; Enter invalid username; Click sign up button
Test Data	ak!i
Expected Results	Notify user of invalid username
Post Condition	Sign up screen refreshes
Actual Result	User notified of invalid username
Status	COMPLETE

Table E.4: Test Cases for signing up

TID	3
Test Name	Create an account
Test Case	Enter invalid password
Pre-Condition	User does not have an account
Test Steps	Click on sign up button; Enter invalid password; Click sign up button
Test Data	12
Expected Results	Notify user of invalid password

Post Condition	Sign up screen refreshes
Actual Result	User notified of invalid password
Status	COMPLETE

Table E.5: Test Cases for signing up

TID	4
Test Name	Create an account
Test Case	Enter mismatching password
Pre-Condition	User does not have an account
Test Steps	Click on sign up button; Enter mismatching passwords; Click sign up button
Test Data	password: 123456 confirm password: 124356
Expected Results	Notify user of invalid password
Post Condition	Sign up screen refreshes
Actual Result	User notified of invalid password
Status	COMPLETE

Table E.6: Information on test Cases for Log In

Project Name	Skouted
Module Name	Log In
Created by	Daniel Alarms
Date of creation	21-03-2019
Reviewed by	Akriti Pathania
Date of review	24-03-2019
Executed by	Daniel Alarms
Date of execution	24-03-2019

Table E.7: Test Cases for logging in

TID	5
Test Name	Log in to account
Test Case	Enter valid username and valid password
Pre-Condition	User has an account
Test Steps	Enter username; Enter password; Click sign in button
Test Data	valid username and valid password
Expected Results	User is allowed access into the application
Post Condition	Screen changes to Feed
Actual Result	User is allowed access into the application
Status	COMPLETE

Table E.8: Test Cases for logging in

TID	6
Test Name	Log in to account
Test Case	Enter invalid username and correct password
Pre-Condition	User has an account
Test Steps	Enter username; Enter password; Click sign in button
Test Data	invalid username and valid password
Expected Results	Notify user of an error and deny access
Post Condition	Log in screen refreshes
Actual Result	User notified of an error and denied access
Status	COMPLETE

Table E.9: Test Cases for logging in

TID	7
Test Name	Log in to account
Test Case	Enter invalid username and invalid password
Pre-Condition	User has an account
Test Steps	Enter username; Enter password; Click sign in button

Test Data	invalid username and invalid password
Expected Results	Notify user of an error and deny access
Post Condition	Log in screen refreshes
Actual Result	User notified of an error and denied access
Status	COMPLETE

Table E.10: Test Cases for logging in

TID	8
Test Name	Log in to account
Test Case	Enter valid username and invalid password
Pre-Condition	User has an account
Test Steps	Enter username; Enter password; Click sign in button
Test Data	valid username and invalid password
Expected Results	Notify user of an error and deny access
Post Condition	Log in screen refreshes
Actual Result	User notified of an error and denied access
Status	COMPLETE

Table E.11: Information on test Cases for Feed

Project Name	Skouted
Module Name	Feed
Created by	Daniel Alarms
Date of creation	21-03-2019
Reviewed by	Akriti Pathania
Date of review	24-03-2019
Executed by	Daniel Alarms
Date of execution	24-03-2019

Table E.12: Test Cases for Feed

TID	9
Test Name	Using Feed
Test Case	Liking a video
Pre-Condition	User has followers
Test Steps	Click on feed; Click like button under video;
Test Data	NA
Expected Results	Like button changes colour once clicked and like counter get incremented by 1
Post Condition	Darker like button and like count changes
Actual Result	Like button changes colour once clicked and like counter get incremented by 1
Status	COMPLETE

Table E.13: Test Cases for Feed

TID	10
Test Name	Using Feed
Test Case	Unliking a video
Pre-Condition	User has followers
Test Steps	Click on feed; Click like button under video;
Test Data	NA
Expected Results	Like button reverts to normal colour when clicked and like counter gets decremented by 1
Post Condition	Lighter like button and like count changes
Actual Result	Like button reverts to normal colour when clicked and like counter gets decremented by 1
Status	COMPLETE

Table E.14: Test Cases for Feed

TID	11
Test Name	Using Feed
Test Case	Commenting on video
Pre-Condition	User has followers
Test Steps	Click on feed; Click comment button under video; Type a comment; Click send
Test Data	"Any random comment"
Expected Results	Comment should get added to the list of comments under the video, sorted by its date
Post Condition	Comment is added to comment section according to its date
Actual Result	Comment gets added to the list of comments under the video, sorted by its date
Status	COMPLETE

