TeamMate Finder

By
Giannis Konstantinidis

Project unit: M24739
Supervisor: Elisavet Andrikopoulou

May 2023

Acknowledgements

I'd like to thank my supervisor, Elisavet Andrikopoulou, for her constant support, guidance, and expertise during my dissertation. Her extensive knowledge and constructive feedback were invaluable in developing my research and steering me in the proper direction. I appreciate her commitment, patience, and availability in going above and beyond to ensure my success. I'd also like to thank my family and friends, who have continually inspired and motivated me.

Abstract

This project aims to develop an Android application that helps users find compatible teammates to game with. With the COVID-19 pandemic forcing people to stay home, many individuals have become distant from their friends and have lost contact with them, resulting in many people having no one to enjoy gaming with. The proposed Android application seeks to solve this issue by providing a platform for users to connect with like-minded individuals with similar gaming interests. The application will incorporate various features, such as personalized profiles, matchmaking, and searching algorithms, to help users find potential teammates. By providing an accessible and user-friendly platform for gamers to connect, this application has the potential to enhance social connections, promote mental health and well-being, and improve overall gaming experiences.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

This chapter initiates the project and explains its basis. The key aims and objectives of the project are also presented along with a general overview of the report's structure.

1.1 Project Basis

This project aims to address the issue of social isolation and limited social connections among gamers caused by various reasons such as COVID-19, mental health issues, shyness and social awkwardness (Kowal et al., 2021). Many individuals struggle to find compatible gaming partners due to the lack of in-person interactions or difficulty in expressing their interests and preferences. The proposed Android application seeks to solve this problem by providing a user-friendly platform that enables users to connect with like-minded individuals with similar gaming interests. The application will offer personalized profiles, matchmaking, and searching algorithms to facilitate the process of finding compatible gaming partners. By creating a platform that promotes social connections and fosters a sense of community among gamers, this project has the potential to enhance mental health and well-being, promote social interaction, and improve overall gaming experiences for users. This project is significant because it addresses a crucial problem faced by the gaming community and aims to enhance the well-being of users through improved social connections and experiences.

1.2 Project aim and objectives

The aim of this project was to develop a platform where individuals could find a compatible teammate for gaming and ideally, establish strong real-life friendships. In order to accomplish the project's aim effectively, it is necessary to fulfill the following set of objectives:

- Investigate various methodologies and practices to identify the optimal approach for creating the application.
- Collect and extract requirements by utilizing appropriate methods, such as conducting research and a survey.
- Research similar applications.
- Implement and test the application to verify that it satisfies the specified requirements gathered.
- Assess the achievements and shortcomings of the project and identify any modifications that may need to be implemented in the future.

1.3 Project constraints

The project is faced with multiple constraints that require careful consideration for successful completion while delivering a high-quality product within the given timeline and budget. Firstly, the project's timeline is limited to 8 months, which demands efficient resource management to meet all requirements on time. Secondly, insufficient clarity exists regarding which requirements to implement in case of a low survey response rate, creating uncertainty and potential project delays.

Thirdly, the budget is constrained, no available funds for additional resources to accelerate the project's completion. Fourthly, the project's potential for a faster and better-quality final product is achievable if completed by a team, highlighting the importance of collaboration and teamwork in achieving project goals.

Finally, the project's feasibility is uncertain due to all of the constraints mentioned above.Addressing these constraints is essential for the project's success.

1.4 Project deliverables

The final report, android application, and source code are the deliverables for this project. The final report documents the entire software development process, including literature review, methodology, requirements gathering, design, development, testing, and implementation and evaluation, along with the justification of every choice made.

The Android application along with the source code is the primary software output of the project and should meet all the requirements outlined in the final report. It should be user-friendly, functional, and meet the needs of the end-users. The application should be tested thoroughly to ensure it functions as intended and is free of any bugs or errors.

1.5 Project Approach

The project approach comprises several key steps that are essential for the successful development of this project. First, background research and analysis of similar existing applications should be conducted to gain a better understanding of the market and identify those applications' advantages and disadvantages. A survey should then be conducted to gather requirements and feedback from potential end-users. Research and survey results should be utilized to elicit a clear and concise set of requirements for the application. The next step is to choose a suitable database and platform to build the application upon, ensuring that it meets the requirements and is scalable for future enhancements. Finally, research should be conducted to check if any other technologies or tools are required to successfully implement the requirements gathered, ensuring that the application is developed using the latest and most appropriate technology stack. By following this project approach, it can be ensured that the resulting application meets the needs and

expectations of the end-users and is developed in a systematic and efficient manner.

1.6 Legal, ethical, professional, and social issues

The ethical form has been filled and submitted and can be seen in Appendix C.

1.7 The report

In Chapter 2, a literature review will be presented that is based on research and analysis of existing gamer finder applications and their implemented features.

In Chapter 3, various development models will be discussed, including their advantages and disadvantages. Then the appropriate model will be chosen, along with an explanation of the reasons behind the decision.

In Chapter 4, an overview of the user requirements gathering process will be provided, which involved researching existing similar applications and conducting a survey. The collected requirements will be presented and ranked based on their importance.

In Chapter 5, the construction of the system will be discussed, which includes system architecture design, use case design for user interaction, and graphical user interface (GUI) design.

In Chapter 6, the implementation and testing of the application will be discussed in detail, based on the previously gathered user requirements and designs. The chapter will provide an in-depth understanding of the development and testing process of the application.

In Chapter 7, a review and reflection of the application development process will be provided, along with an assessment of the developed system and a comparison of the final outcome with the user requirements. The chapter will also evaluate the methodology utilized for the development of the application.

In Chapter 8, the project concludes and offers a glimpse into possible future improvements.

Chapter 2: Literature Review

This literature review aims to analyze some of the most popular applications similar to the one being developed, highlighting their unique features and distinguishing characteristics. A thorough analysis will be conducted to identify key features and provide insights into the strengths and weaknesses of each application, this is important because it will help gain insights on UI, UX, features, and performance, identify opportunities for improvement, and avoid potential design flaws.The features gathered in combination with the survey results will serve as the application’s requirements.

Gaming can be an effective way for individuals who struggle with social awkwardness or mental health issues to find friends and improve their well-being. Studies have shown that engaging in collaborative gameplay with others can promote social interaction and reduce feelings of loneliness and isolation, providing a sense of belonging and support (Perry et al., 2018). Moreover, gaming has been found to have therapeutic benefits, including alleviating symptoms of depression and anxiety by filling a void in the patient's life and providing a healthy outlet for stress relief if gaming with the appropriate team (Prochnow et al., 2021). Overall, gaming provides a unique opportunity for individuals to find social connections and improve their mental health and well-being.

2.1 Android vs IOS

Prior to the initiation of the app development stage, research was conducted to determine the appropriate operating system for the app. The market share held by each operating system was the key factor in this decision-making process. It was discovered through research that Android, with a market share of approximately 70%(refer to Figure 2.1), is the most widely used mobile operating system globally. As a result, it was concluded that creating the app on Android would be more accessible and could benefit more people. (Turner, 2023).

Year	Android (%)	iOS (%)
2023	71.65	27.71
2022	71.47	27.85
2021	71.89	27.34
2020	73.06	26.28
2019	75.47	22.71
2018	75.45	20.47
2017	72.63	19.65
2016	69.11	19.29
2015	64.2	20.2
2014	53.65	23.95
2013	39.21	24.03
2012	27.41	24.04
2011	19	22.29
2010	8.82	25.48
2009	2.41	34.01

Figure 2.1 Android vs IOS(Turner, 2023)

2.2 Tech Review

It would be beneficial for the development process of the application to analyze some of the most popular existing applications that aim to solve the same problem as the one this project is attempting to address. This analysis would help in identifying key features and would allow for a more comprehensive understanding of user preferences, leading to the creation of an innovative and competitive application. In order to find those similar applications google play was used to download the ones with the most downloads and the highest rating.

2.2.1 GamerLink LFG: Teams & Friends

GamerLink Inc (GamerLink Inc., 2014) introduced a free application called GamerLink in 2014, which is classified as a social networking app available on both Google Play Store and Apple App Store. With over 500k downloads, the app has received a 4.4-star rating and requires users to be at least 12 years old. The app's primary objective is to enable users to "find and meet gaming friends" (GamerLink Inc., 2014). To achieve this objective, the app includes various essential features such as login, in-app messaging, profile creation, and search functionality.

After using the application for a while, certain areas where it was lacking were identified. Firstly, the application is primarily blog post-based, meaning that users create a post describing the game they're playing and the number of players they need. However, when using the search feature, the application only searches for blog posts and not actual users, making it impossible to search for users directly. Another limitation noticed was the lack of specific skill criteria for each game. The same criteria - beginner, experienced, veteran, and pro - were used for all games, making it less objective and less helpful for gamers who require more specific criteria. These limitations helped understand why the application failed to solve the problem that the author is trying to tackle.

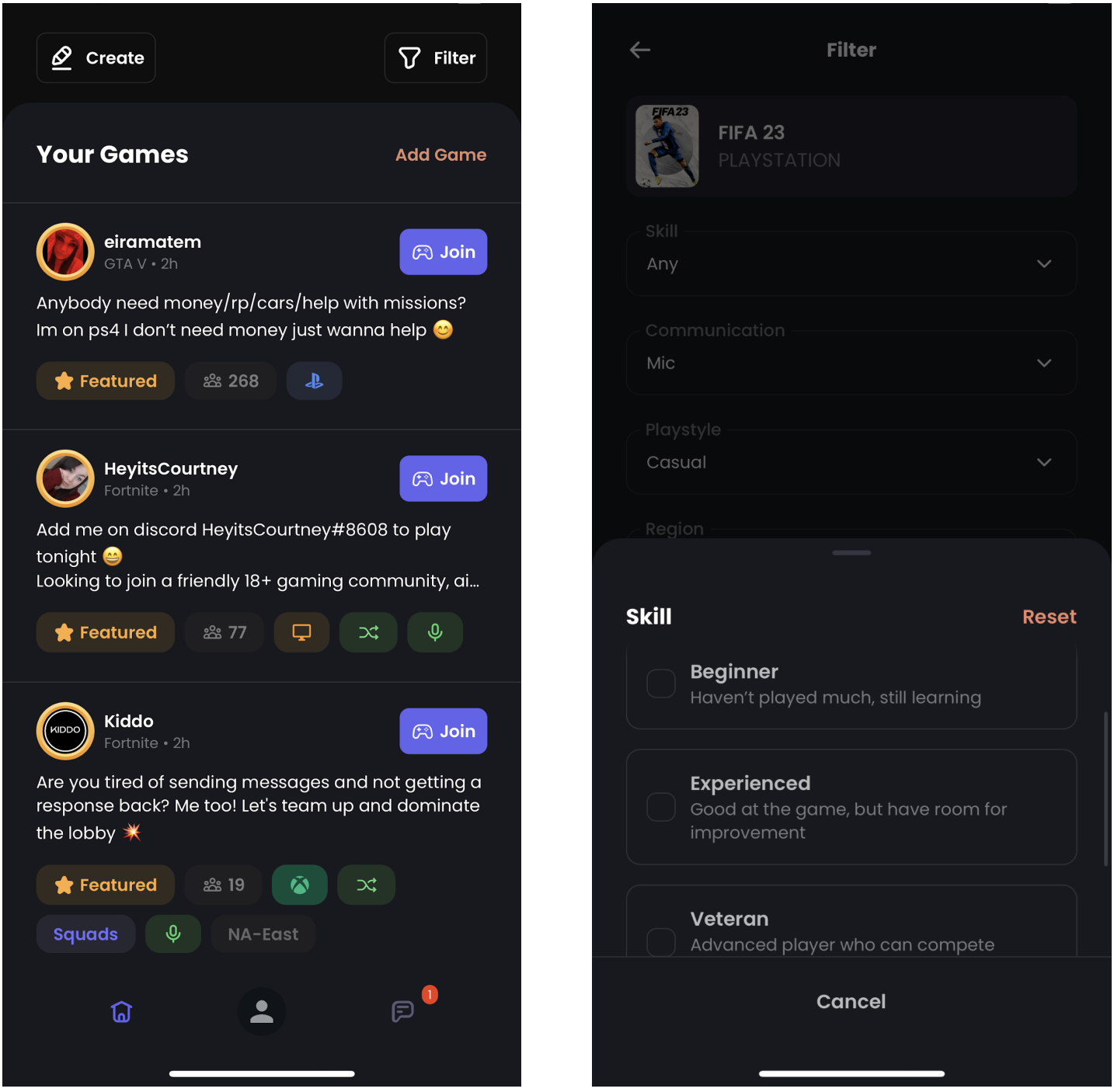


Figure 2.2: (GamerLink Inc., 2014)

2.2.2 Plink: Team up, Chat & Play

The Plink(DogApps, 2018) application, is available on the Android Play Store and Apple App Store, and has over 5 million downloads and a 3-star rating. The app's goal is to assist gamers in finding compatible teammates according to their specific needs. It only offers a console login feature that allows users to log in using their existing accounts on Playstation, Xbox, Nintendo, PC, or mobile by implementing this kind of log in the application gains access to the user's console account, including all their stats and games ever played. Additionally, Plink's features include in-app messaging, detailed in-game statistics, personalized matchmaking, and gender preferences, making it stand out from other apps in its category. The app's unique personalized matchmaking feature ensures that users are matched with compatible teammates based on their gaming style, preferences, and skill level.

Upon seeing the features offered by this application, the author was left to ponder why the current problem of finding compatible gaming teammates still exists. However, after using the app for a day, it became clear why it has failed to achieve its intended purpose. The primary reason for this is the lack of transparency, as the app fails to inform users that it is not free until they log in and provide access to their personal data. Furthermore, the pricing of the application is rather steep, at €5.99 per week, which may prove to be a significant expense for teenagers. Another contributing factor to the app's failure is the lack of privacy it offers to users. This is largely due to the fact that the app only allows users to log in using their console accounts, which grants the app unrestricted access to all of their data without the user having any control over it.

Despite these limitations, this application has the potential to solve the problem of finding compatible gaming teammates if the aforementioned issues are resolved.

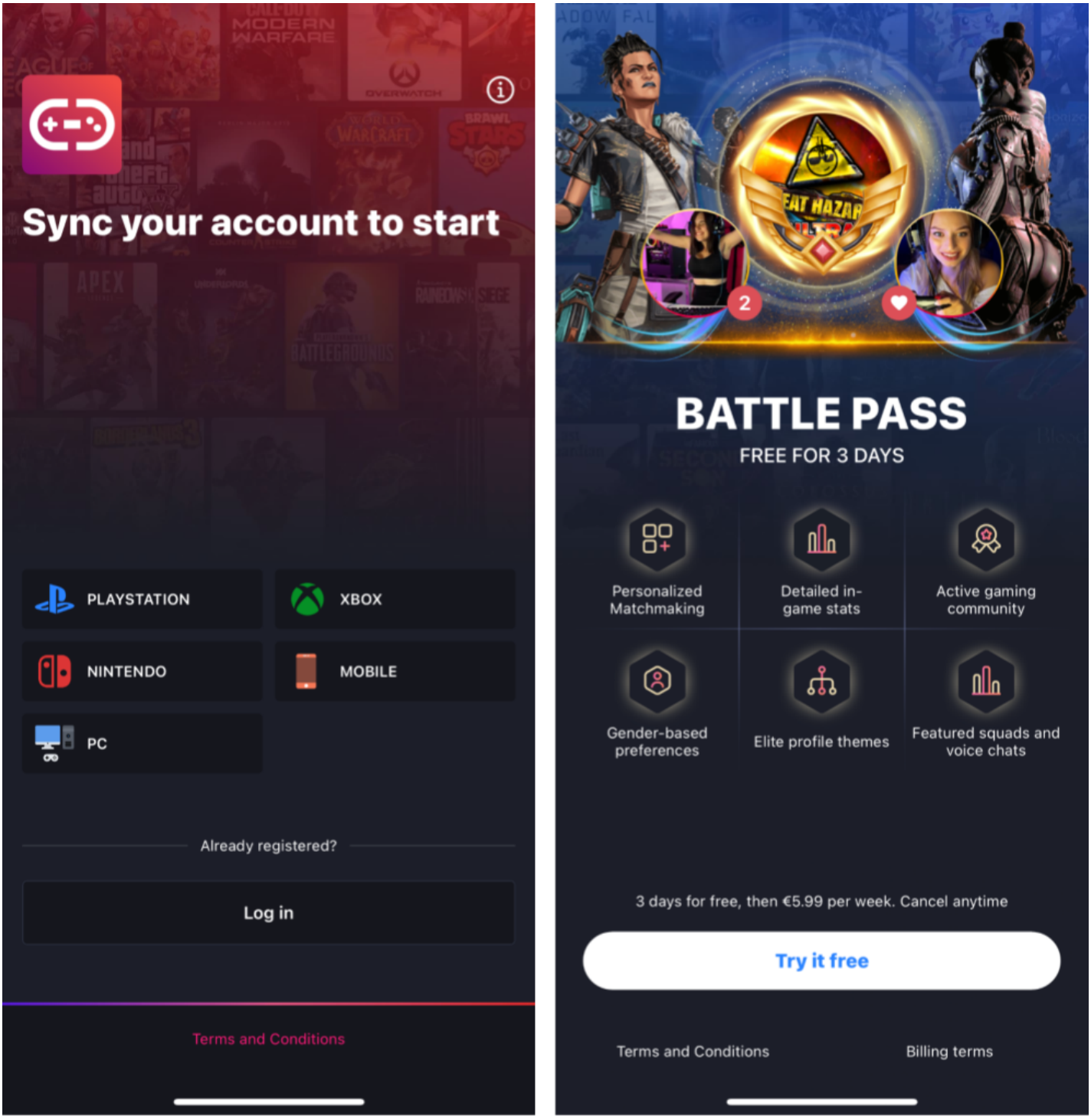


Figure 2.3:(DogApps, 2018)

2.2.3 eBlitz - Find Gaming Friends

The free eBlitz application (eBlitz Ltd, 2021), which is designed to facilitate the process of finding compatible teammates for gamers, is currently available on both the Android Play Store and the Apple app store. The application has received over 100k downloads and boasts a 4.5-star rating. To accomplish its primary objective, the application employs several features, including a Tinder-like feature (Tinder, 2013) that allows users to swipe left or right to match or skip other players, as well as in-app messaging, partner rating, Discord voice chat integration, login feature, and profile.

Upon thorough use of the application, certain limitations were identified that revealed why this application was unable to address the gap in the market and solve the problem of finding suitable teammates for gaming. Firstly, the application automatically recommends users instead of allowing the user to select criteria for the search. This means that users have no control over who shows up in the search and what game they want to play, making it less effective in finding compatible matches. Additionally, the application's game ranking criteria for users is very generic as users can only choose if they want to play competitively or not. There are no specific ranking criteria for each game, which is a significant drawback for the majority of gamers.