Table E.15: Test Cases for Feed

TID	12
Test Name	Using Feed
Test Case	Playing video
Pre-Condition	User has followers
Test Steps	Click on feed; Click on a video; Repeat for clip viewing page
Test Data	NA
Expected Results	Video should play
Post Condition	Video is playing
Actual Result	Video plays
Status	COMPLETE

Table E.16: Test Cases for Feed

TID	13
Test Name	Using Feed

Test Case	Pausing video
Pre-Condition	User has followers
Test Steps	Click on feed; Click on a video; Repeat for clip viewing page
Test Data	NA
Expected Results	Video should pauses
Post Condition	Video is paused
Actual Result	Video pauses
Status	COMPLETE

Table E.17: Information on test Cases for Profile Page

Project Name	Skouted
Module Name	Profile Page
Created by	Daniel Alarms
Date of creation	21-03-2019
Reviewed by	Akriti Pathania
Date of review	24-03-2019
Executed by	Daniel Alarms
Date of execution	24-03-2019

Table E.18: Test Cases for Profile Page

TID	14
Test Name	Using Profile Page
Test Case	Header video plays at entrance
Pre-Condition	User has atleast one video
Test Steps	Click on profile page;
Test Data	NA
Expected Results	Header video should play automatically
Post Condition	Header video continues playing automatically
Actual Result	Header video play automatically

Status	COMPLETE

Table E.19: Test Cases for Profile Page

TID	15
Test Name	Using Profile Page
Test Case	Playing videos
Pre-Condition	User has atleast two videos
Test Steps	Click on profile page; Click on a video
Test Data	NA
Expected Results	Redirect to clip viewing page to play video
Post Condition	Screen changes to clip viewing page
Actual Result	Redirect to clip viewing page to play video
Status	COMPLETE

Table E.20: Test Cases for Profile Page

TID	16
Test Name	Using Profile Page
Test Case	Editing player data on page
Pre-Condition	User has an account
Test Steps	Click on profile page; Click on the edit profile button; Edit height; Click save
Test Data	new height: 1.83m
Expected Results	Profile page screen should update to show new height
Post Condition	Player's height data is updated to be 1.83m
Actual Result	Profile page screen updates to show new height
Status	COMPLETE

Table E.21: Test Cases for Profile Page

TID	17
Test Name	Using Profile Page
Test Case	Editing bounded player data on page
Pre-Condition	User has an account
Test Steps	Click on profile page; Click on the edit profile button; Edit height to one above 2.10m; Click save
Test Data	new height: 2.20m
Expected Results	Edit profile should not allow result
Post Condition	Remain on edit profile screen
Actual Result	Edit profile does not allow result
Status	COMPLETE

Table E.22: Test Cases for Profile Page

TID	18
Test Name	Using Profile Page
Test Case	Editing player profile picture on page
Pre-Condition	User has an account
Test Steps	Click on profile page; Click on the edit profile button; Click on add new picture button; Upload a picture; Click save
Test Data	Any random picture
Expected Results	Profile page screen should update to show new profile picture
Post Condition	Player's picture is set to be the uploaded one
Actual Result	Profile page screen updated to show new profile picture
Status	COMPLETE

Table E.23: Information on test Cases for Search

Project Name	Skouted
Module Name	Search
Created by	Daniel Alarms

Date of creation	21-03-2019
Reviewed by	Akriti Pathania
Date of review	24-03-2019
Executed by	Daniel Alarms
Date of execution	24-03-2019

Table E.24: Test Cases for Search Page

TID	19
Test Name	Using Search Page
Test Case	Search for a player on the platform
Pre-Condition	User and searched player have an account
Test Steps	Click on search page; Type player's name in search bar
Test Data	A player's name who has an account on this platform
Expected Results	Search page screen should display a list of player profile's whose name begins with the characters entered
Post Condition	The list displays all the players with the entered name
Actual Result	Search page screen displays a list of player profile's whose name begins with the characters entered
Status	COMPLETE

Table E.25: Test Cases for Search Page

TID	20
Test Name	Using Search Page
Test Case	Search for a player who is not on the platform
Pre-Condition	User has an account and searched player does not
Test Steps	Click on search page; Type player's name in search bar
Test Data	A player's name who does not have an account on this platform

Expected Results	Search page screen should not end up displaying any person's profile who has that name
Post Condition	The list displays all the players with the entered name
Actual Result	Search page screen does not display any person's profile, who has that name
Status	COMPLETE