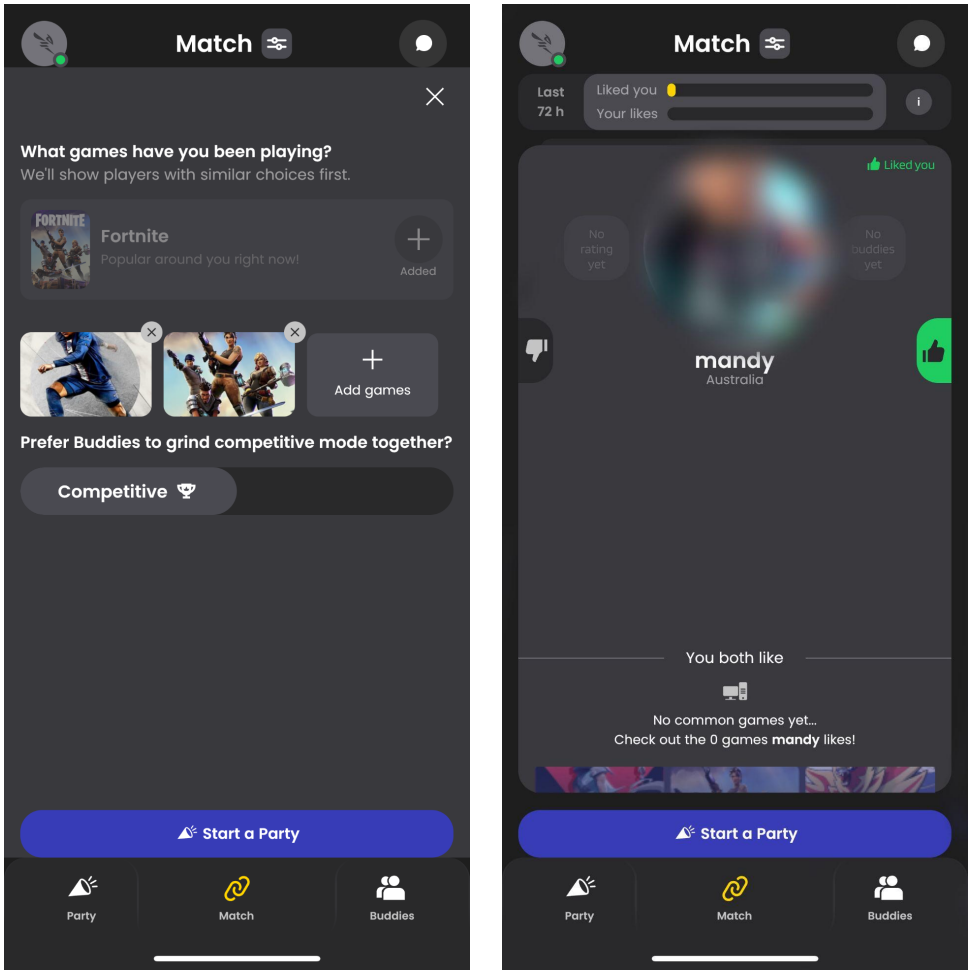


Figure 2.4: (eBlitz Ltd, 2021)

2.2.4 Epal: Play games, Meet friends

Epal (Epal Inc, 2020) is a well-known application categorized under the social networking section of both the Android Play Store and Apple app store. The application aims to assist users in meeting friends worldwide through playing video games. Unlike the previous applications reviewed, Epal has some unique features, such as a blog post feature, video posting feature, image posting feature, and live streaming feature, in addition to the usual login, profile creation, search, voice call, and in-app messaging features. However, these additional features do not contribute in any way towards addressing the problem of finding compatible teammates.

After conducting a comprehensive analysis of Epal (Epal Inc, 2020), it was determined that the app's primary objective is not to aid gamers in locating compatible teammates. Instead, it serves as a social media tool, as evidenced by its emphasis on features such as blog, video, and image posting, as well as live streaming. However, it neglects crucial features such as login, profile creation, search, voice call, and in-app messaging, which are critical for addressing the issue of finding suitable teammates. Unfortunately, these essential features are underdeveloped in this application. The search feature, in particular, lacks skill criteria or support for all consoles, which is a significant drawback for gamers.

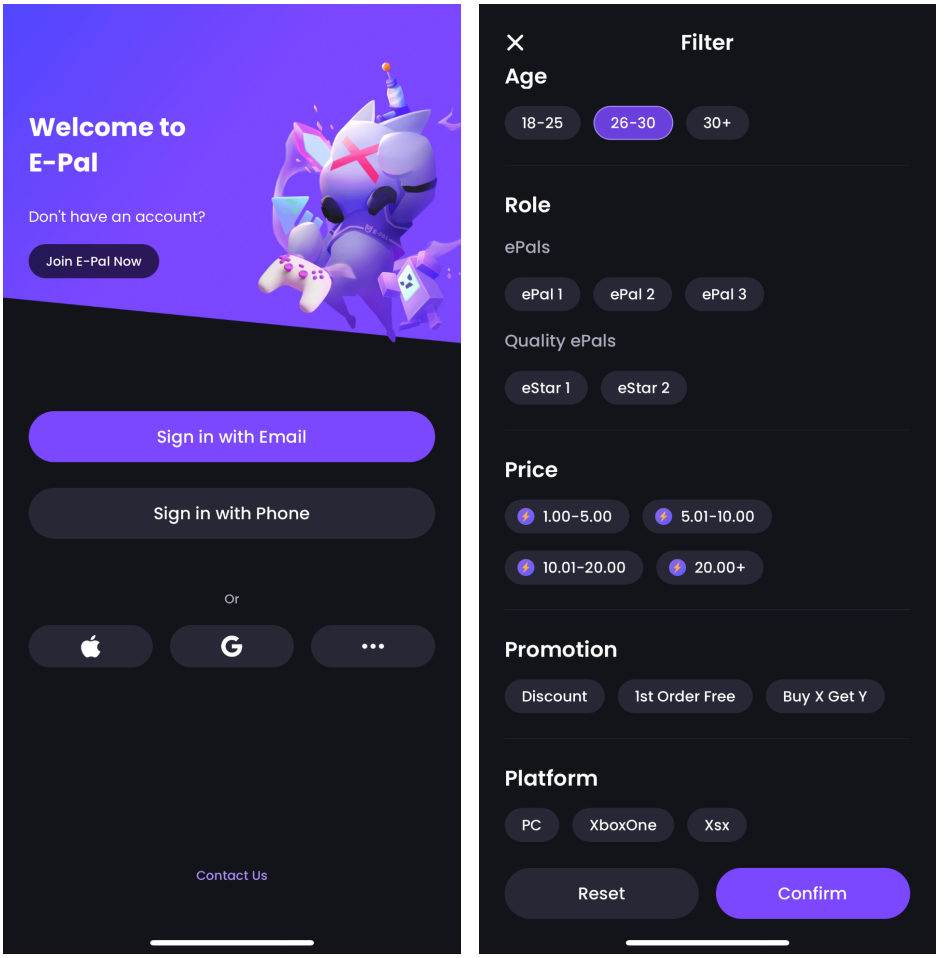


Figure 2.5: (Epal Inc, 2020)

2.2.5 Applications Comparison

The table below (refer to Table 2.1) provides a comprehensive comparison and summary of the features available in each application.

Features	GamerLink (GamerLink Inc, 2014)	Plink(DogApps, 2018)	Eblitz(eBlitz Ltd, 2021)	Epal(Epal Inc,2020)
Login Feature	X	X	X	X
Profile	X	X	X	X
In app messages	X	X	X	X
Skill criteria	X	X	-	-
Tinder(Tinder, 2013) like swipe feature	-	X	X	-
Gamer rating	-	X	-	-
Game stats	X	X	X	-
Free	X	-	-	-
Rating	4.4	2.9	4.5	4.4
Downloads	500k+	5M+	100k+	500k+

Table 2.1: Tech review

After analyzing the table above, it is clear that none of the current applications meet all the required features. Plink(DogApps, 2018) meets all the necessary features except for its high price, which unfortunately limits its accessibility for many users. GamerLink Inc (GamerLink Inc., 2014) is the best available option as it only lacks two essential features and is in second place. EBlitz (eBlitz Ltd, 2021) holds the third position as it is missing three essential features, and it being a paid application is a huge disadvantage. Lastly, Epal (Epal Inc, 2020) is inadequate in many of the critical features, landing it in last place. These results indicate that none of the existing applications can achieve the objective of this project entirely. Therefore, the development of a new application is necessary to fulfill this goal.

Upon careful examination and analysis of numerous applications, it has become apparent that no existing free application fully meets all of the requirements for resolving the problem of finding compatible gamers to play with. This underscores the urgent need for a novel application that can effectively tackle this challenge and provide gamers with a comprehensive and accessible solution. Through the use of a survey and this literature review, the requirements for such an application will be gathered, providing the foundation for the development of an effective and user-friendly solution.

Chapter 3: Methodology

This chapter will explore various agile software methodologies before going into the chosen methodology and the reasoning behind its selection. The reason why only agile methodology was considered for this project is due to its flexibility, iterative nature, and ability to accommodate changes and feedback quickly, which is particularly beneficial when working on a small-scale project with a limited time (Cohen et al., 2004). Choosing the right software development methodology can significantly improve a project's chances of success. It provides clear guidelines for decision-making, facilitates early issue identification and mitigation, and increases adaptability to changing requirements (Geambaşu et al., 2011). This promotes efficient, effective, and high-quality software development practices and reduces the risk of encountering issues later in the project lifecycle. Ultimately, selecting the appropriate methodology ensures that the final software product meets the requirements.

3.1 Agile Methodologies

Agile software practices foster agility by prioritizing active customer collaboration, improved quality, minimal documentation, and expedited time-to-market. By embracing these principles, Agile methodologies allow for changes to be made to project requirements at any point in the development process, resulting in significant time, cost, and resource savings (Ahmed et al., 2010). In contrast, traditional methodologies that do not prioritize agility, such as Waterfall, can lead to delays and cost overruns due to their rigid and sequential nature, which can make it difficult to adapt to changes in project requirements or customer needs (Mccormick, 2012). Even though they place a significant emphasis on teamwork and collaboration, certain Agile methodologies can be adapted to function well on an individual level. One of those approaches is Scrum which can help an individual developer by providing a framework for managing workloads, tracking progress, and identifying obstacles. The use of sprints, daily stand-ups, and retrospectives can help the individual developer stay focused, track progress, and continuously improve their work process. The Scrum framework also allows for flexibility in adapting to changes in project requirements and priorities (L. Wait Rising & Janoff, 2000). Another approach is Kanban which will be discussed in detail later on in this chapter.

3.1.1 Kanban Methodology

Kanban is an Agile methodology that focuses on visualizing the workflow, limiting work in progress, and maximizing efficiency. It originated from the manufacturing industry and has been adapted for software development. Kanban emphasizes on continuous delivery of small, valuable increments of work, with an emphasis on completing one task before starting another. The methodology uses a Kanban board to visually represent the workflow and limit the amount of work in progress.

Tasks are added to the board and moved through different stages, with the goal of completing them as quickly and efficiently as possible. The primary advantages of implementing the Kanban approach are(Ahmad et al., 2013):

- Better software delivery timeframes
- Enhanced software quality
- Better understanding of whole processes
- Greater consistency in delivery
- Reduction in customer reported defects
- Fast feature feedback
- Minimized documentation
- Increased productivity

3.2 Selected Methodology

After looking at multiple different agile methodologies such as Scrum and Kanban the Kanban framework was chosen because it is very flexible, adaptive and allows changes to be made at any point during the development process. It focuses on delivering continuous value, working in small batches, and minimizing waste, all of which are well-suited to individual projects where resources and timelines may be limited such as this project. The visual nature of Kanban, with its use of a board, also makes it easy for an individual developer to manage and prioritize their work. Additionally, Kanban emphasizes on the importance of continuous improvement, making it an effective tool for an individual developer looking to improve their skills and processes over time.

After conducting a survey and literature review to gather requirements, those requirements will be analyzed and broken down into smaller features which will then be placed on the Kanban board. The features will be prioritized based on their importance in the project. By using the Kanban board, the developer will have a clear view of the time frame of features to be developed. The Continuous integration and testing nature of agile methodologies will provide the developer with rapid feedback upon completion of each feature, leading to enhanced product quality and project success.

This chapter provides an overview of the agile development methodologies, with a focus on the Kanban framework, which was chosen for implementation in this project. After extensive research into various methodologies, Kanban was selected for its core principles, visual approach, and ability to be adapted for individual projects. The framework's emphasis on faster software delivery, improved software quality, and increased productivity aligns well with the goals of this project and is expected to play a significant role in its successful completion.

Chapter 4: Requirements Capture

The requirements gathering process is a crucial step in software development that lays the foundation for a successful project. It involves identifying and documenting the specific needs of the project which enables it to successfully achieve its goal. In this chapter, the methods used to gather those requirements will be discussed.

4.1 Requirements Elicitation

The project's requirements were collected from two primary sources to ensure the accuracy and completeness of the requirements. Gathering requirements from a single source can lead to a limited understanding of the project's needs, resulting in incomplete or inaccurate specifications. Initially, the literature review yielded the initial requirements which means that the core features were collected. Next, along with those requirements some more requirements that the author thought to be useful were added in order to guarantee the success of this project and incorporated all of them into a survey, which potential users evaluated and provided feedback on. In order to make the most out of the Kanban software development methodology, after analyzing the results of the survey, the chosen requirements will be prioritized based on their level of importance, ranging from "Must Have", “Should Have”, “Could Have” and "May have in the future". This ranking system indicates the highest to lowest priority of the gathered requirements and is known as the MoSCoW prioritisation (*MoSCoW Prioritization*, 2022).

4.2 Survey Results Summary

To gather the requirements for the application, a survey was created using Google Forms (Google Forms, 2019) and shared with potential users. Google Forms was chosen due to its user-friendly interface, clear response summary, and the author’s previous experience with it. The survey aimed to identify the top three favorite games among users and the significance of each feature. 47 responses were received providing clear and valid results. The number of responses obtained is considered adequate for a small-scale project, especially for an individual project where resources and time are limited. The survey results indicated that the most popular games were Fifa (Electronic Arts, 2022), Pubg (Level Infinite, 2018), and League Of Legends (Riot Games Inc, 2020) (refer to Figure 4.1).

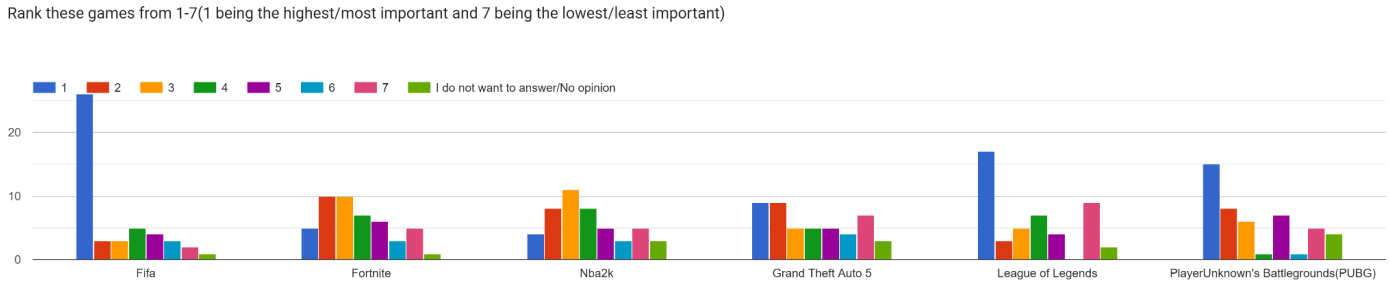


Figure 4.1: Survey game results

Although the survey results for the application's features were not as straightforward as the game results, the author used their judgement to identify the most controversial ones. The survey revealed that out of the 14 potential features, the seven most popular ones which means that they got rated higher by most participants were the Login feature, game selection feature, profile feature, skill criteria search feature, location criteria feature, game stats feature, and gamer rank feature, as indicated in Figure 4.2. These results provided a clear roadmap for implementing the most sought-after features, which would enhance the user experience and increase the application's success.

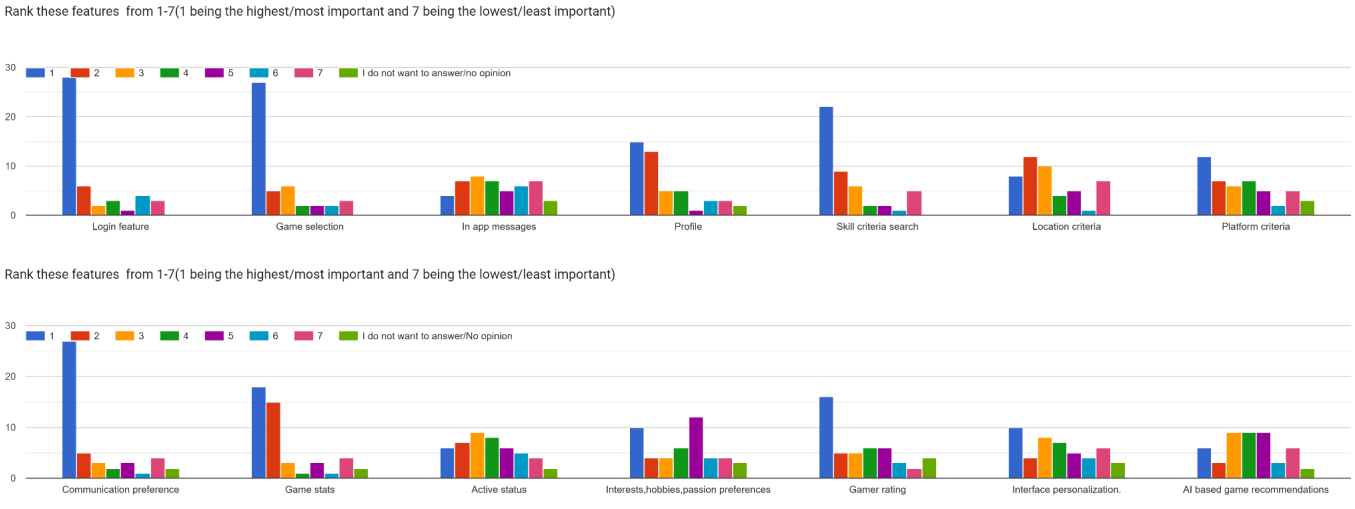


Figure 4.2: Survey feature results

4.3 Requirements

Based on the information obtained from the survey results during the requirement elicitation process, the functional and non-functional requirements for the application were identified. A total of 18 functional requirements and 7 non-functional requirements were collected during this process.

The 25 gathered requirements will be categorized into four different priority levels based on the MoSCoW prioritization method, which includes:

- **Must have:** Requirements of this priority are essential for the succession of this project and cannot be left out, if any of this requirements isn't successfully implemented then this project will be considered a failure.Therefore, these requirements cannot be omitted or overlooked.