Table E.26: Test Cases for Search Page

TID	21
Test Name	Using Search Page
Test Case	Do a search for all players that match a set criteria
Pre-Condition	A player has an account that matches the criteria set
Test Steps	Click on search page; Enter the search criteria for the player
Test Data	
Expected Results	Search page screen should display one person's profile
Post Condition	One profile is displayed in a list
Actual Result	Search page screen displays one person's profile
Status	COMPLETE

Table E.27: Test Cases for Search Page

TID	22
Test Name	Using Search Page
Test Case	Do a search for all players that do not match a set criteria
Pre-Condition	No player has an account that matches the criteria set
Test Steps	Click on search page; Enter the search criteria for the player
Test Data	
Expected Results	Search page screen should not display anyone's profile
Post Condition	Search page screen displays no results
Actual Result	Search page screen should not display anyone's profile

Status	COMPLETE

Table E.28: Test Cases for Search Page

TID	23
Test Name	Using Search Page
Test Case	Visit player's profile page
Pre-Condition	User and another player has an account
Test Steps	Click on search page; Enter the search criteria for the player; tap on player's tab
Test Data	NA
Expected Results	Search page screen should redirect to that player's profile screen
Post Condition	User sees player's profile screen
Actual Result	Search page screen redirects to that player's profile screen
Status	COMPLETE

Table E.29: Information on test Cases for Messaging

Project Name	Skouted
Module Name	Messaging
Created by	Daniel Alarms
Date of creation	21-03-2019
Reviewed by	Akriti Pathania
Date of review	24-03-2019
Executed by	Daniel Alarms
Date of execution	24-03-2019

Table E.30: Test Cases for Messaging Page

TID	24

Test Name	Using Messaging Page
Test Case	Send player a message
Pre-Condition	User and another player has an account
Test Steps	Click on profile page of player; Click on messaging icon; Type message in input bar; Click send
Test Data	"This is a random message"
Expected Results	Message should be seen on both receiver's and sender's screen
Post Condition	Message is posted on the screen to the right (if sender) or left (if receiver), along with the timestamp for when the message was sent
Actual Result	Message is seen on both receiver's and sender's screen
Status	COMPLETE

Table E.31: Test Cases for Messaging Page

TID	25
Test Name	Using Messaging Page
Test Case	Send scout a message
Pre-Condition	User and a scout has an account
Test Steps	Click on profile page of scout; Click on messaging icon; Type message in input bar; Click send
Test Data	"This is a random message"
Expected Results	Message should be seen on both receiver's and sender's screen
Post Condition	Message is posted on the screen to the right (if sender) or left (if receiver), along with the timestamp for when the message was sent
Actual Result	Message is seen on both receiver's and sender's screen
Status	COMPLETE

Table E.32: Test Cases for Messaging Page

TID	26
Test Name	Using Messaging Page
Test Case	Send another user a very long message
Pre-Condition	Two users should have an account
Test Steps	Click on profile page of the other user; Click on messaging icon; Type message in input bar; Click send
Test Data	"This is a very very very very very very very very very very very very very very very very very long test message"
Expected Results	Message should be seen on both receiver's and sender's screen and and covers about 3/4 of the screen
Post Condition	Message is posted on the screen to the right (if sender) or left (if receiver), along with the timestamp for when the message was sent
Actual Result	Message is seen on both receiver's and sender's screen and covers about 3/4 of the screen
Status	COMPLETE

Table E.33: Information on test Cases for Messenger

Project Name	Skouted
Module Name	Messenger
Created by	Daniel Alarms
Date of creation	21-03-2019
Reviewed by	Akriti Pathania
Date of review	24-03-2019
Executed by	Daniel Alarms
Date of execution	24-03-2019

Table E.34: Test Cases for Messenger Page

TID	27
Test Name	Using Messenger Page
Test Case	Display most recent message on person's tab
Pre-Condition	User and another player has an account and a message is sent
Test Steps	Click on messenger icon on feed; click on a tab; Type message; Send message
Test Data	"This is a very very very very very very very very very very very very very very very very very long test message"
Expected Results	Every contact's profile tab should show the latest message sent, which is cut short
Post Condition	Contact's profile tab displays "This is a very very very..." along with the timestamp
Actual Result	Every contact's profile tab shows the latest message sent, which is cut short
Status	COMPLETE

Table E.35: Test Cases for Messenger Page

TID	28
Test Name	Using Messenger Page
Test Case	Click on a contact to message
Pre-Condition	User and another player has an account and user has contacts
Test Steps	Click on messenger icon on feed; click on a contact's tab
Test Data	NA
Expected Results	Page should redirect to the corresponding messaging page for that contact
Post Condition	End up in contact's messaging page screen

Actual Result	Redirects to the corresponding messaging page for that contact
Status	COMPLETE