- **Should have:** The requirements under this priority are of great significance and their implementation would benefit the entire project. The omission of any of these requirements would lead to a significant decrease in the quality of the project. Hence, it is crucial to implement all of these requirements for optimal project outcomes.

- **Could have:** While the implementation of requirements under this priority is not critical for the application's basic functionality, it would significantly enhance the overall user experience. Therefore, their implementation is highly recommended to improve the overall quality of the application.

- **May have in the future:** The low value addition of the requirements under this priority and the time constraints make it improbable that they will be implemented in the current project, but they could be considered for implementation in the future.

4.3.1 Functional Requirements

Functional requirements describe the desired performance or behavior of a system. They specify the services, tasks, or functions that the system should be capable of performing to fulfill its intended purpose (Malan et al., 2001).

The Functional requirements gathered for this project are:

- **Login/Sign Up:** The user must be able to create an account on the application using their email address and password. Subsequently, they should be able to log in to their account using the same email and password.

- **Authentication:** Upon signing up for the application using their email and password, the application must be able to verify that the provided email is valid and belongs to the user.

- **Password Reset:** If, for any reason, the user forgets their password, they should have the option to retrieve access to their account by utilizing the email address they provided during the sign-up process.
- **Database:** To store all user data related to the application, a remote database must be used by the application.
- **Profile creation and customization:** To provide flexibility and autonomy to users, the application should allow them to create their own profiles and provide an option to update their profiles whenever they desire.
- **Require Discord username:** For user profile creation, there should be a mandatory field for inputting the user's Discord username. This requirement is based on the findings of the survey (refer to Figure 4.3).
- **Avatar Selection:** For user profile creation, users should be given the option to select an avatar as their profile picture.
- **Search functionality based on specific criteria:** To facilitate matching, users must be provided with the ability to search for potential other users using a variety of specified criteria.
- **Friend matching:** After utilizing the search functionality, users should be presented with the option to either select the displayed user and match or skip to the next user who meets the search criteria.
- **Matches list:** To provide a convenient experience for the user, the application should allow them to access a comprehensive list of all the users they have matched after using the search functionality.
- **Friend removing:** Users should have the ability to remove a matched user from their matches list at any time, for any reason.
- **User games:** For the sake of improving the user experience, the application should offer users a list of games that they may be interested in.
- **User Rating:** In order to enhance the user experience, the application should provide users with the ability to rate their experience with other users they have matched with.
- **User Rankings:** The users should have the option to input their ranking into the given list of games provided by the games.
- **Navigation bar:** The implementation of a navigation bar in the application would greatly improve the user experience by providing easy access to different sections of the application without the need to navigate through multiple screens. This results in increased efficiency and saves time for the user. Additionally, it makes the application more user-friendly by simplifying the user interface and reducing confusion.
- **Log out:** The users must be able to log out of their account at any given moment.

- **In-App messages:** After matching with another user, the application should allow the users to communicate with each other by exchanging messages within the app.
- **Activity status:** The application should display the online status of a matched user, allowing users to see when they are currently active on the application.

Requirement	Must Have	Should Have	Could Have	May have in the future
Login/Sign up	X	-	-	-
Authentication	X	-	-	-
Password Reset	-	-	X	-
Database	X	-	-	-
Profile creation and customization	-	X	-	-
Require Discord username	-	-	X	-
Avatar selection	-	-	X	-
Search functionality based on specific criteria	X	-	-	-
Friend matching	X	-	-	-
Matches list	X	-	-	-
Friend removing	X	-	-	-
User games	-	X	-	-
User rating	-	-	-	X
User rankings	-	X	-	-
Navigation Bar	-	X	-	-
Log out	X	-	-	-
In-App messages	-	-	-	X
Activity status	-	-	-	X

Table 4.1: Functional requirements prioritization

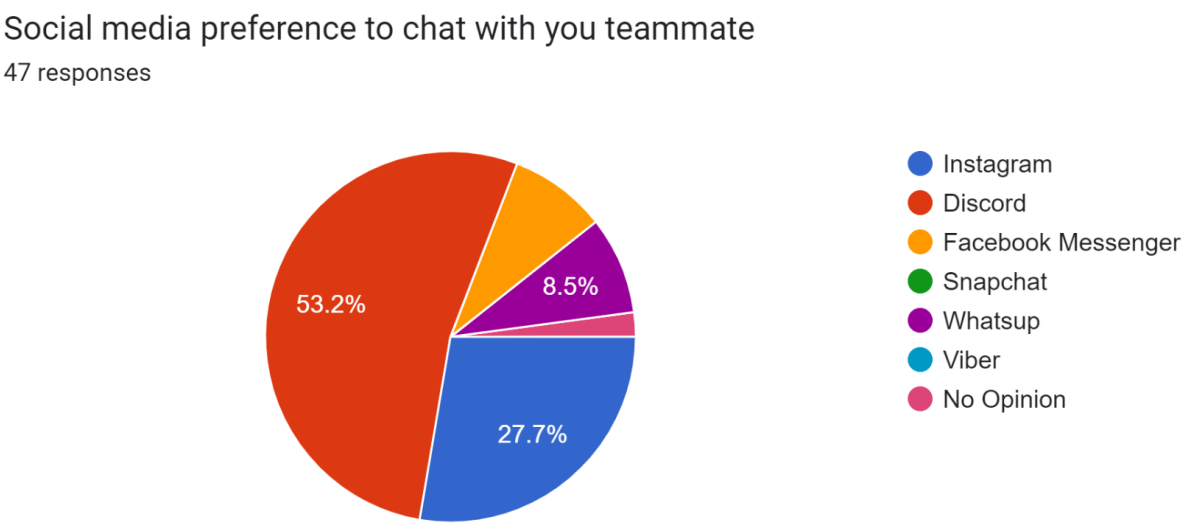


Figure 4.3: Social media preference

4.3.2 Non-functional requirements

While functional requirements define what the software should do, non-functional requirements focus on how the software should perform in terms of quality attributes, rather than just the specific tasks it should accomplish(Chung et al., 2012).

The non-functional requirements gathered for this project are:

- **Android compatibility**: The application should be compatible with most devices and operating systems, to ensure that users can access it from any device they choose.
- **Reliability:** The application should be available and functional at all times, with minimal downtime and effective error handling to prevent crashes and other issues.
- **Scalability:** The application should be able to handle a growing number of users and posts without sacrificing performance or security.
- **Accessibility:** The application should be accessible to users with disabilities, with features such as screen reader compatibility and adjustable font sizes.
- **Performance:** The application should be fast and responsive, with quick loading times for content and minimal latency in messaging and other features.

- **Usability:** The application should be intuitive and easy to use, with clear navigation and user-friendly interfaces.
- **Security:** All user data should be safely stored into the database and unauthorized access should be restricted.

Requirement	Must Have	Should Have	Could Have	May have in the future
Android Compatibility	X	-	-	-
Reliability	X	-	-	-
Scalability	-	-	X	-
Accessibility	X	-	-	-
Performance	X	-	-	-
Usability	-	X	-	-
Security	-	X	-	-

Table 4.2: Non functional requirements prioritization

In this chapter, the methods used to collect requirements for an application were discussed, including a literature review and a survey created using Google Forms (Google Forms, 2019). The survey results provided clear insights into the most sought-after features, enabling the prioritization of the requirements into four categories based on their level of importance. The functional and non-functional requirements for the application were identified, with a total of 25 requirements gathered 18 of which were functional and 7 were non-functional. The prioritization system categorized the requirements into "must have," "should have," "could have," and "may have in the future." The implementation of these requirements will determine the quality of the application and the user experience.

Chapter 5: Design

This chapter will explore the design decisions that will have a significant impact on the outcome of the social networking application to be developed. This application is aimed at helping Android device users find gaming partners. It will be built using Java in Android Studio and will be accessible anytime and anywhere, as long as the user has an internet connection. To enable this, Google Firebase (Firebase, 2012) will be utilized, which will allow users to access their accounts from anywhere and provide key features such as login, real-time database, email authentication, and password reset.

5.1 Architecture

Architecture refers to the high-level design and structure of the system that defines how the various components of the application are organized and interact with each other.

5.1.1 System Architecture

During the process of determining the suitable system architecture for the application, various factors such as functional and non-functional requirements, as well as the overall project goals, were carefully studied. Subsequently, after thorough evaluation, the client-server architectural pattern was selected to encapsulate the application. A client-server system is a software architecture that consists of both a client and a server, where the client sends requests and the server responds to them. This architecture enables inter-process communication by allowing the exchange of data between the client and server, with each performing different functions (Oluwatosin, 2014).

The main advantages of using a client-server architectural model are (Hura, 1995):

- **Scalability:** Client-server architecture is highly scalable as it allows for multiple clients to connect to the same server, and new servers can be added as the load on the system increases.
- **Centralized Management**: In a client-server architecture, the server acts as the central point of management, allowing for easier management of the system and better data sharing.
- **Reliability:** Client-server architecture can provide better reliability as it allows for redundancy and fault tolerance in the system.

While the client-server architecture offers several benefits, it is not free of drawbacks which are (Hura, 1995):

- **Single Point of Failure:** Client-server architecture can be vulnerable to single point of failure if the server fails or is overwhelmed by the number of requests from clients.

- **Latency:** Client-server architecture can have higher latency as the communication between the client and server happens over a network, which can add delay.
- **Cost:** Client-server architecture can be expensive as it requires dedicated server hardware and IT personnel to maintain the server.

Although the client-server architectural model has both advantages and disadvantages, it was chosen for this project due to the use of Google Firebase, a mobile and web application development platform provided by Google. This platform allows us to take advantage of the benefits of client-server architecture while avoiding the cost disadvantages since it is free to use (Firebase, 2012). The use of this architectural model in combination with Google Firebase provides more advantages than disadvantages and ultimately benefits the project the most (refer to Figure 5.1).

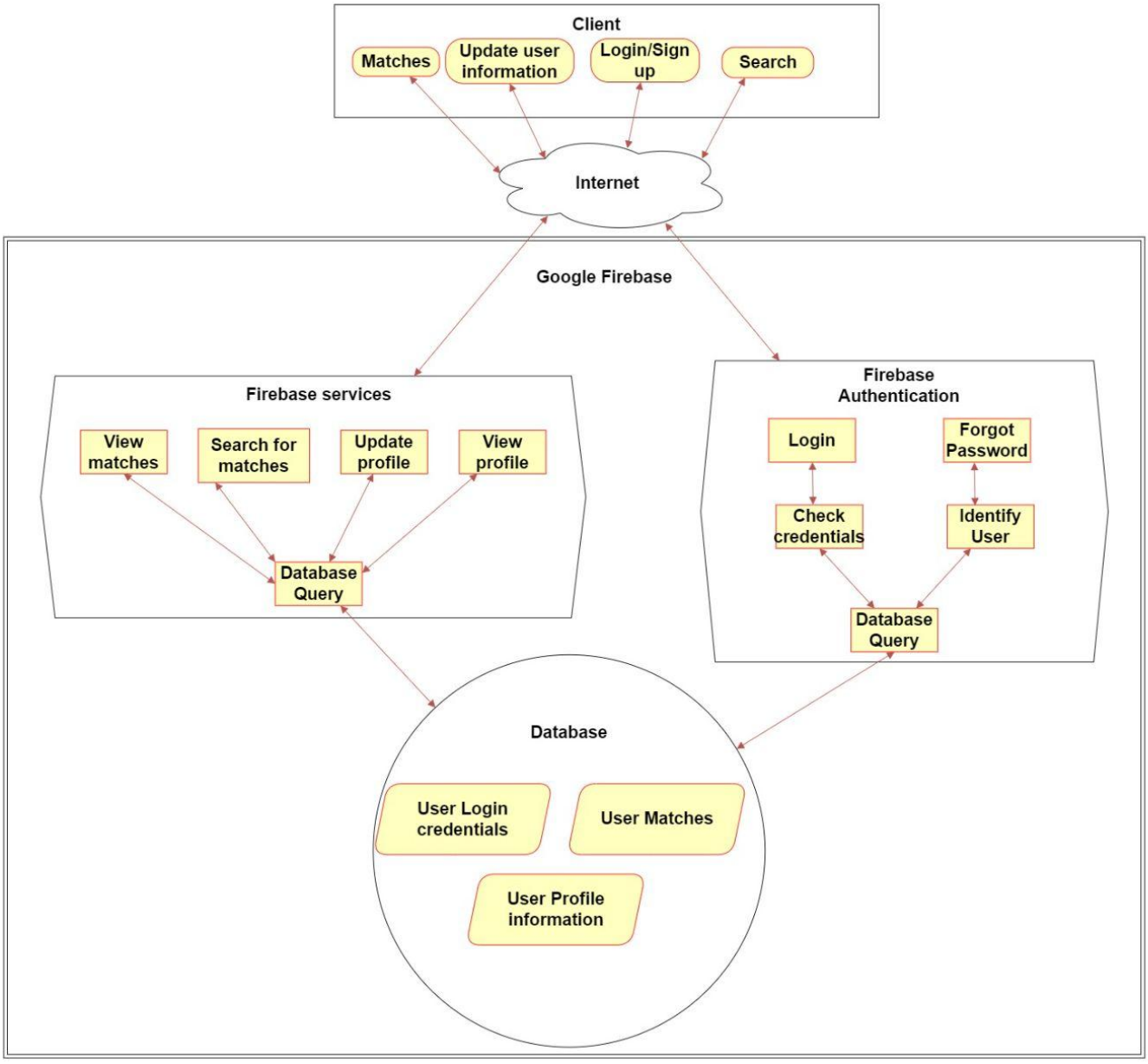


Figure 5.1: System Architecture

5.1.2 Android Architecture

Android Studio is the official integrated development environment (IDE) for developing Android applications (Android Developers, 2014). It is built on the IntelliJ IDEA platform and provides developers with a comprehensive suite of tools for building Android apps, including a code editor, code analysis tools, visual layout editor, emulator, and debugging tools. It can be used to create android applications by the use of either Kotlin or Java programming languages and specifically for this project Java is going to be used. Android apps consist of two essential parts: the front-end user interface (UI), which encompasses activities, fragments, and layout files that define the app's appearance and user interactions, and the back-end logic, comprising classes and objects that manage data storage and retrieval, network requests, and application logic. The front-end UI controls how the app looks and behaves, while the back-end logic interacts with the UI to display data and respond to user input.

5.1.3 Firebase

Google Firebase (Firebase, 2012) is the service that was chosen to support this project, it is a platform that assists developers in building and managing mobile and web applications by providing various tools and services, such as real-time databases, authentication, cloud storage, push notifications, hosting, and machine learning.Firebase's real-time database (Google, 2012) is one of its most popular features, providing developers with a cloud-hosted NoSQL database that can store and sync data in real time across multiple devices. The data in the real-time database is structured as JSON objects which allows for flexible and scalable data storage. Firebase also offers user authentication and authorization services for secure application building and user data protection. These features are essential for efficiently achieving project goals.

5.2 Use cases

Use case diagrams are an essential modeling technique in software development. They visually represent the functional requirements of a system or software application and illustrate the various interactions between the system and its users. Each use case outlines specific actions or tasks that the system is designed to perform. Use case diagrams are helpful in identifying potential problems early in the development process, defining the scope of the system, and better overall understanding. Additionally, they provide a framework for test planning and validation, ensuring that the system is functioning correctly and meeting user needs (refer to Figure 5.2).

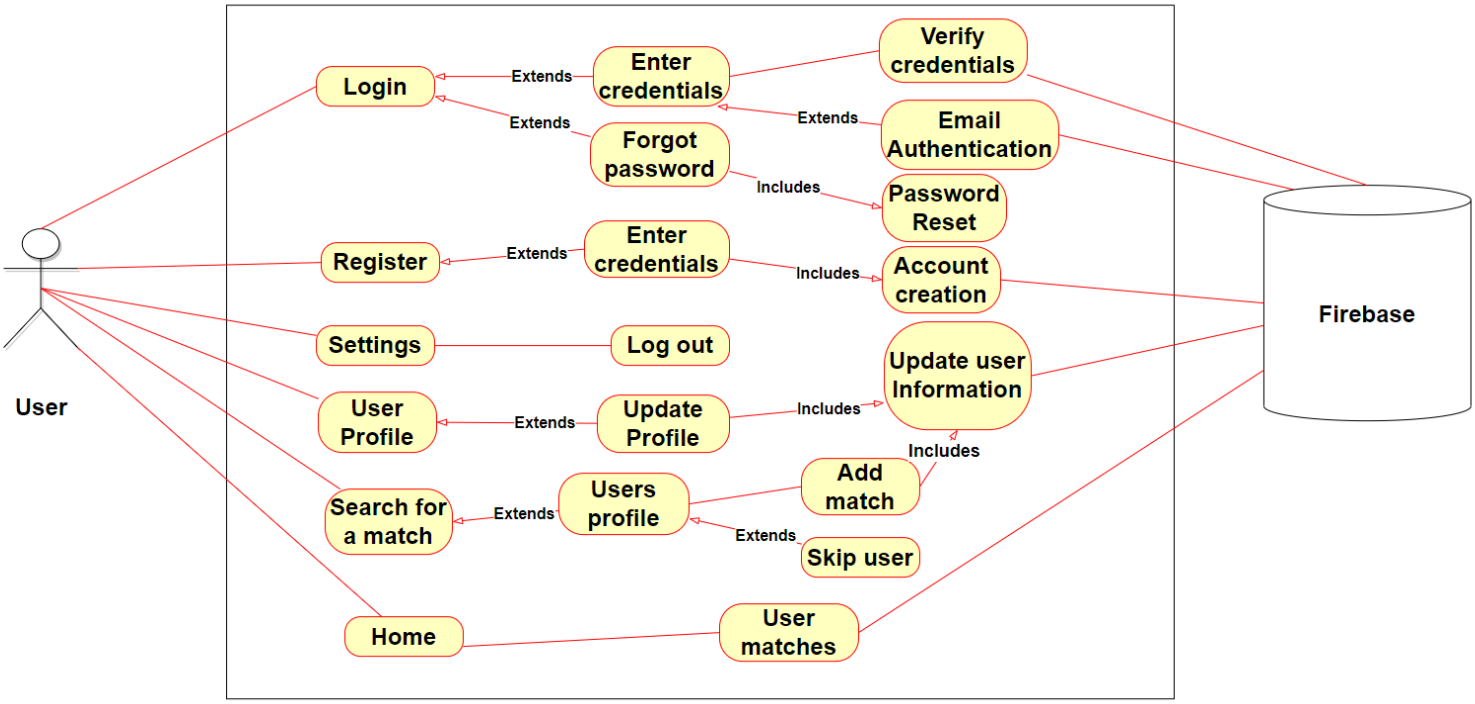


Figure 5.2: Use case Diagram

5.3 User interface designs

A well-designed user interface (UI) is crucial in modern software development and digital product design as it enhances usability, increases engagement, and directly affects user experience (UX), ultimately impacting a product's success. The purpose of UI design is to create an intuitive, aesthetically pleasing, and easy-to-use interface that allows users to interact with a product or system in an efficient and effective way, making it essential for creating engaging, intuitive, and effective user experiences while also reducing the need for user support and training, increasing productivity, and improving overall efficiency. In order to enhance user experience and accessibility, the application will implement proper user interface design principles (Darejeh, 2013). A principle that the application will use is colour contrasting, which means that the colour combinations that will be used will have high contrast black and white or complementary colours to improve legibility and visual appeal but also mitigate any confusion (Peter Zsolt Bodrogi, 2003). Also for this application SP(Scale-independent Pixels) will be used instead of the traditional DP(Density-independent Pixels) for font sizes because SP scales the text size based on the user's device font size preference, which is important for accessibility as it allows users with eyesight problems to increase the font size on their devices, without the application's font size becoming disproportionately large or small. Icons will also be used in addition to text labels to enhance the accessibility and ease of use of the application, especially for users with visual impairments or cognitive disabilities.