Table E.36: Information on test Cases for Upload Post

Project Name	Skouted
Module Name	Upload Post
Created by	Daniel Alarms
Date of creation	21-03-2019
Reviewed by	Akriti Pathania
Date of review	24-03-2019
Executed by	Daniel Alarms
Date of execution	24-03-2019

Table E.37: Test Cases for Upload Post Page

TID	29
Test Name	Using Upload Post Page
Test Case	Upload video from phone's gallery
Pre-Condition	User has an account
Test Steps	Click the add icon on upload post page; select 'gallery'; Choose video; Click upload
Test Data	Video on phone
Expected Results	Screen should indicate that the video has been successfully uploaded
Post Condition	Video should be displayed on the user's profile page
Actual Result	Screen indicated that the video has been successfully uploaded
Status	COMPLETE

Table E.38: Test Cases for Upload Post Page

TID	30
Test Name	Using Upload Post Page
Test Case	Upload video from phone's camera
Pre-Condition	User has an account
Test Steps	Click the add icon on upload post page; select 'camera'; Record a video; Click upload
Test Data	Camera recorded video
Expected Results	Screen should indicate that the video has been successfully uploaded
Post Condition	Video should be displayed on the user's profile page
Actual Result	Screen indicated that the video has been successfully uploaded
Status	COMPLETE

Appendix F

Questionnaire for over 18's

1. What do you identify as?

   *(Please tick **one** box)*

   ☐ Female

   ☐ Male

   ☐ Prefer not to say

   ☐ Other

2. Which device have you tested the application on?

   *(Please tick **one** box)*

   ☐ Android

   ☐ iOS

3. What is your first reaction to the product?

   *(Please tick **one** box)*

   ☐ Amazing

   ☐ Good

   ☐ Neither good, nor bad

   ☐ Bad

☐ Terrible

4. How visually appealing is our application design?

   *(Please tick **one** box)*

   ☐ Extremely appealing

   ☐ Very appealing

   ☐ appealing

   ☐ Not very appealing

   ☐ Not at all appealing

5. How easy is it to navigate through the application?

   *(Please tick **one** box)*

   ☐ Very easy

   ☐ Easy

   ☐ Neither easy nor difficult

   ☐ Difficult

   ☐ very difficult

6. How easy is it to create an account with *Skouted* ?

   *(Please tick **one** box)*

   ☐ Very easy

   ☐ Easy

   ☐ Neither easy nor difficult

   ☐ Difficult

   ☐ very difficult

7. How easy is it to use the *Skouted* application?

   *(Please tick **one** box)*

   ☐ Very easy

☐ Easy

☐ Neither easy nor difficult

☐ Difficult

☐ very difficult

8. Has the application design been maintained consistently throughout?
   (Please tick all the boxes that apply)

   ☐ Yes

   ☐ Not in the feed page

   ☐ Not in the profiles page

   ☐ Not in the messaging page

   ☐ Not in the search page

   ☐ Not in the upload post page

   ☐ Not in any of the pages

9. Which feature(s) of the application is/ are most valuable to you?
   (Please tick all the boxes that apply)

   ☐ Being able to upload videos

   ☐ Being able to upload match statistics

   ☐ Being able to view other player's profiles

   ☐ Being able to communicate with scouts

   ☐ Being able to communicate with other players

   ☐ All the above

Appendix G

Meeting Minutes

Meeting	Wednesday 23rd January 2019, 12:30pm, MSB2.24
Attendees	Daniel Alarms, Ignacio Borrego Melendez-Valdes, Samuel George Ogden, Akriti Pathania, Harjot Singh, Aliyah Stevens

Agenda:

1. Previous week's tasks update
2. Integrate front-end and back-end of messaging page
3. Plan for the application demo with the customer
4. Tidy up loose end from other pages for customer demo

1 - Previous week's tasks update:

1. **Daniel:** Finished implementing back-end for messaging page
2. **Ignacio and Harjot:** Tidied up loose ends of feed page
3. **Akriti:** Finished implementing front-end of messaging page
4. **Sam:** Working on design for search page
5. **Aliyah:** Working on ethics documentation

2 - Integrate front-end and back-end of messaging page

1. Some issues cropped up when integrating the two ends together

3 - Plan for the application demo with the customer

1. Decided to do a Skype call showcasing the feed page, clip-viewing page, messaging page, upload post page and profile page.
2. Date of demo day: Wednesday 30th January 2019
3. Time of demo: Around 3pm
4. Every team member should be available that at that time, on that day

4 - Tidy up loose end from other pages for customer demo

1. Profile page and clip-viewing page require tidying up

Actions for next Wednesday's meeting:

1. **Daniel and Akriti:** Continue to integrate the back-end and front-end of the messaging page
2. **Ignacio:** Tidy up loose ends of clip-viewing page
3. **Harjot:** Tidy up loose ends of profile page
4. **Sam:** Continue to work on the design for the search page
5. **Aliyah:** continue to work on ethics documentation
6. **Everyone:** Continue planning demo for client

Appendix H

User Manual

H.1 Administrator Manual

H.1.1 Application Overview

The application contains the following microservices:

Name	*Description*
signup	*Microservice to handle user registrations/signups*
media	*Microservice to handle media uploads*
landing-page	*React Landing page for Skouted*
mobile	*React Mobile front-end for Skouted application*
desktop	*React Desktop front-end for Skouted application*
feed	*Microservice to handle activity feed*

H.1.2 Installation and Deployment

Install Docker from https://docker.com and NodeJS from https://nodejs.org.

Begin by installing all the dependencies, by executing the script *./install-deps.sh*. This will run *npm i* for each subdirectory/service in the repo. Run this script again after you pull changes from the codebase to make sure you've kept up-to-date.

To start every microservice simultaneously, run the command *docker-compose up*. Any changes to code will force the affected service(s) to reload.

To start an individual microservice, *docker-compose up [name]*.

To install a package for one of the aforementioned sub-repos, run *npm i [package name]* inside the corresponding directory.

Running *npm run lint* will check your code style and report any errors.

Running *NODE_ENV=test docker-compose up [name]* or *npm test* will run all the tests under *test/* for the folder *name*, and also produce a code coverage report in text, as well as HTML (in the *coverage* folder).

H.1.3 API Documentation

Developers can find the documentation for the *Skouted* APIs in the HTML API documentation files, located under the *api_documentation* folder.

H.2 User Manual

H.2.1 Installation

To install Skouted, users must follow the deployment url and access the *install* page. They will then be presented with the screen in Figure H.1, where they must follow the on-screen instructions.

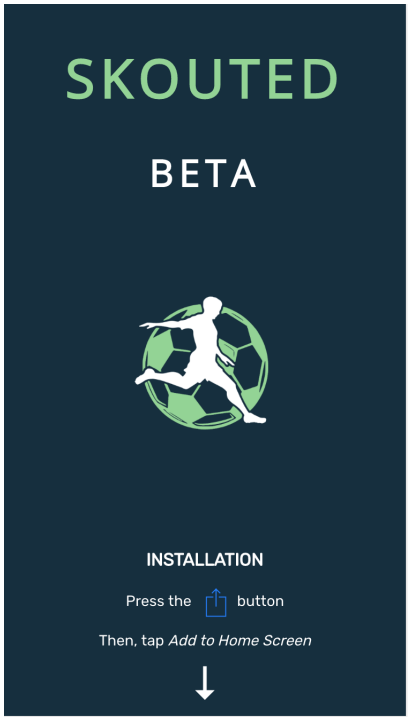


Figure H.1: *Skouted* installation page

H.2.2 Sign-Up

To signup, the user must follow the process presented in figures H.2 and H.3. For the purposes described in the UI designs section, the user must provide all the data in the required fields.

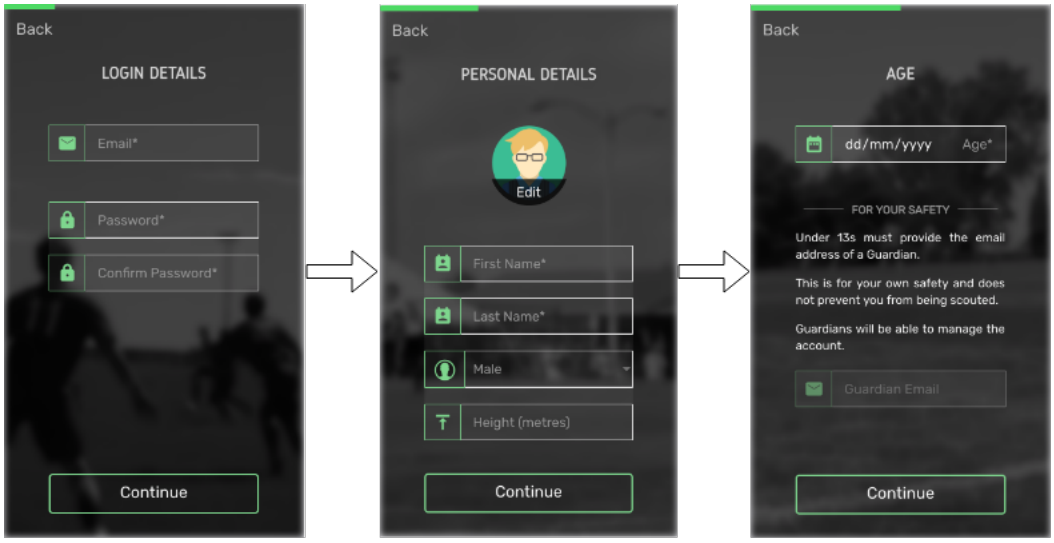


Figure H.2: Sign Up Process Part 1

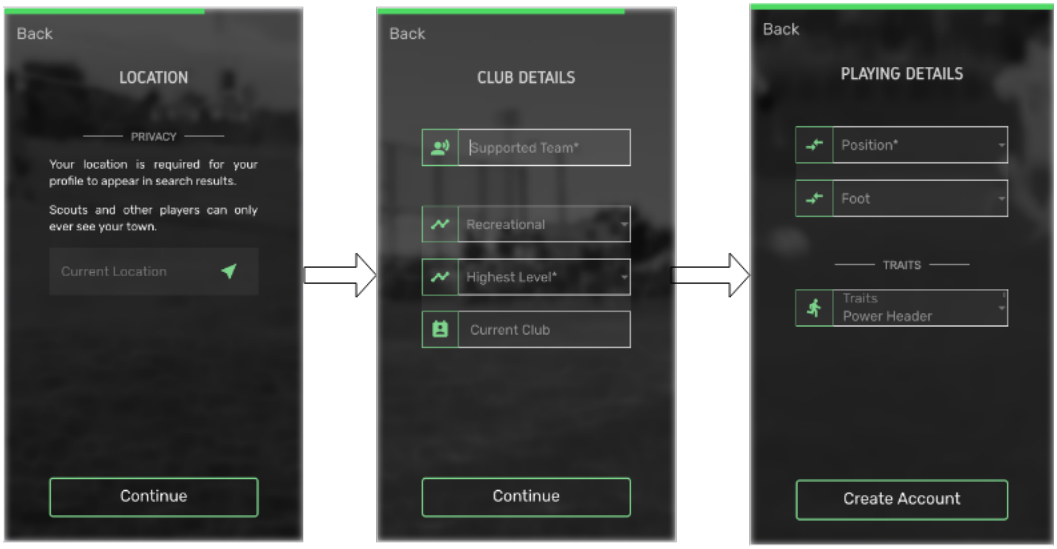


Figure H.3: Sign Up Process Part 2

H.2.3 Login

To login, the user simply needs to access the entry point of the *Skouted* application and enter their credentials.

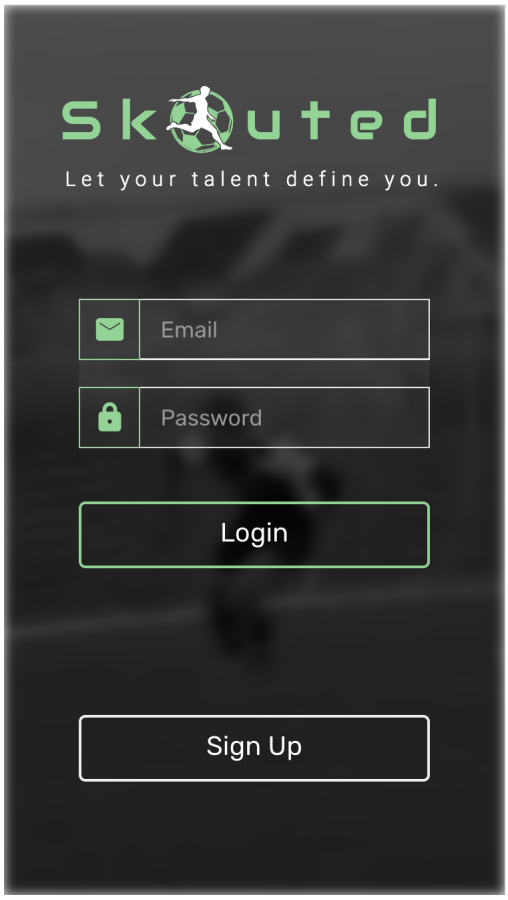


Figure H.4: *Skouted* login

H.2.4 Feed

The feed contains the posts from the user's followees, as well as access to the messaging page. You can like or comment on another player's video by clicking on the thumbs up or speech bubble icon, respectively.