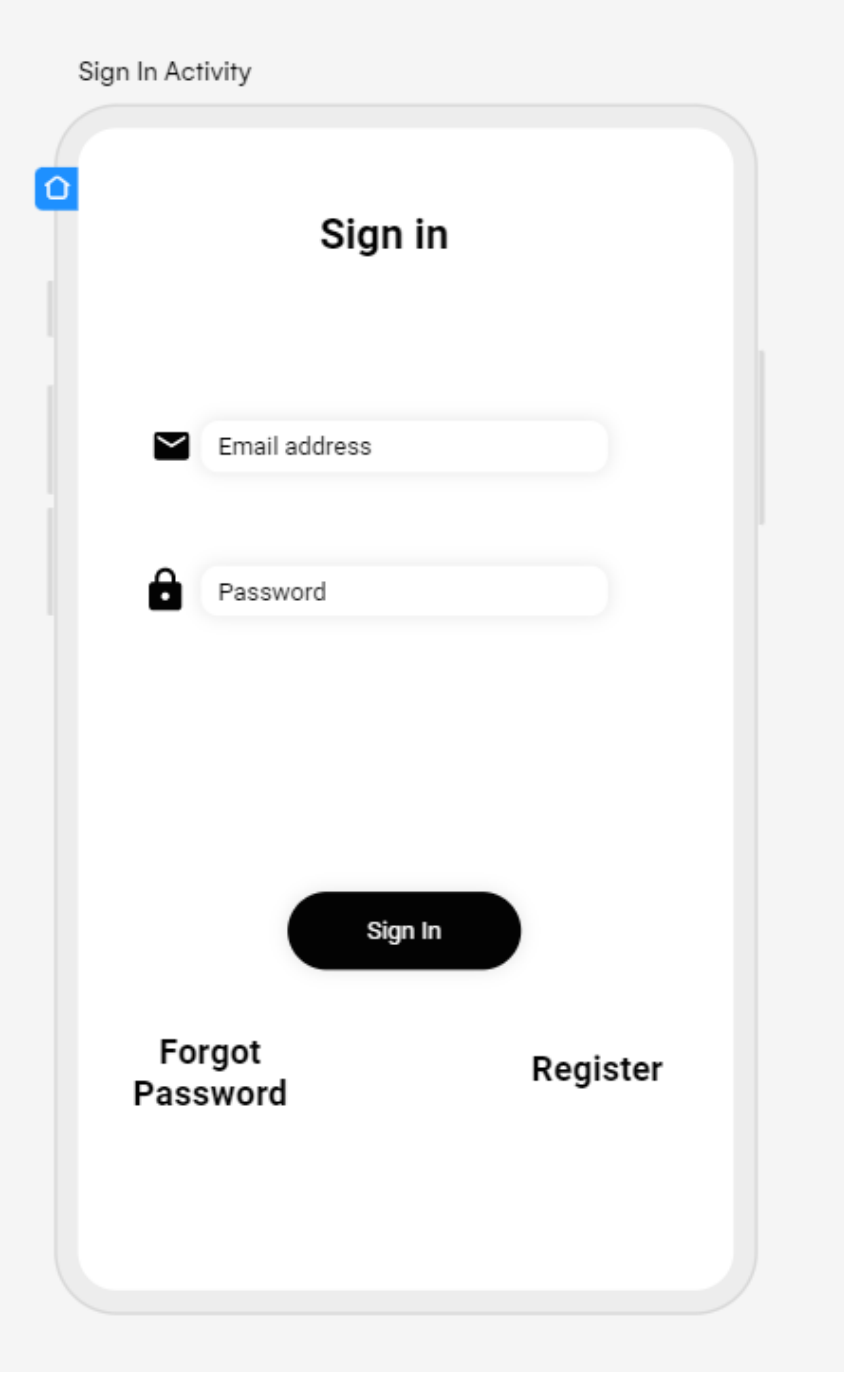


Figure 5.3: Sign in UI Design

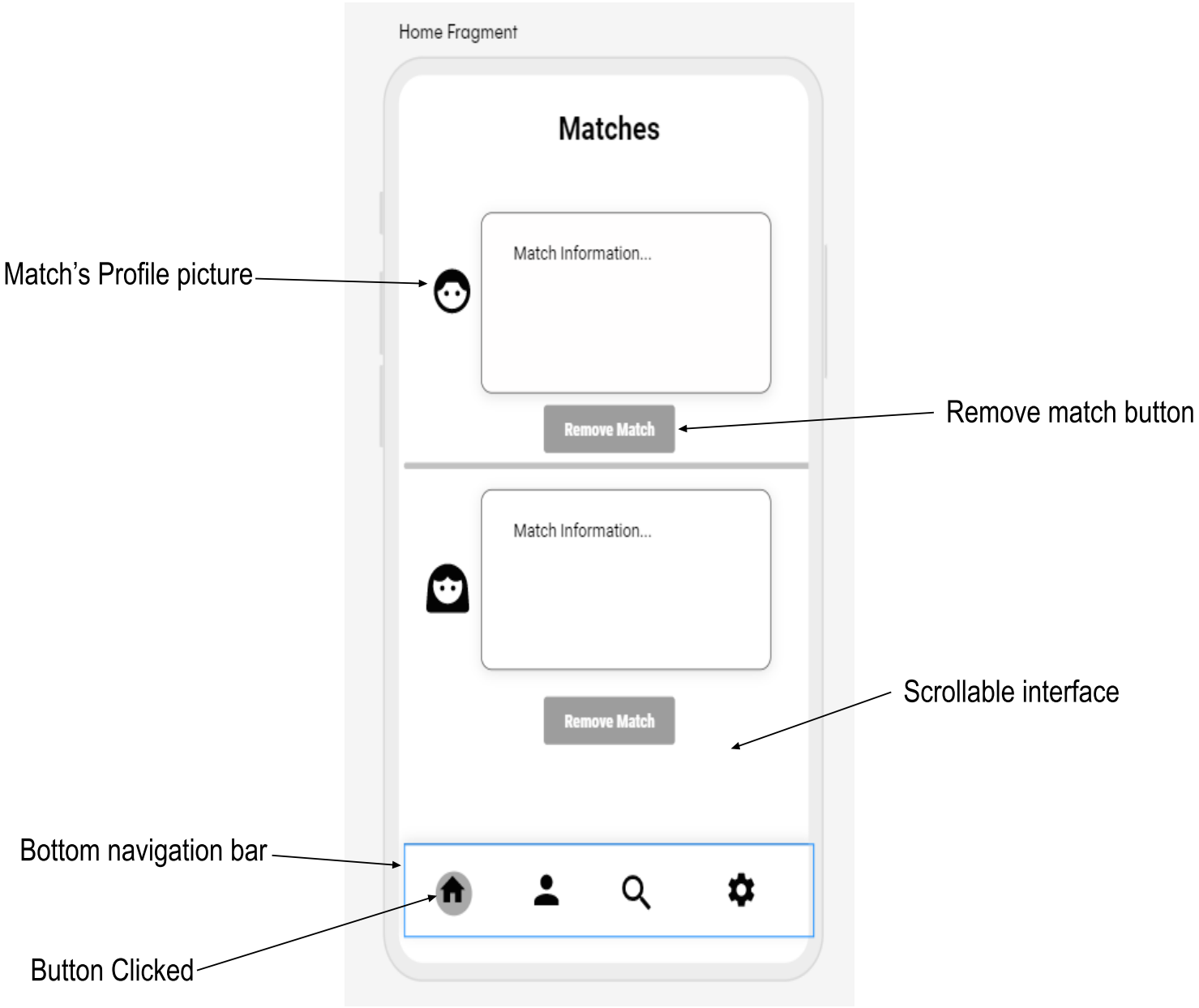


Figure 5.4: Home UI Design

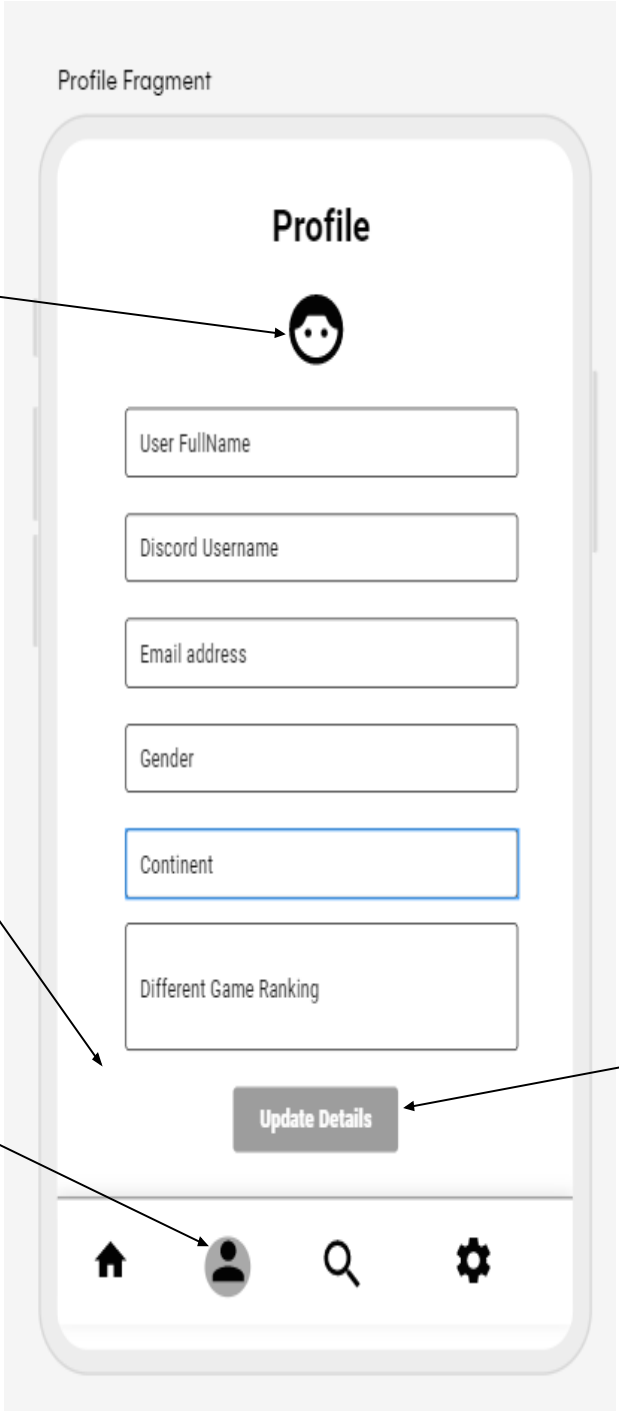


Figure 5.5: Profile UI Design

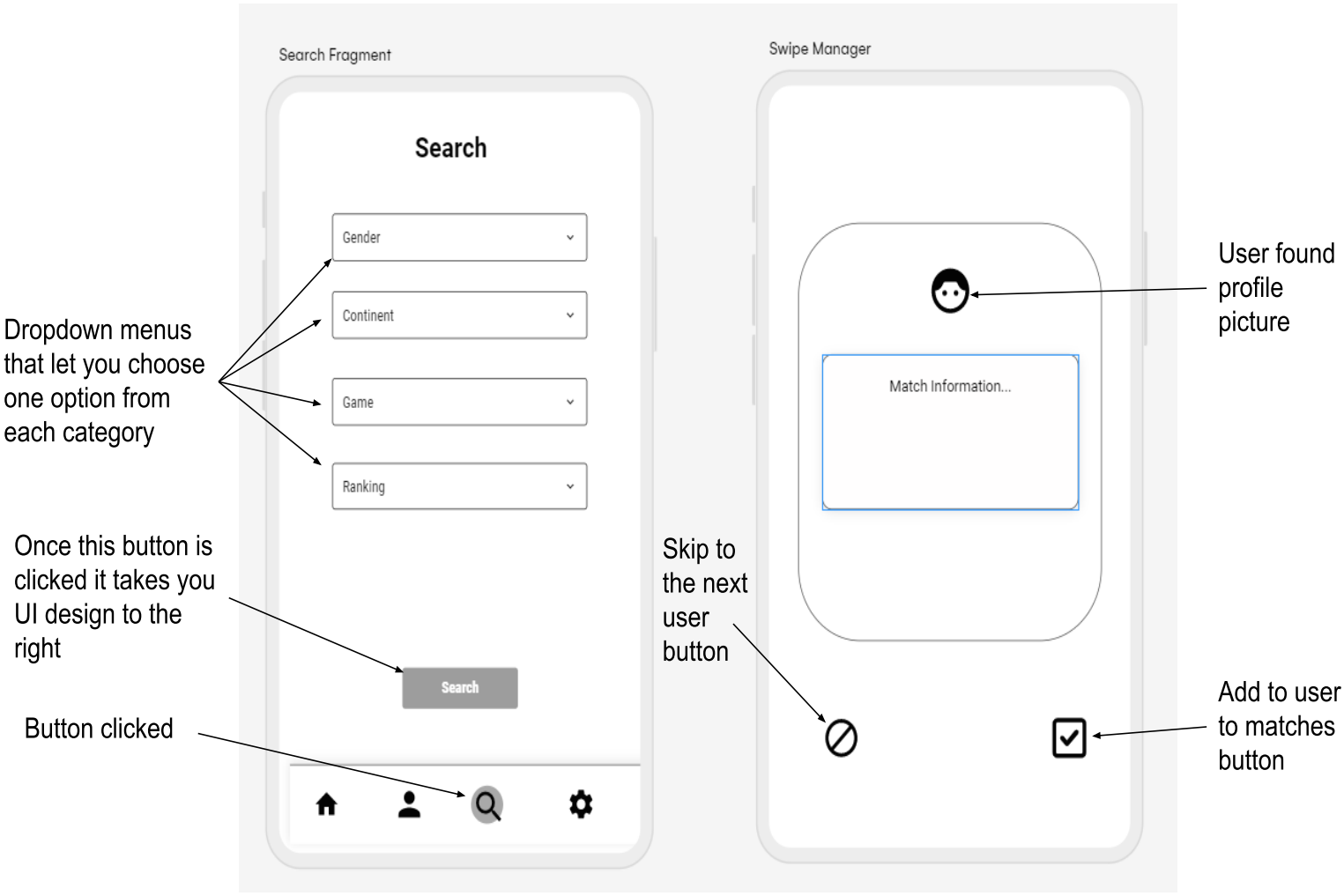


Figure 5.6: Search UI Design

In this chapter, the essential design considerations that must be taken into account before constructing an application were explored. Specifically, the client-server architecture was identified as the most appropriate system architecture for the project, given its numerous benefits. The Use Case diagram was utilized to provide a comprehensive context for understanding how users will interact with the system. Additionally, UI designs were meticulously crafted to establish a visual representation of the system's interface. All of these designs will serve as a valuable guide for developing the application, ultimately helping to achieve the desired outcome.

Chapter 6: Implementation and Testing

In this chapter, the implementation stage of the project's development will be explored. The development of each feature will be thoroughly discussed, and the reasoning behind every decision made will be provided. The development process will be guided by the requirements gathered in Chapter 4, with priority given to the "Must have," "Should have," and "Could have" requirements, as defined in the same chapter. These features are crucial for addressing the problem presented in the first chapter. After each feature is developed, appropriate testing will be conducted to ensure its successful implementation before moving on to the development of the next feature. If creating a new feature requires modifying previously developed features, those changes will be made and tested before moving on to the development of the new feature.

The project was developed using the Java programming language and the latest version of Android Studio, with their default libraries being incorporated. Google Firebase and the services it provides were utilized in order to successfully implement the majority of the application's features.

The project did not employ a specific testing framework, instead manual testing was conducted to ensure the precise implementation of each feature. This testing method was employed to ensure the efficiency, ease of use, and overall usability of the application, as well as to guarantee the successful fulfillment of each feature's intended purpose.

6.1 Feature implementation and testing

6.1.1 Login and Sign Up

In today's digital age, the creation of a login and sign up feature on an application has become a necessity as it provides users with a straightforward and convenient way to establish their accounts and access personalized content, features, and functionality within the application. By incorporating this feature, a secure and personalized user experience can be provided, enabling users to save their preferences and access their data, thus making the user experience more enjoyable and efficient. Moreover, the user's data can be protected, and unauthorized access to sensitive information, such as personal details can be prevented.

The login and sign up feature is the foundation of a social networking app thus it will be developed first. The development of the login and sign up feature was a pretty straightforward process since Google Firebase provides an authentication service that handles the whole process of creating an account for the user, assigning a unique id to the account created and after that handles the sign in process.

Upon launching the application, the user is directed to the sign-in screen (refer to Figure 6.1). Since this is the user's first time accessing the application, an account needs to be created. To do so, the user must click the register button, which will redirect them to the registration screen (refer to Figure 6.1). Here, the user can input their email address, password, and full name to create an account. Once the application verifies that the entered email address has not been previously used and that the password matches in both fields, the user is then taken to the User details screen where the user will be able to create his custom profile. The registration code can be seen in Figure 6.2, and Sign in code can be seen in Figure 6.3.

Testing

To complete the development process of all the aforementioned features, it was necessary to test their functionality and ensure that they were working as intended, both visually and logically. The application was run on an Android emulator to achieve this. An account was successfully registered, and after that, the application was logged in using the email and password used during the registration process. After that, Google Firebase Authentication was checked to make sure that an account was actually created. After successfully completing the same test for the second time this indicated that the testing of the sign-up and login features was successful and that the development of those features had come to an end.