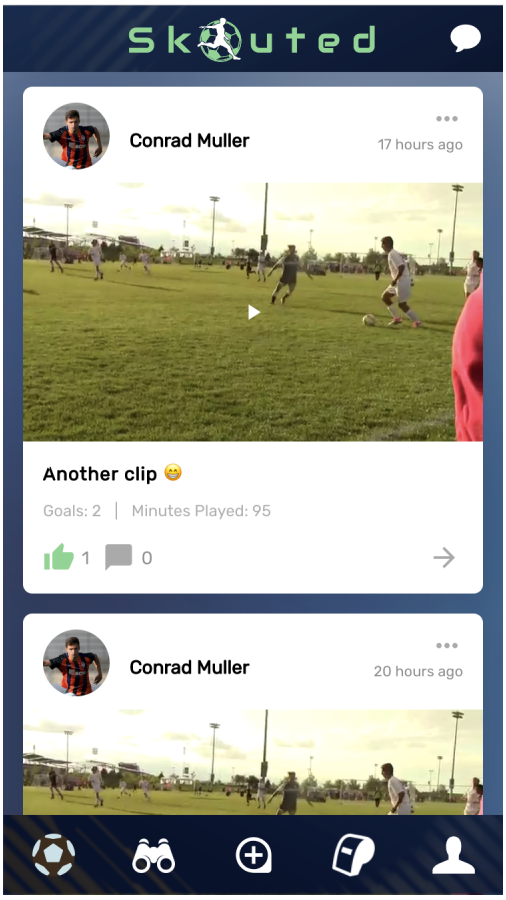


Figure H.5: *Skouted* feed

H.2.5 Search

To search for specific players or scouts, the user must interact with the interface presented in Figure H.6. The user can refine the search through modifying the different filters, subsequently scrolling down and pressing the search button. Additionally, the user can search by name by using the top search bar to type in the desired name.

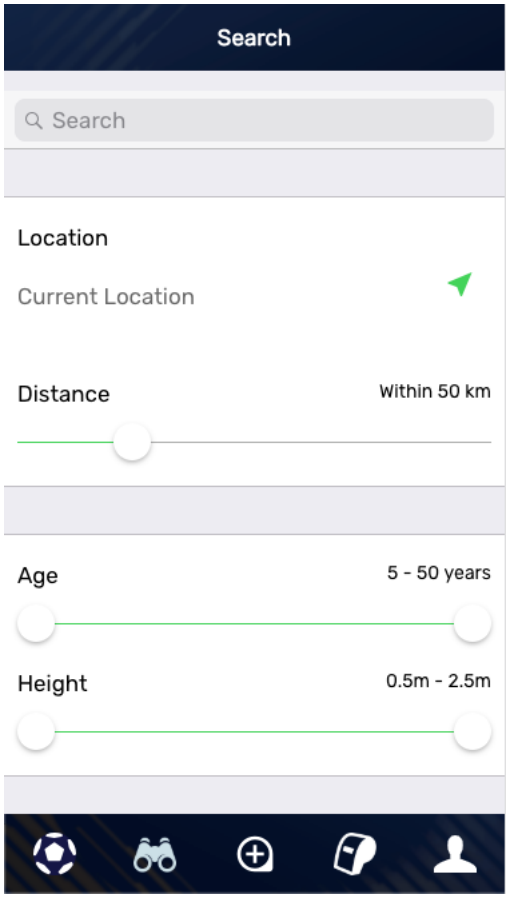


Figure H.6: *Skouted* feed

H.2.6 Messaging

The messaging page can be accessed using the top-right corner of the feed page featuring a speech bubble icon. The user can then message the users through the action flow shown in Figures H.7 and H.8.

Figure H.7: *Skouted* Messaging Page

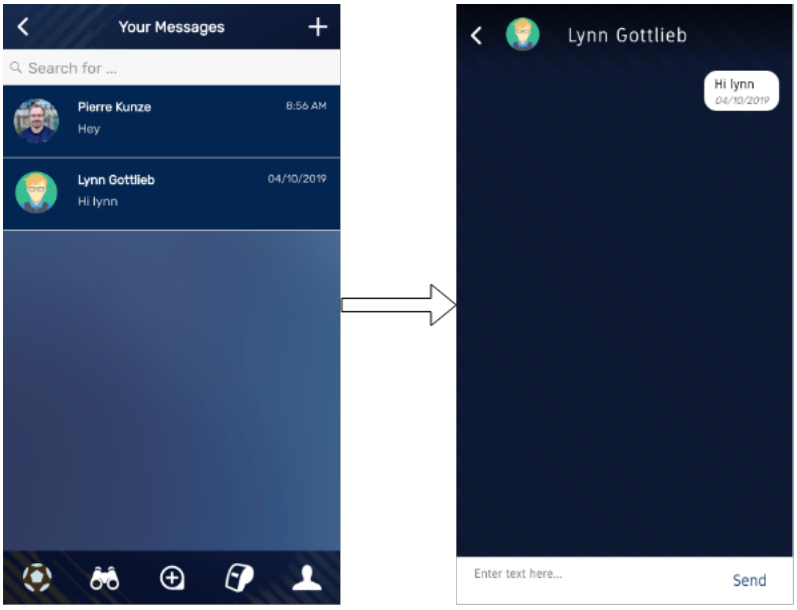


Figure H.8: *Skouted* Messaging Flow

H.2.7 Profile Page

The user icon in the bottom navbar takes you to the profile page. This page is where all your statistical data as a player is displayed, in addition to where all your uploaded video clips are stored. To view a video clip, click on the video, you will be taken to a page where you see your video and like and comment on the video. On the top right corner of your screen you can see a pencil icon, which allows you to edit your player statistics and your profile picture.

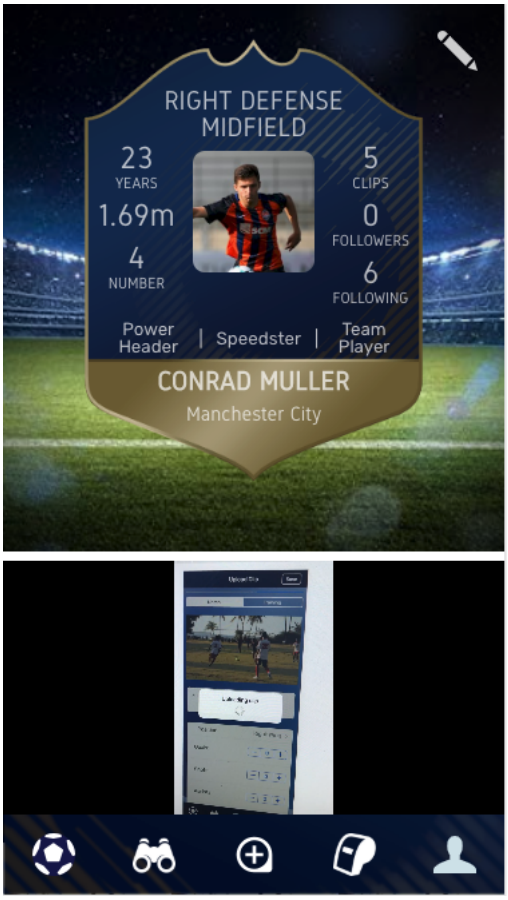


Figure H.9: Player Profile UI

H.2.8 Upload Post Page

The add icon at the bottom navbar takes you to a page where you can upload your posts. This page is where you can upload upload your video clips. If you want to upload a video clip of a match or a training session, then click on the 'Match' or 'Training' tab respectively. To upload a video click on the 'Select Video' button and choose between selecting a video from your gallery or recording the video from your phone's camera. Before you save your video, you have the options to give a brief description of your video in the 'Add a description' box, in addition to providing your performance statistics in the video.

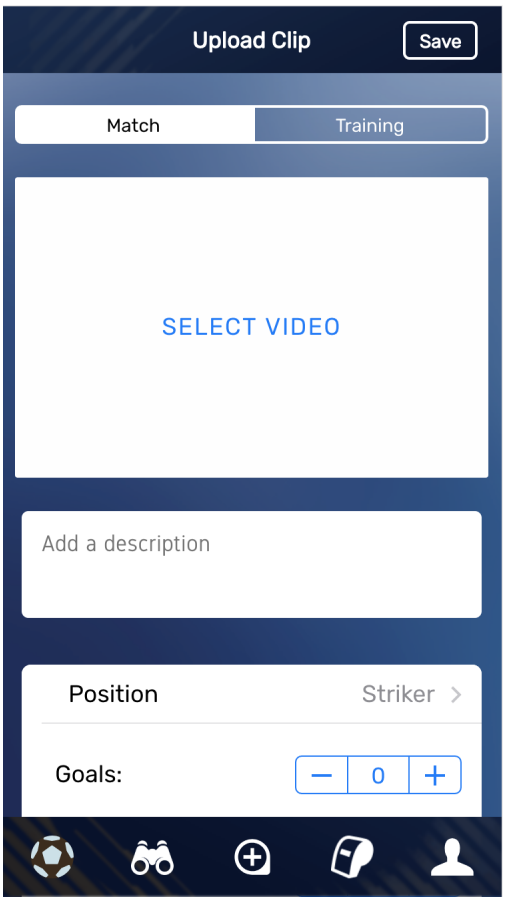


Figure H.10: Upload Post UI Part 1