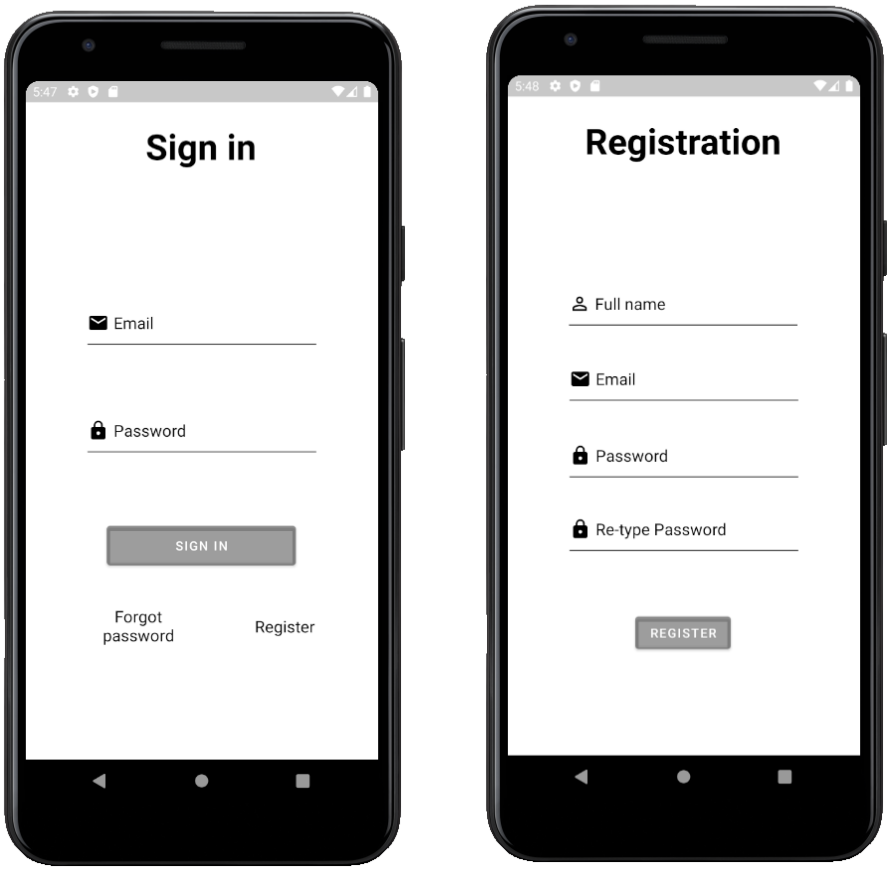


Figure 6.1: Sign in and Registration screen

```
mAuth.createUserWithEmailAndPassword(email,Password1) //create a new user account with the given email and password
        .addOnCompleteListener(new OnCompleteListener<AuthResult>() { // add a listener to check if the task is completed
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) { // check if the task is successful
                if (task.isSuccessful()){
                    // create a new user object with the given FullName and email
                    User user = new User(FullName,email, Gender: "", continent: "", FifaRanking: "", PubgRanking: "", LolRanking: "", DiscordName: "", Avatar: "");
                    FirebaseDatabase.getInstance().getReference( path: "Users") // get a reference to the "Users" node in the Firebase Realtime Database
                            // set the value of the child node with the current user's UID to the user object created earlier
                            .child(FirebaseAuth.getInstance().getCurrentUser().getUid()).setValue(user).addOnCompleteListener(new OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void> task) { // add a listener to check if the data is saved successfully
                                    if (task.isSuccessful()){
                                        Toast.makeText( context: RegistrationActivity.this,  text: "User has been registered successfully", Toast.LENGTH_SHORT).show();
                                        // sign in the newly registered user
                                        mAuth.signInWithEmailAndPassword(email,password1.getText().toString()).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                                            @Override
                                            public void onComplete(@NonNull Task<AuthResult> task) { // add a listener to check if the task is completed

                                                if (task.isSuccessful()){
                                                    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
                                                    startActivity(new Intent( packageContext: RegistrationActivity.this, UsersDetails.class)); // start the UsersDetails activity
                                                }
                                            }
                                        });
                                    }else {
                                        Toast.makeText( context: RegistrationActivity.this,  text: "Failed to register", Toast.LENGTH_SHORT).show(); // display an error message
                                    }
                                }
                            });
                }else {
                    Toast.makeText( context: RegistrationActivity.this,  text: "Failed to register", Toast.LENGTH_SHORT).show(); // display an error message
                }
```

Figure 6.2: Registration Code

```
63      private void userLogin() {
64          String email = Email.getText().toString().trim();// Get email input from user and trim whitespace
65          String password = Password.getText().toString().trim();// Get password input from user and trim whitespace
66          if (email.isEmpty()){ // Validate email input
67              Email.setError("Email is required");
68              Email.requestFocus();
69              return;
70          }if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()){ // make sure an email is entered
71              Email.setError("Please enter a valid email!");
72              Email.requestFocus();
73              return;
74          }if (password.isEmpty()){ // Validate password input
75              Password.setError("Password is required");
76              Password.requestFocus();
77              return;
78          }if (password.length()< 6){ // Validate password input
79              Password.setError("Password minimum length is 6 characters");
80              Password.requestFocus();
81              return;
82          }mAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() { // Authenticate user with Firebase Authentication
83              @Override
84              public void onComplete(@NonNull Task<AuthResult> task) {
85                  if (task.isSuccessful()){ // Check if authentication was successful
86                      FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser(); // Get current user
87                      if (user.isEmailVerified()) { // Check if user's email is verified
88                          startActivity(new Intent( packageContext: SignInActivity.this, MainActivity.class)); // Start MainActivity
89                      }else {
90                          user.sendEmailVerification(); // Send email verification to user
91                          Toast.makeText( context: SignInActivity.this,  text: "Check your email fo verification link", Toast.LENGTH_SHORT).show();
92                      }
93                  }else{
94                      Toast.makeText( context: SignInActivity.this,  text: "Failed to login! Please check your credentials", Toast.LENGTH_SHORT).show();
95                  }
```

Figure 6.3: Sign in Code

6.1.2 Email verification and Password reset

Following the development of the login and sign-up features, the implementation of two additional features was necessary for the completion of the application's authentication. The first of these was the email verification feature, designed to ensure that only real email addresses were used and that the user who created the account was the true owner. The second feature was the password reset function, which allowed users to regain access to their accounts in the event of a forgotten password or the desire to change it for any reason.

Both of these features will be implemented exclusively within the sign-in activity, making them accessible only from the sign-in screen. The email verification feature will operate as follows: the user upon attempting to log into their newly created account for the first time, gets a pop-up message indicating that the account has yet to be verified. An email notification will be sent to the email address used during the account creation process, prompting the account ownership to be verified by the user. This security feature is a service provided by Google Firebase to enhance the overall security of the application and ensure that valid and legitimate email addresses are used during account creation (refer to Figure 6.4). The password reset feature will function as follows: when the user clicks on the "forgot password" button, they will be directed to a screen where they can enter the email address associated with their account. Once the email address is entered, the user will receive an email containing instructions on how to reset their password. The user can then follow the instructions provided in the email to enter a new password and successfully reset it. This feature will help ensure that users can easily regain access to their accounts in case of a forgotten password, thereby enhancing the overall usability and accessibility of the application (refer to Figure 6.5).

Testing

The email verification and password reset features were tested separately to minimize the possibility of errors and to aid in identifying any issues. The password reset and email verification features were tested using an Android emulator. The application was run on the emulator to ensure the functionality of both features and confirm that they worked as intended. The email verification feature was checked by attempting to log in to a newly created account. A pop-up message was received indicating that the account had not been verified. An email notification was then sent to the email address used during the account creation process, prompting the user to verify their account ownership, after successful verification, login to the application was achieved, indicating the successful implementation of the feature. The password reset feature was tested by clicking the "forgot password" button on the sign in screen and entering the email address associated with the account. A password reset email was received, after following the instructions contained in the email the new password was successfully set and login to the application using the new password was achieved. This meant that the implementation of the password reset feature was successful. By testing the email verification feature and password reset feature we also made sure to test the registration and login features to ensure that all features were working correctly and that the changes we made did not affect

the system's overall functionality.Testing both features successfully marked the end of their development.

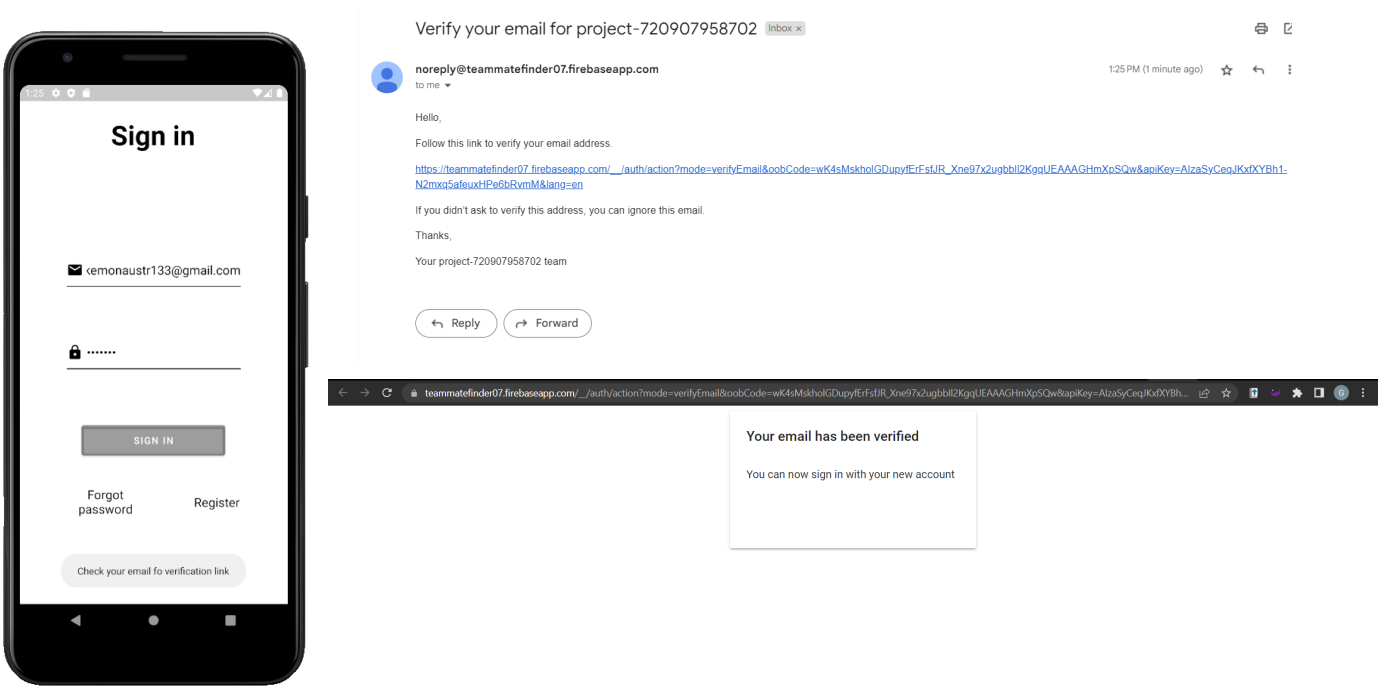


Figure 6.4: Email verification process

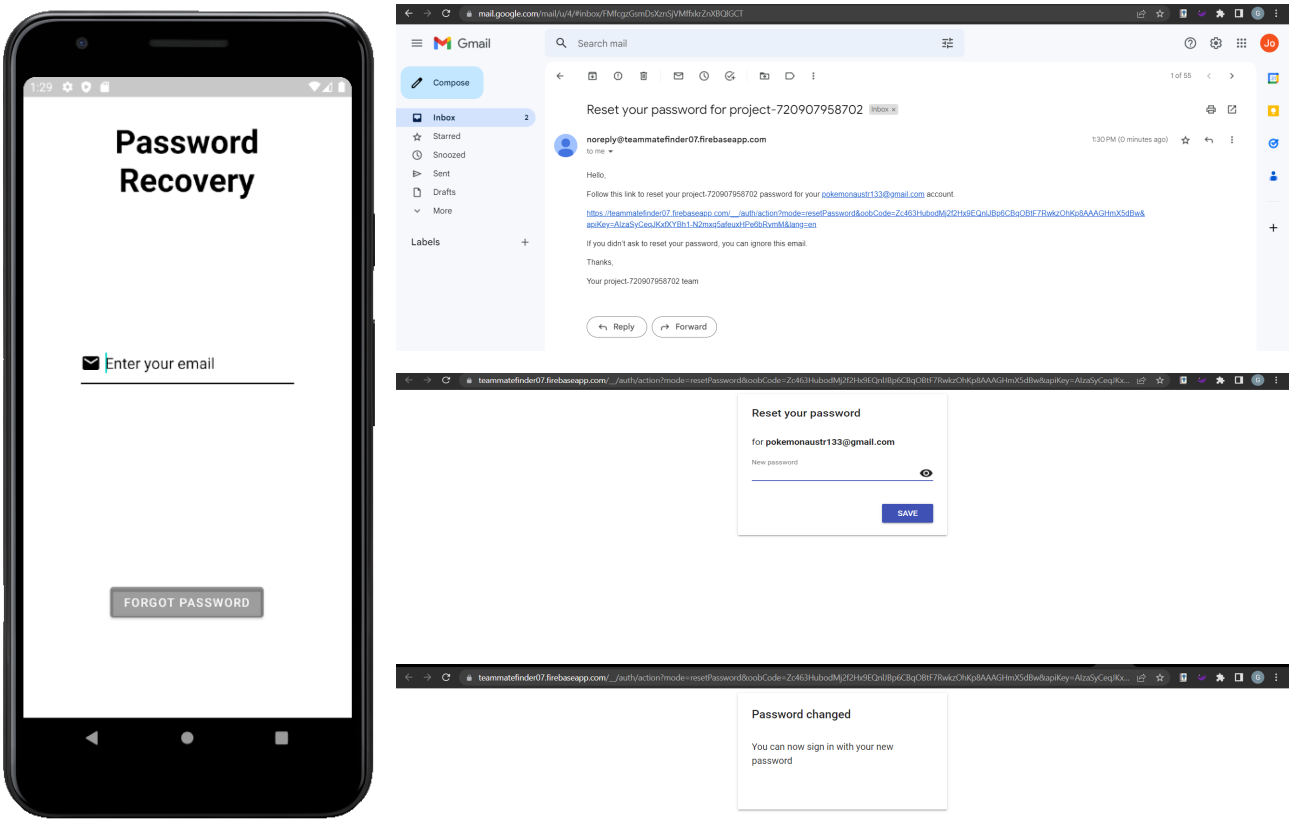


Figure 6.5: Password reset process

6.1.3 Profile creation and Real-time Database

The reason profile creation and real-time database features were chosen to be developed next is because the profile creation feature is part of the registration feature. It is recommended to implement the Firebase Realtime Database before adding the profile creation feature, as it provides a secure and reliable solution for storing user data. By setting up the database first, you can ensure that the app can instantly update and retrieve the user's data in real time. This approach also allows you to configure the database rules to ensure that only authorized users can access their own data, providing an extra layer of security. Firebase Realtime Database is a cloud-hosted NoSQL database offered by Google that allows developers to store and synchronize data in real time across multiple clients. It provides a flexible, scalable, and reliable solution for managing app data that can be accessed from anywhere via Firebase SDK. The way the real-time database will be used in this project is in combination with Firebase authentication to store the user profile information along with their potential matches. By using these two Firebase services together, you can create a robust and secure app that provides a seamless user experience. The real-time database architecture can be seen in Figure 6.6.

After the Firebase Realtime Database was integrated into the project, the profile creation feature was developed using this feature. The feature works by directing the user to the user details screen (refer to Figure 6.7) after account creation, where they can select an avatar for their profile picture, enter their Discord username, specify their gender, continent, and game rankings. Once all required fields are completed, the data is stored in the Firebase Realtime Database, which allows for easy access and updating of the information in real time. The code that takes the user profile information and stores it in the real-time database can be seen in Figure 6.8. The code in Figure 6.8 creates a HashMap object called userGames and populates it with key-value pairs representing the user's profile information. The keys are strings representing the type of information being stored (such as "FifaRanking" or "DiscordName"), and the values are the corresponding variables retrieved from the user interface. Once this HashMap is created and populated, it is used to update the user's profile data in the Firebase Realtime Database.

Testing

In order to ensure the successful implementation of both the profile creation feature and the real-time database, thorough testing was conducted. First and foremost, the profile creation feature was tested using the Android emulator. This was done to verify that the feature functioned as intended and that the screen was scrollable. Once the profile creation feature was tested successfully, with no errors encountered and the user was redirected to the home page as expected, the next step was to verify that the users' profile information was correctly stored in the real-time database. After this check, it was confirmed that both features were implemented successfully into the application. With the completion of these tests, the development process of both features came to an end.

Figure 6.6: Realtime Database Architecture

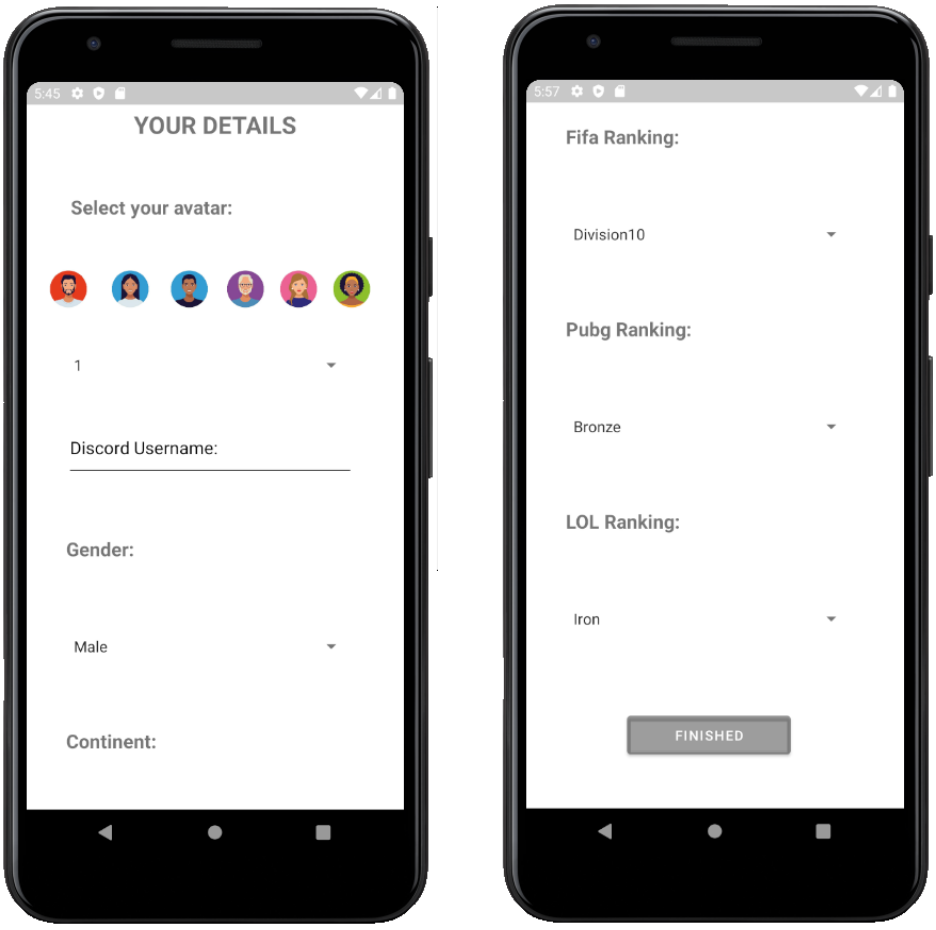


Figure 6.7: User Details Screen

```
private void storeUserDetails(){
    // Obtain information from interface
    String Gender = UserGender.getSelectedItem().toString();
    String Continent = UserContinent.getSelectedItem().toString();
    String FifaRanking = UserFifaRanking.getSelectedItem().toString();
    String PubgRanking = UserPubgRanking.getSelectedItem().toString();
    String LolRanking = UserLolRanking.getSelectedItem().toString();
    String DiscordUsername = UserDiscordUsername.getText().toString().trim();
    String Avatar = UserAvatar.getSelectedItem().toString();
    // Store user information as key-value pairs in a HashMap
    HashMap userGames = new HashMap();
    userGames.put("FifaRanking",FifaRanking);
    userGames.put("Gender",Gender);
    userGames.put("LolRanking",LolRanking);
    userGames.put("PubgRanking",PubgRanking);
    userGames.put("continent",Continent);
    userGames.put("DiscordName",DiscordUsername);
    userGames.put("Avatar",Avatar);
    user = FirebaseAuth.getInstance().getCurrentUser(); // get current signed in user
    userID = user.getUid(); // get current users id
    userDbReference = FirebaseDatabase.getInstance().getReference( path: "Users");// get a reference to the current user with the database
    userDbReference.child(userID).updateChildren(userGames).addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if (task.isSuccessful()){ // Check if the data was successfully uploaded to the database
                Toast.makeText( context: UsersDetails.this,  text: "Details updated", Toast.LENGTH_SHORT).show();
                startActivity(new Intent( packageContext: UsersDetails.this, MainActivity.class));
            }else {
                Toast.makeText( context: UsersDetails.this,  text: "Upload failed", Toast.LENGTH_SHORT).show();
            }
        }
    });
```

Figure 6.8: User Details Code

6.1.4 Navigation Bar and Logout

Navigation and logout features were developed at this point in the project because all of the other features depend on the navigation bar. A navigation bar was placed at the bottom of the application to provide users with a simple and familiar way to navigate between different areas. This approach was selected because it is widely used in popular social networking applications. The navigation bar features four clickable images that correspond to the main sections of the application: Home, Profile, Search, and Settings (refer to Figure 6.9). Text is also displayed once the desired image is clicked in order to increase accessibility and usability and make the navigation process of the application more efficient. When a certain image is clicked, the corresponding screen will be displayed, such as the home button screen displaying each user's matches list, the profile button screen displaying the current user's profile details, the search button allowing the user to select the search criteria, and the settings button giving the user the option to log out of the application. The code for the bottom navigation bar (refer to Figure 6.10) is used to define the behavior of the navigation bar, including the fragment change that occurs when a button is clicked. The code for the navigation bar works as follows:

The “bottomNavigationView” in the layout file is set up with an “onItemSelectedListener” to listen for item selection events. Whenever an item is selected, the corresponding fragment is loaded using the “fragmentChange()” function. This function takes in a “Fragment” object as a parameter and uses the “FragmentManager” to begin a “FragmentTransaction”. The current fragment is then replaced with the new fragment using the “replace()” method and the transaction is committed using “commit()”. By default, the “HomeFragment” is set as the default fragment to be displayed when the app starts, and it is loaded using the “fragmentChange()” function. This allows users to navigate between different sections of the application by selecting items in the navigation bar, and the corresponding fragments are loaded dynamically using the “FragmentTransaction”. Overall, the code above provides a simple and familiar way for users to navigate between different areas of the application using a bottom navigation bar. The process of implementing the logout feature was quite straightforward due to the availability of the feature in Google Firebase. The Firebase Auth API includes a method to log the current user out of the application, which can be easily implemented with just a few lines of code (refer to Figure 6.11).

Testing

The testing process for the navigation bar and the logout feature was carried out using an Android emulator. The emulator was launched, and the application was opened. The navigation bar was observed to be located at the bottom of the screen, featuring four clickable images corresponding to the main sections of the application. The images were clicked one by one, and it was verified that each button navigated to the corresponding section of the application. The text displayed below the images was also checked to ensure that it corresponded to the section of the application that was being navigated to.

To test the logout feature, the user navigated to the Settings section of the application and clicked the Logout button. It was verified that the user was successfully logged out of the application and directed to the login screen.

The testing process was repeated multiple times to ensure that the navigation bar and logout feature were functioning as intended, without any errors or bugs. Since the features were tested and verified without any issues, it can be concluded that it marked the end of their development.

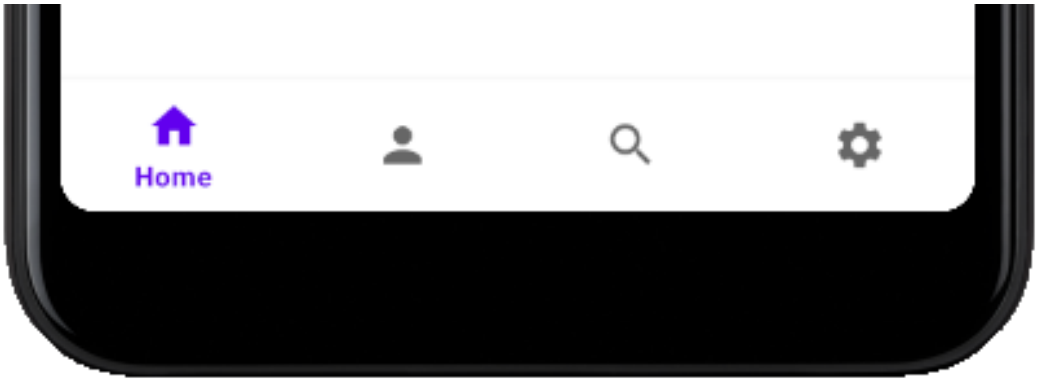


Figure 6.9: Navigation Bar

```
public class MainActivity extends AppCompatActivity {
    3 usages
    ActivityMainBinding binding; // creates a binding object for the main activity layout
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());// inflates the main activity layout using the binding object
        setContentView(binding.getRoot()); // sets the content view of the main activity layout to the root view of the binding object
        fragmentChange(new HomeFragment()); // sets the home fragment as the default when the app starts
        binding.bottomNavigationView.setOnItemSelectedListener(item -> {// sets an item selected listener for the bottom navigation view
            switch (item.getItemId()){
                case R.id.home:
                    fragmentChange(new HomeFragment()); // loads the home fragment when the home item is selected
                    break;
                case R.id.profile:
                    fragmentChange(new ProfileFragment()); // loads the profile fragment when the profile item is selected
                    break;
                case R.id.search:
                    fragmentChange(new SearchFragment()); // loads the search fragment when the search item is selected
                    break;
                case R.id.settings:
                    fragmentChange(new SettingsFragment()); // loads the settings fragment when the settings item is selected
                    break;}
            return true; // returns true to indicate that the item selection has been handled
        });
    }
    5 usages
    private void fragmentChange(Fragment fragment){ // a helper method for changing the displayed fragment
        FragmentManager fragmentManager = getSupportFragmentManager();// gets the fragment manager instance
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction(); // begins a new fragment transaction
        fragmentTransaction.replace(R.id.frame_layout,fragment);// replaces the current fragment with the new one
        fragmentTransaction.commit();// commits the transaction
    }
```

Figure 6.10: Navigation Bar Code

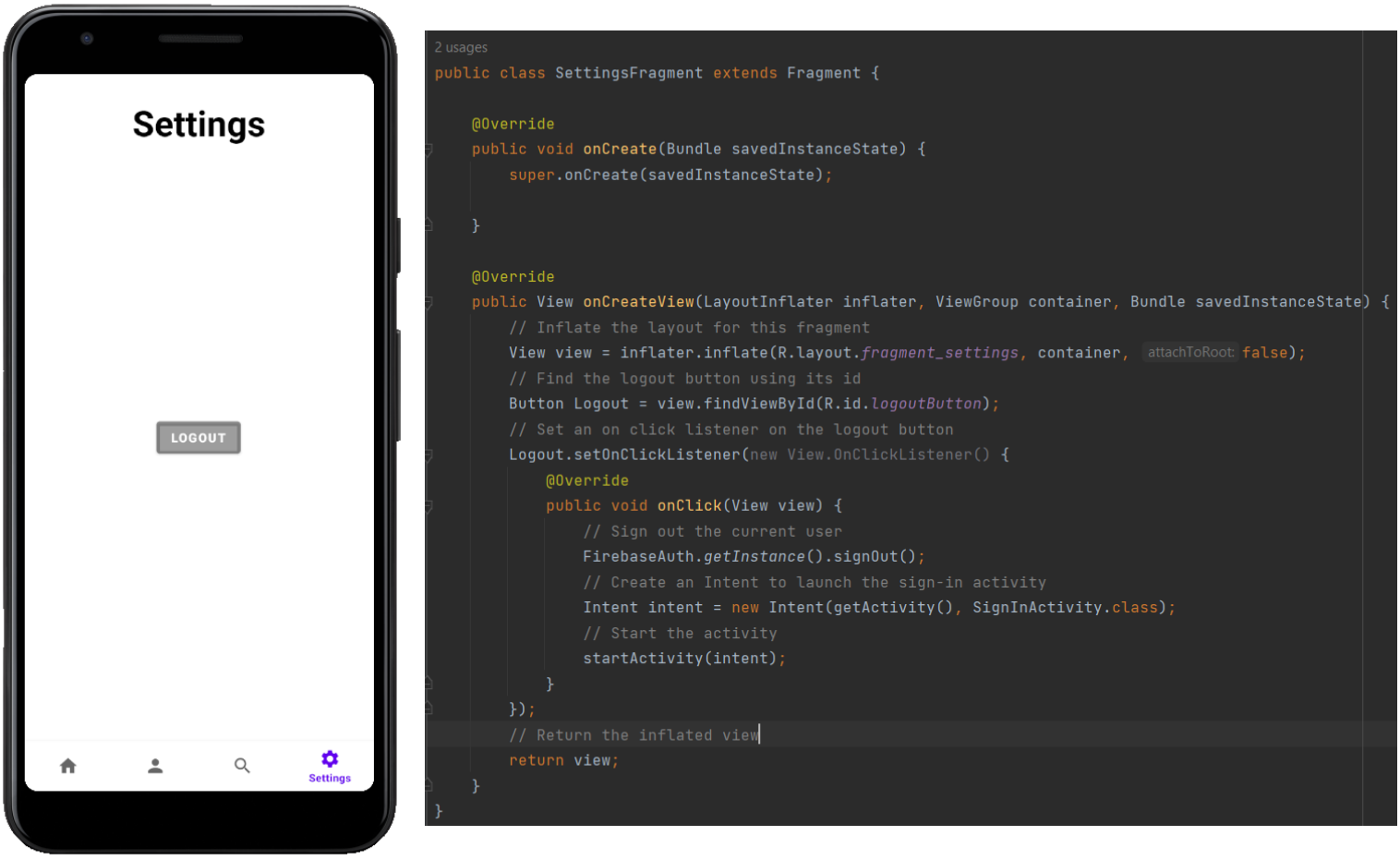


Figure 6.11: Logout screen and Code

6.1.5 Profile Customization

The Profile customization feature builds upon the previously developed Profile creation and Database features to provide users with a complete profile customization experience. It comprises two main parts:

1. Data retrieval from the Realtime Database and display of the retrieved data.
2. Profile data update that enables users to modify their profile information as needed.

When a user registers to the application for the first time, the user is prompted to select an avatar as their profile picture, input their Discord username, and select their gender, region, and game rankings. This information is then stored in the Firebase Realtime Database. When the user logs in to their account and navigates to the profile page, the ProfileFragment class (refer to Figure 6.12 and Figure 6.13) is responsible for accessing and displaying the user's profile information from the Realtime Database. The ProfileFragment class initializes a “FirebaseUser” object

and retrieves the user's ID from Firebase Authentication. It then accesses the Realtime Database reference and attaches a “ValueEventListener” to it.

The “onDataChange” method of the “ValueEventListener” retrieves the user's profile information from the Realtime Database, and the user's profile data is displayed in the corresponding UI elements of the profile page. For example, the user's name, email address, gender, region, game rankings, and Discord username are displayed (refer to Figure 6.14). Additionally, the user's profile picture is displayed by setting the ImageView's resource to the corresponding drawable resource from the avatarDrawables array.

Finally, a button is provided on the profile page to allow the user to change their game rankings. When the user clicks the button, the app navigates to the UsersDetails activity where the user can update their profile information. In case of any error, while accessing the database, an appropriate message is displayed using a Toast.

Testing

Manual testing of the profile data update and retrieval was done using the Android emulator. The testing involved creating a user account, logging in, and then updating the user's profile information. The updated information was then checked in the realtime database to ensure that the changes were reflected accurately.

To test the profile data update feature, different fields of the profile were edited, such as the user's full name, email address, gender, continent, and gaming rankings. The changes made to the user's profile were saved to the realtime database using Firebase Database. The database was then checked to ensure that the updates were accurately reflected in the user's profile.

Similarly, testing was also done to retrieve the user's profile data from the realtime database. This involved checking if the data retrieved from the database was the same as the data displayed on the user's profile in the app.

To ensure the accuracy of the manual testing, various scenarios were considered, such as testing the update and retrieval of different combinations of user profile data. Additionally, the app was tested on different Android emulator devices to ensure compatibility with various screen sizes and resolutions.

Overall, manual testing of the profile data update and retrieval feature using the Android emulator was essential in ensuring that the feature worked accurately and efficiently, and the data was correctly stored and retrieved from the realtime database.

```
37      @Override
38      public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); }
41      @Override
42      public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) { // Inflate the layout for this fragment
43          View view =  inflater.inflate(R.layout.fragment_profile, container,  attachToRoot: false); // Get current user and user ID
44          user = FirebaseAuth.getInstance().getCurrentUser();
45          userID = user.getUid(); // Get reference to Users data in Firebase database
46          reference = FirebaseDatabase.getInstance().getReference( path: "Users").child(userID);
47          // Add a ValueEventListener to the reference to update UI whenever the data changes
48          reference.addValueEventListener(new ValueEventListener() {
                2 usages
49              @Override
50              public void onDataChange(@NonNull DataSnapshot snapshot) {
51                  // get references to the TextViews, ImageView, and Button in the layout
52                  userFullName =  view.findViewById(R.id.ProfileUsername);
53                  userEmailAddress =  view.findViewById(R.id.ProfileEmail);
54                  userGender =  view.findViewById(R.id.profileGender);
55                  userContinent =  view.findViewById(R.id.profileContinent);
56                  userFifaRanking =  view.findViewById(R.id.ProfileFifa);
57                  userLolRanking =  view.findViewById(R.id.ProfileLol);
58                  userPubgRanking =  view.findViewById(R.id.ProfilePubg);
59                  rankingChangeBtn =  view.findViewById(R.id.changeRankingBtn);
60                  userDiscordName = view.findViewById(R.id.ProfileDiscord);
61                  profilePicture = view.findViewById(R.id.usersProfPic);
62                  // retrieve user data from the database
63                  String name = snapshot.child( path: "UserFullName").getValue().toString();
64                  String email = snapshot.child( path: "UserEmail").getValue().toString();
65                  String gender = snapshot.child( path: "Gender").getValue().toString();
66                  String continent = snapshot.child( path: "continent").getValue().toString();
67                  String Fifa = snapshot.child( path: "FifaRanking").getValue().toString();
```

Figure 6.12: Profile Fragment Code part 1

```
68                  String Lol = snapshot.child( path: "LolRanking").getValue().toString();
69                  String Pubg = snapshot.child( path: "PubgRanking").getValue().toString();
70                  String Discord = snapshot.child( path: "DiscordName").getValue().toString();
71                  int Avatar = Integer.parseInt(snapshot.child( path: "Avatar").getValue().toString());
72                  // set the TextViews, ImageView, and Button to display the retrieved user data
73                  userFullName.setText(name);
74                  userEmailAddress.setText(email);
75                  userGender.setText(gender);
76                  userContinent.setText(continent);
77                  userFifaRanking.setText(Fifa);
78                  userLolRanking.setText(Lol);
79                  userPubgRanking.setText(Pubg);
80                  userDiscordName.setText(Discord);
81                  profilePicture.setImageResource(avatarDrawables[Avatar-1]); // Subtract 1 because array index starts at 0
82                  // set OnClickListener on the rankingChangeBtn to start UsersDetails activity
83                  rankingChangeBtn.setOnClickListener(new View.OnClickListener() {
84                      @Override
85                      public void onClick(View view) {
86                          startActivity(new Intent(getActivity(), UsersDetails.class));
87                      }
88                  });
89              }
90              @Override
91              public void onCancelled(@NonNull DatabaseError error) {
92                  Toast.makeText(getActivity(),  text: "Something went wrong", Toast.LENGTH_SHORT).show();
93              }
94          });
95          return view;
96      }
97  }
```

Figure 6.13: Profile Fragment Code part 2

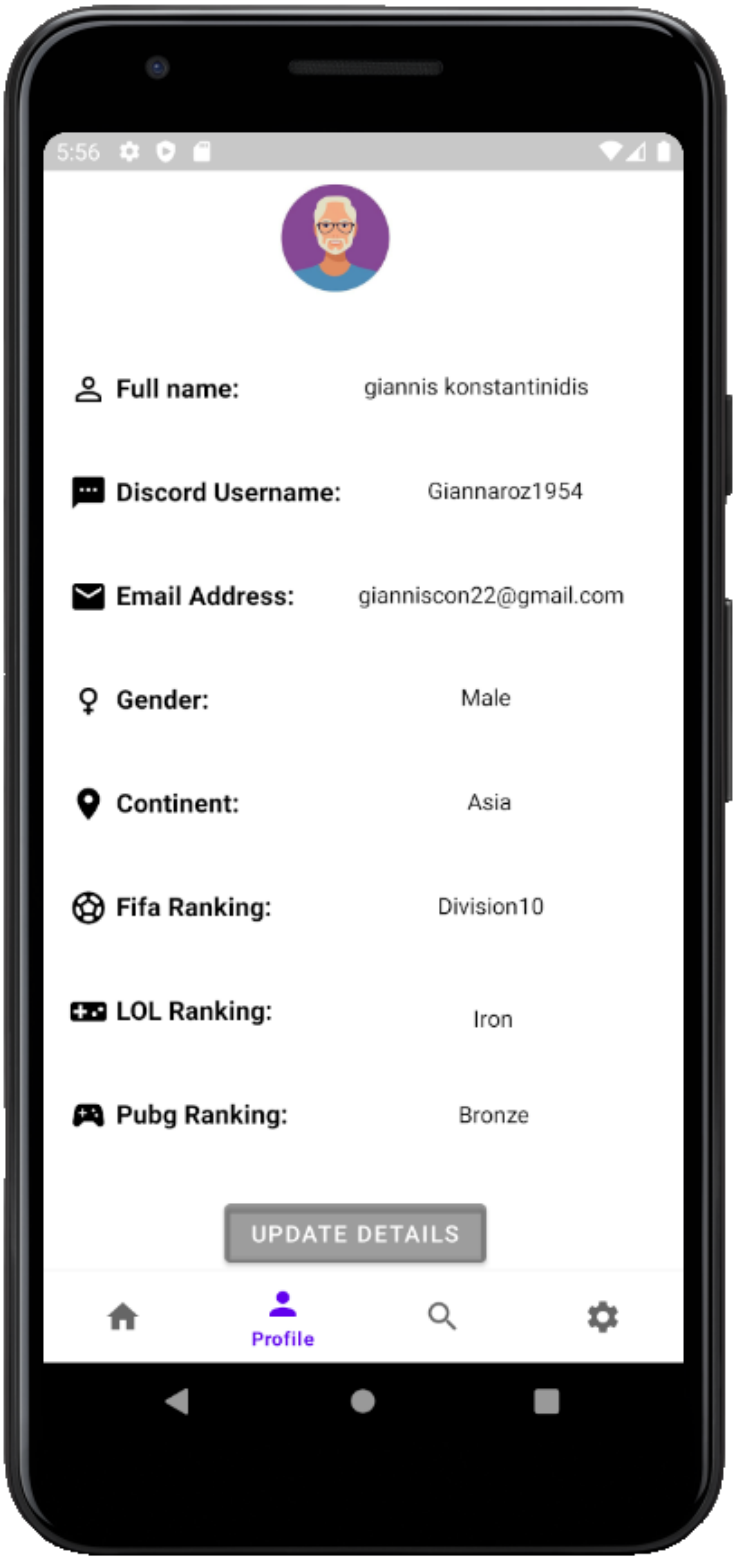


Figure 6.14: Profile Screen

6.1.6 Search and Matching Feature

The development of the search feature involved several steps. Firstly, the search criteria were defined, which included the user's preferred gender, region, game, and rank. These criteria were implemented in the search fragment layout using spinners and a button (refer to Figure 6.15). The search criteria were defined as follows:

- **Gender**: The user can select their preferred gender from a spinner that contains options for male, female, and other.
- **Region**: The user can select their preferred region from a spinner that contains options for North America, South America, Europe, Asia, Africa, Australia and Antarctica.
- **Game**: The user can select their preferred game from a spinner that contains options for FIFA, League of Legends, and PUBG.
- **Rank**: The user can select their preferred rank from a spinner that is populated dynamically based on the selected game (refer to Figure 6.16).

Once the user selects their preferred search criteria and clicks the search button, the criteria are passed to the SwipeManager class using an intent (refer to Figure 6.17). Then the SwipeManager class queries the database to find users that match the specified criteria and displays them to the user using a CardView. The CardView Contains the searched user’s avatar, fullname, email, game and game ranking (refer to Figure 6.18).

The onCreate() method initializes the SwipeManager class and retrieves the search criteria data from the intent, which is used to filter the potential matches. The currentUserID is retrieved from Firebase to get the ID of the current user. The user's email address is also retrieved in order to be used to avoid showing the current user in the search results. The usersList ArrayList is populated with all of the registered users and their data from the Firebase Realtime Database. This time the userList instead of type User is of type UserMatches, the reason behind this is because the UserMatches class contains a field that the User class doesn’t which is the Uuid field which will hold the user’s Uuid.The reason for the Uuid field will be explained later.

The while loop in the onCreate() method is used to iterate through the usersList ArrayList until a suitable match is found. If no match is found, a toast message is displayed, and the user is redirected back to the main activity. If a match is found, the relevant data is displayed on the screen, and the userFound variable is set to true so that the while loop will stop and the application will wait for user input before searching for the next match (refer to Figure 6.19).

The FloatingActionButton named yesHandler is implemented to allow users to accept a match and the noHandler FloatingActionButton to skip to the next user. Initially, the approach for adding match information to the database was to include all the details of each user in both users' userMatches field (refer to Figure 6.20) to simplify the process of searching for the current user's matches and displaying them. However, this method resulted in unsynchronized data storage in the UserMatches field. For instance, if a user updated their profile, the changes would not reflect in the userMatches field, leading to outdated data. Another problem that

this produced was a lot of unnecessary data duplication. To resolve this issue, it was decided to add only the Uuid of the matched user in both users' databases (refer to Figure 6.21). This method solved the problem of unsynchronized data and data duplication (refer to Figure 6.22).

In summary, the SwipeManager class is responsible for managing the user search of potential matches and either accept or reject them. It retrieves the search criteria data from the intent, filters the potential matches, and populates the usersList ArrayList with all of the registered users' data from the Firebase Realtime Database. Once a suitable match is found, the relevant data is displayed on the screen, and the user can accept the match by clicking the yesHandler FloatingActionButton or skip to the next user by clicking the noHandler FloatingActionButton.

Testing

Since this is the most important feature of the whole project their testing had to be extensive.To ensure that the search and swipe feature works as intended, several tests were conducted using the Android emulator. The tests were aimed at verifying that the search criteria were correctly implemented in the search fragment layout, the SwipeManager class can properly filter users based on the selected criteria, and the CardView displays relevant user data. The tests were conducted on different Android emulator devices with varying screen sizes and resolutions to ensure that the application is responsive and user-friendly on different devices.

The first test involved selecting different search criteria combinations to ensure that the SwipeManager class can correctly filter users based on the selected criteria. The test was conducted using different user accounts to ensure that the current user's email address was correctly retrieved and used to filter out the current user from the search results. The results of the test were satisfactory, as the SwipeManager class was able to filter users based on the selected criteria and display relevant user data on the CardView.

The second test focused on the FloatingActionButton named yesHandler and noHandler. The test involved clicking the yesHandler button to accept a match and clicking the noHandler button to skip to the next user. The test was conducted using different user accounts to ensure that the match information was correctly added to the database and that the Uuid of the matched user was correctly stored in both users' databases. The test results were satisfactory, as the match information was correctly added to the database, and the Uuid of the matched user was correctly stored in both users' databases.

Overall, the tests showed that the search and matching feature works as intended and that the application is responsive and user-friendly on different Android emulator devices

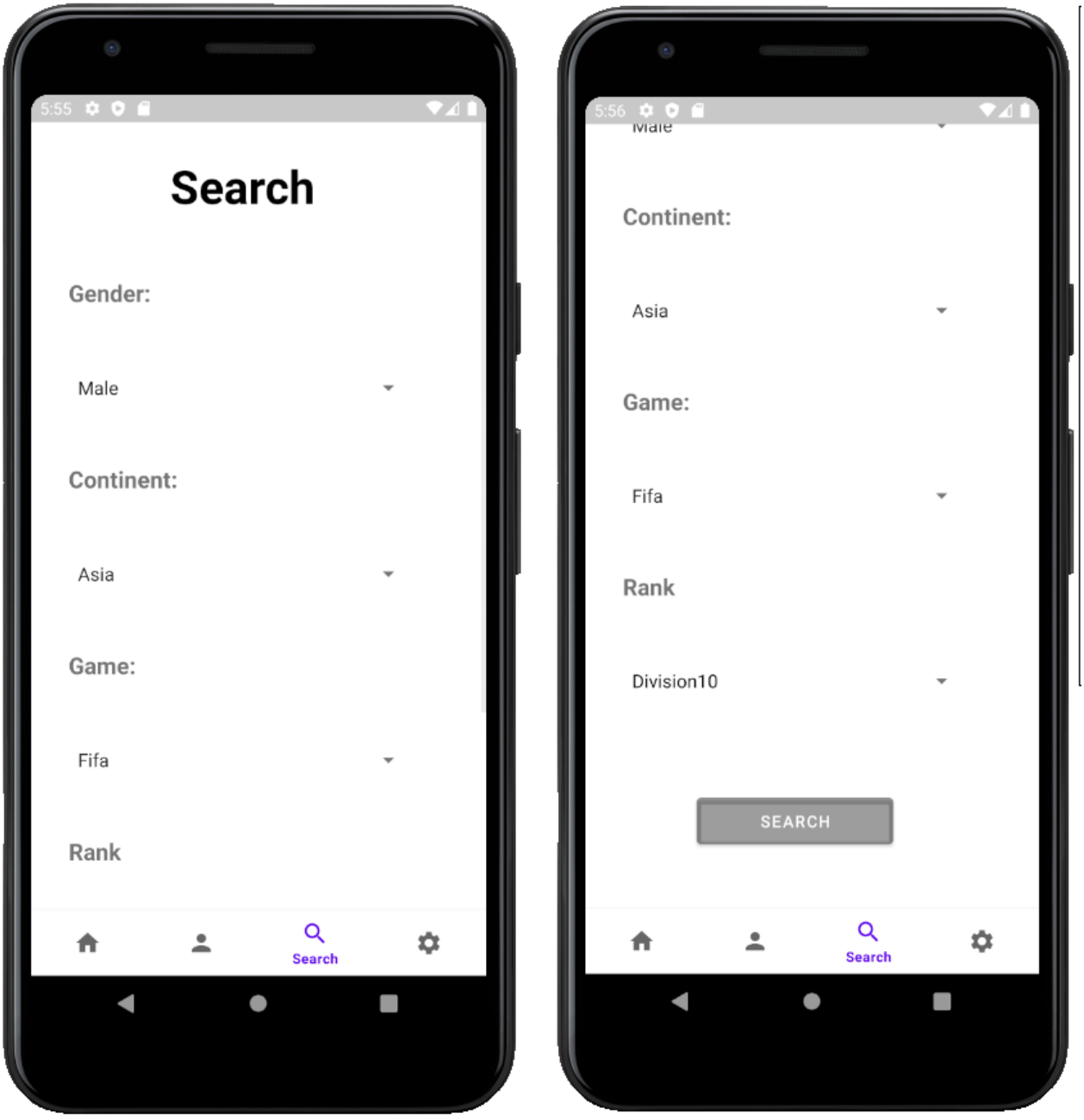


Figure 6.15: Search Screen

```java
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_search, container, attachToRoot: false); // Inflate the layout for this fragment
    // Get references to the Spinner views
    matchGender = (Spinner) view.findViewById(R.id.searchGender);
    matchContinent = (Spinner) view.findViewById(R.id.searchContinent);
    matchGame = (Spinner) view.findViewById(R.id.searchGame);
    matchRank = (Spinner) view.findViewById(R.id.searchGameRank);
    SearchBtn = (Button) view.findViewById(R.id.searchBtn);
    matchGame.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {// Set an OnItemSelectedListener for game spinner
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            String selectedItem = (String) parent.getItemAtPosition(position); // Get the selected item from game spinner
            // Update the data source for rank spinner based on the selected item in spinner1
            if (selectedItem.equals("Fifa")) {
                ArrayAdapter<CharSequence> rankAdapter1 = ArrayAdapter.createFromResource(requireContext(),
                        R.array.FifaRankings, android.R.layout.simple_spinner_dropdown_item);
                matchRank.setAdapter(rankAdapter1);
            } else if (selectedItem.equals("League Of Legends")) {
                ArrayAdapter<CharSequence> rankAdapter2 = ArrayAdapter.createFromResource(requireContext(),
                        R.array.LOLRankings, android.R.layout.simple_spinner_dropdown_item);
                matchRank.setAdapter(rankAdapter2);
            } else if (selectedItem.equals("PUBG")) {
                ArrayAdapter<CharSequence> rankAdapter3 = ArrayAdapter.createFromResource(requireContext(),
                        R.array.PUBGRankings, android.R.layout.simple_spinner_dropdown_item);
                matchRank.setAdapter(rankAdapter3);
            }
        }
        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            // Do nothing
        }
    });
```

Figure 6.16: Dynamic population of Rank Spinner

```java
    SearchBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(getActivity(), SwipeManager.class);// create an intent to start the SwipeManager Activity
            // send search criteria to SwipeManager activity once its started
            intent.putExtra( name: "gender", matchGender.getSelectedItem().toString());
            intent.putExtra( name: "region", matchContinent.getSelectedItem().toString());
            intent.putExtra( name: "game", matchGame.getSelectedItem().toString());
            intent.putExtra( name: "rank", matchRank.getSelectedItem().toString());
            startActivity(intent); // start the SwipeManager with the intent
        }
    });
    return view;
}
```

Figure 6.17: Search criteria pass to SwipeManager class

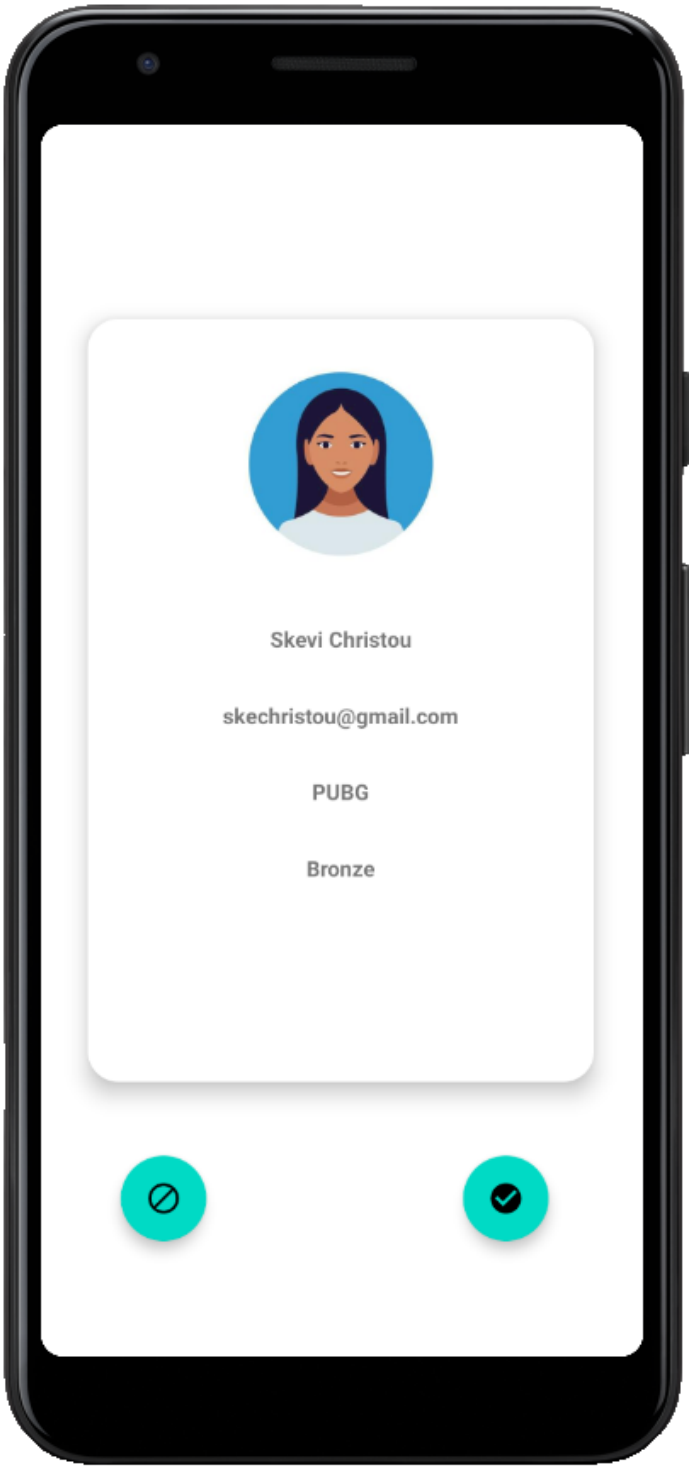


Figure 6.18: SwipeManager Screen

```
while (!userFound) {
    if (userCounter>=usersList.size()){
        Toast.makeText( context: SwipeManager.this,  text: "No more users", Toast.LENGTH_SHORT).show();
        startActivity(new Intent( packageContext: SwipeManager.this, MainActivity.class));
        userCounter=0;
        break;
    }
    else if (usersList.get(userCounter).continent.equals(selectedRegion) && usersList.get(userCounter).Gender.equals(selectedGender) &&
            selectedGame.equals("Fifa") && selectedRank.equals(usersList.get(userCounter).FifaRanking) && !usersList.get(userCounter).UserEmail.equals(currentUserEmail))
        NameTextView.setText(usersList.get(userCounter).UserFullName);
        EmailTextView.setText(usersList.get(userCounter).UserEmail);
        GameTextView.setText(selectedGame);
        RankTextView.setText(usersList.get(userCounter).FifaRanking);
        UsersAvatar.setImageResource(avatarDrawables[Integer.parseInt(usersList.get(userCounter).Avatar)-1]);
        userFound = true;
    } else if (usersList.get(userCounter).continent.equals(selectedRegion) && usersList.get(userCounter).Gender.equals(selectedGender) &&
            selectedGame.equals("League Of Legends") && selectedRank.equals(usersList.get(userCounter).LolRanking)  && !usersList.get(userCounter).UserEmail.equals(curre
        NameTextView.setText(usersList.get(userCounter).UserFullName);
        EmailTextView.setText(usersList.get(userCounter).UserEmail);
        GameTextView.setText(selectedGame);
        RankTextView.setText(usersList.get(userCounter).LolRanking);
        UsersAvatar.setImageResource(avatarDrawables[Integer.parseInt(usersList.get(userCounter).Avatar)-1]);
        userFound = true;
    } else if (usersList.get(userCounter).continent.equals(selectedRegion) && usersList.get(userCounter).Gender.equals(selectedGender) &&
            selectedGame.equals("PUBG") && selectedRank.equals(usersList.get(userCounter).PubgRanking) && !usersList.get(userCounter).UserEmail.equals(currentUserEmail))
        NameTextView.setText(usersList.get(userCounter).UserFullName);
        EmailTextView.setText(usersList.get(userCounter).UserEmail);
        GameTextView.setText(selectedGame);
        RankTextView.setText(usersList.get(userCounter).PubgRanking);
        UsersAvatar.setImageResource(avatarDrawables[Integer.parseInt(usersList.get(userCounter).Avatar)-1]);
        userFound = true;
    } else {
        userCounter++;
```

Figure 6.19: User Filtering using search criteria

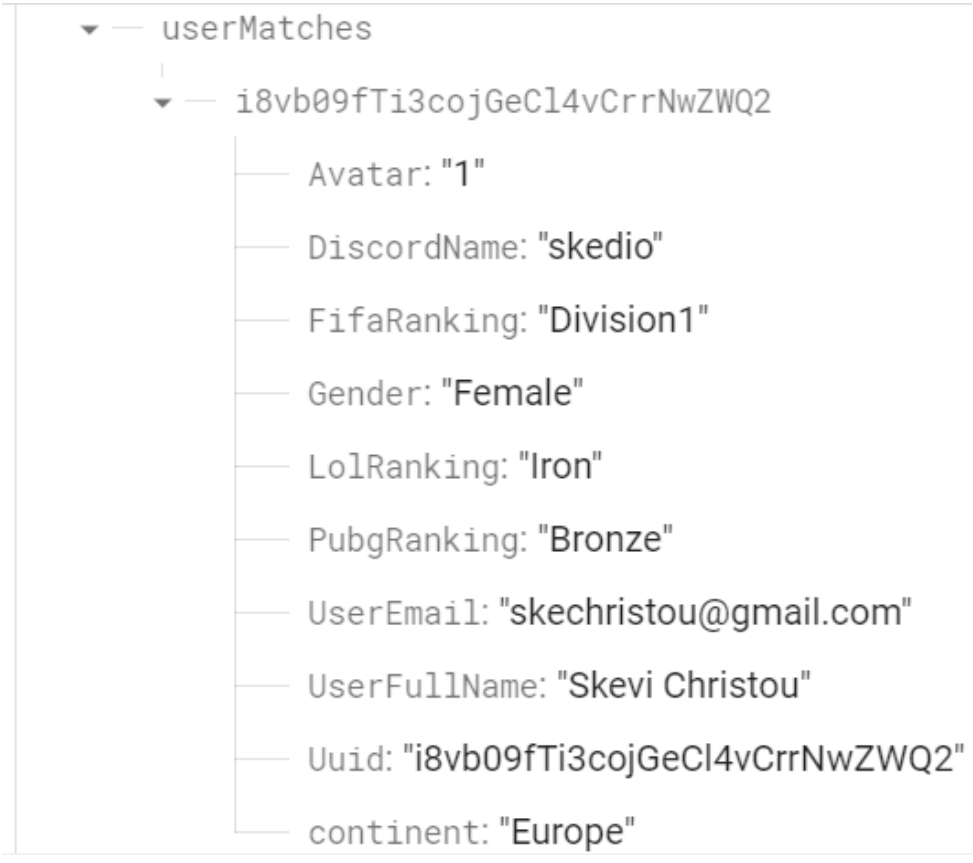


Figure 6.20: Storing all user details in the userMatches

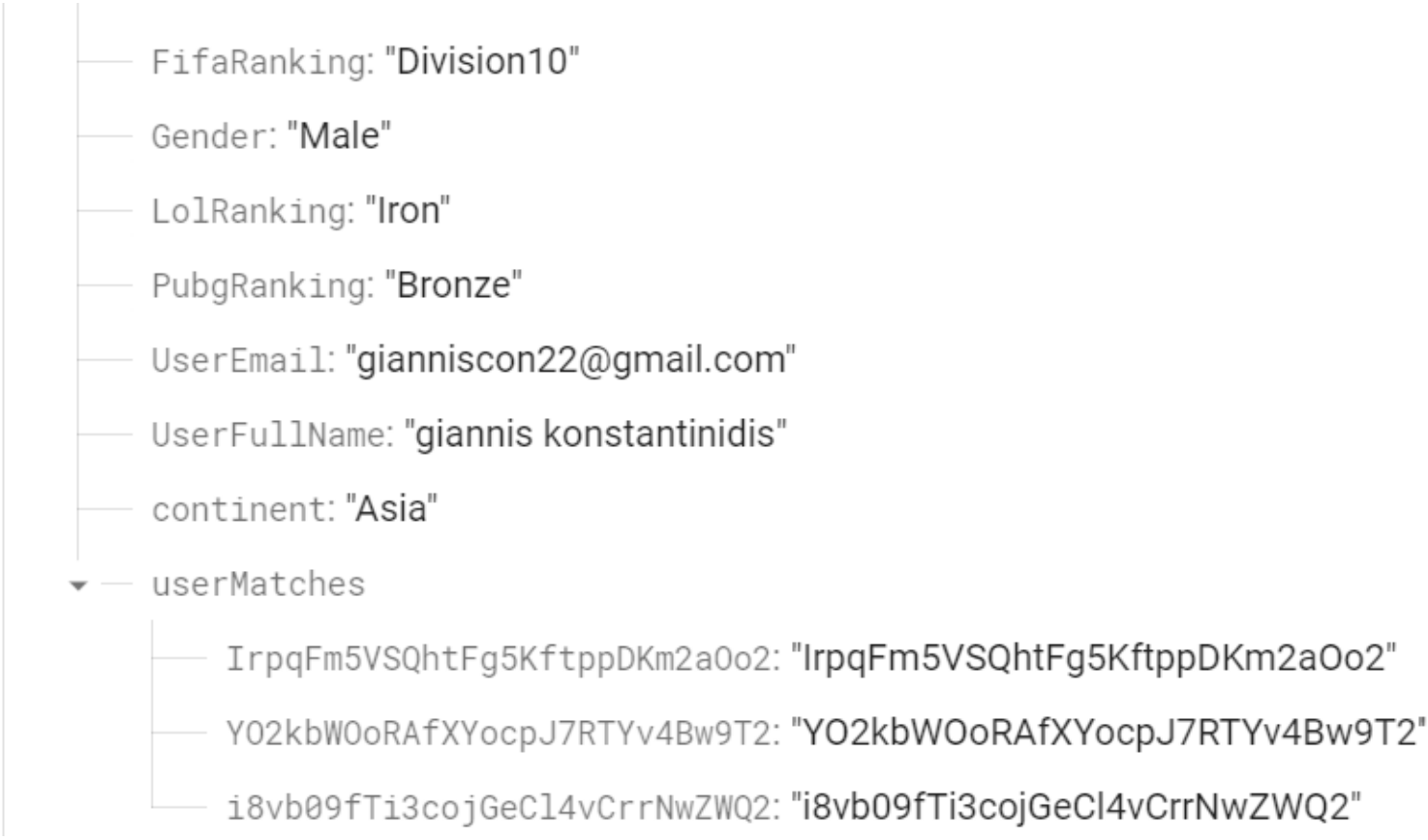


Figure 6.21: Storing only the Uuid in the userMatches

```
    yesHandler.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String matchId = usersList.get(userCounter).Uuid;
            if (userFound=true) {
                // Get a reference to the userMatches path within the current user
                DatabaseReference userMatchesRef = usersRef.child(currentUserId).child( pathString: "userMatches").child(matchId);
                // adds the match into the logged In User database
                userMatchesRef.setValue(matchId).addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        // getting a reference to the matched users database
                        DatabaseReference OtherUserMatchRef = usersRef.child(matchId).child( pathString: "userMatches").child(currentUserId);
                        OtherUserMatchRef.setValue(currentUserId);
                    }
                });
                userFound=false;
                userCounter++;}
            // function that finds the next user matching the criteria
            findNextUser();
        }});
    noHandler.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(userFound=true){
                userFound=false;
                userCounter++;
            }
            findNextUser();
        }
    });
} else {
    Toast.makeText( context: SwipeManager.this,  text: "Failed,try again", Toast.LENGTH_SHORT).show();
```

Figure 6.22: Match Addition Code

6.1.7 Matches Display and Management Feature

Both of these features are going to be developed within the Home Fragment which is the interface of the home button of the navigation bar. To create the user interface (refer to Figure 6.23) for these features, two layout files were utilized in order to achieve the desired outcome. The "fragment_home.xml" file was used to create the overall screen structure, which encompasses a TextView for displaying the screen title and a ListView for listing the matches. The "matchesList" ID was assigned to that ListView, which is later utilized to populate it with the matches data in the HomeFragment Java class. The second layout file, "matches.xml" defined the layout of each item in the ListView, which corresponds to each individual match. It contains an ImageView for displaying the match's profile picture, and a LinearLayout to house the match's name, description, and delete button. The name and description were presented using TextViews, while the delete button was denoted by a Button. The way that the Home Fragment java class (refer to figures 6.24 and 6.25) achieves the functionality of these features is as follows: Within the onCreateView() method, the fragment's layout is inflated using the "fragment_home.xml" file, and the current user's ID is retrieved from Firebase. Utilizing this ID, the "userMatches" node in the Firebase database is accessed, which contains a list of all the matches for the user. The fragment retrieves the list of match IDs for the user and uses them to look up the corresponding matches in the Firebase database. For each match, various attributes are retrieved such as the user's name, discord username, gender, region, and gaming rankings, which are then used to create a UserMatches object. These objects are then added to the matches ArrayList.Upon retrieving all matches and adding them to the matches ArrayList, the fragment creates a custom ArrayAdapter called MyAdapter. This adapter is responsible for displaying each match in the ListView utilizing a custom layout file called "matches.xml". The getView() method in MyAdapter is called for each item in the matches ArrayList, and it inflates the "matches.xml" layout file for each item. The various TextViews and ImageView in the layout file are then populated with the appropriate data from the UserMatches object. Lastly, the fragment sets the ListView's adapter to the MyAdapter object, which causes the matches to be displayed in the ListView. Overall, this process provides a user-friendly way to display matches for a specific user, using Firebase as a data source and custom layouts to enhance the user experience.

Testing

In the testing of the match display and match deletion feature, a systematic approach was taken to cover all possible scenarios. The display of matches on the UI was tested for various cases, including matches with no data, a single match, and multiple matches. During the testing of this feature an error was noticed, which lead to data duplication. The reason for that error was because every time the home button was pressed new data was added on top of the previous data of the matches within the matches ArrayList. This was solved by adding some lines of code which cleared the matches ArrayList every time the home button is pressed which solved the problem of data duplication. The deletion of matches was tested by verifying that the delete button functioned correctly and that the database was

updated accordingly. Overall, the testing was successful, and all scenarios were handled appropriately.

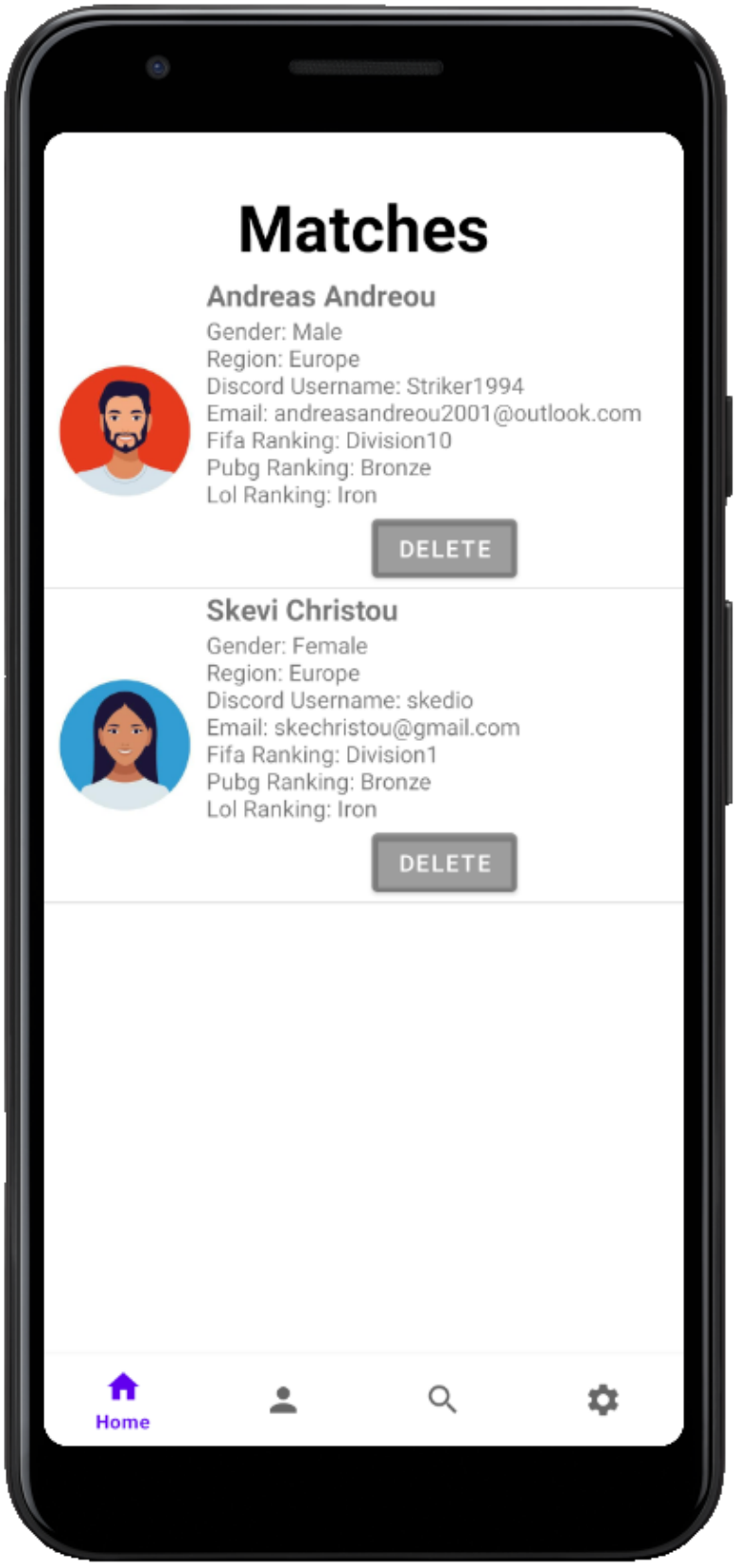


Figure 6.23: Home Screen

```java
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_home, container,  attachToRoot: false);
    user = FirebaseAuth.getInstance().getCurrentUser();
    userID = user.getUid();
    reference = FirebaseDatabase.getInstance().getReference( path: "Users").child(userID).child( pathString: "userMatches");
    reference.get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
        @Override
        public void onComplete(@NonNull Task<DataSnapshot> task) {
            if (task.isSuccessful()) {
                matchesUuid.clear(); // Clear the existing data from the matchesUuid ArrayList to avoid data duplication
                for (DataSnapshot ds : task.getResult().getChildren()) {
                    String Uuid = ds.getKey();
                    matchesUuid.add(Uuid);
                }
                DatabaseReference rootRef = FirebaseDatabase.getInstance().getReference();
                DatabaseReference usersRef = rootRef.child( pathString: "Users");
                usersRef.get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<DataSnapshot> task) {
                        if (task.isSuccessful()) {
                            matches.clear(); // Clear the existing data from the matches ArrayList to avoid data duplication
                            for (DataSnapshot ds : task.getResult().getChildren()) {
                                String email = ds.child( path: "UserEmail").getValue(String.class);
                                String FullName = ds.child( path: "UserFullName").getValue(String.class);
                                String gender = ds.child( path: "Gender").getValue(String.class);
                                String continent = ds.child( path: "continent").getValue(String.class);
                                String Fifa = ds.child( path: "FifaRanking").getValue(String.class);
                                String Pubg = ds.child( path: "PubgRanking").getValue(String.class);
                                String Lol = ds.child( path: "LolRanking").getValue(String.class);
                                String Uuid = ds.getKey();
                                String Discord = ds.child( path: "DiscordName").getValue(String.class);
                                String profilePic = ds.child( path: "Avatar").getValue(String.class);
                                for (String id : matchesUuid) {
                                    if (Uuid.equals(id)) {
                                        matches.add(new UserMatches(FullName, email, gender, continent, Fifa, Pubg, Lol, Uuid, Discord, profilePic));
                                    }
                                }
                            }

                            matchesList = view.findViewById(R.id.matchesList);
                            MyAdapter adapter = new MyAdapter(getActivity(), matches);
                            matchesList.setAdapter(adapter);
                        }
                    }
                });

            } else {
                Toast.makeText(getActivity(),  text: "Failed,try again", Toast.LENGTH_SHORT).show();
            }
        }
    });
    return view;
}
```

Figure 6.24: Home Fragment part 1

```
private static class MyAdapter extends ArrayAdapter<UserMatches> {
    1 usage
    public MyAdapter(Context context, ArrayList<UserMatches> matches) {
        super(context, R.layout.matches, matches);
    }
    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        if (convertView == null) {
            convertView = LayoutInflater.from(getContext()).inflate(R.layout.matches, parent,  attachToRoot: false);
        }
        TextView name = convertView.findViewById(R.id.matchName);
        ImageView image = convertView.findViewById(R.id.profilePic);
        Button deleteButton = convertView.findViewById(R.id.deleteButton);
        TextView desc = convertView.findViewById(R.id.matchDescription);
        UserMatches currentMatch = matches.get(position);
        name.setText(currentMatch.UserFullName);
        String description = "Gender: " + currentMatch.Gender + "\n" +
                "Region: " + currentMatch.continent + "\n" +
                "Discord Username: " + currentMatch.DiscordName + "\n" +
                "Email: " + currentMatch.UserEmail + "\n" +
                "Fifa Ranking: " + currentMatch.FifaRanking + "\n" +
                "Pubg Ranking: " + currentMatch.PubgRanking + "\n" +
                "Lol Ranking: " + currentMatch.LolRanking;
        desc.setText(description);
        // Set the avatar image based on the profilePic value
        int avatarIndex = Integer.parseInt(currentMatch.Avatar) - 1; // Subtract 1 because array index starts at 0
        image.setImageResource(avatarDrawables[avatarIndex]);
        // Handle delete button click
        deleteButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                DatabaseReference matchRef = FirebaseDatabase.getInstance().getReference( path: "Users")
                        .child(userID).child( pathString: "userMatches").child(currentMatch.Uuid);
                matchRef.removeValue().addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if (task.isSuccessful()) {
                            Toast.makeText(getContext(),  text: "Match deleted successfully", Toast.LENGTH_SHORT).show();
                            // Remove the deleted match from the ArrayList and notify the adapter
                            matches.remove(currentMatch);
                            notifyDataSetChanged();
                        } else {
                            Toast.makeText(getContext(),  text: "Failed to delete match", Toast.LENGTH_SHORT).show();
                        }
                    }
                });
            }
        });

        return convertView;
    }
}
```

Figure 6.25: Home Fragment part 2

6.2 User Testing

After completing all the features of the project and updating the Kanban board (refer to figure 6.26), user testing was conducted at the School of Computing Research showcase where the project received a recognition award. User testing was done to ensure that the application was user-friendly and met the needs of the target audience. All of the application's features were extensively tested by the users, and positive feedback was gathered, indicating that the application was well-received by users. No improvements or changes were needed based on the feedback received, which meant that the application was considered complete. The user testing phase played a critical role in ensuring that the final product was intuitive and provided a seamless user experience for users, it is expected that the needs of our target audience will be met by the application, and confidence is held in this regard.

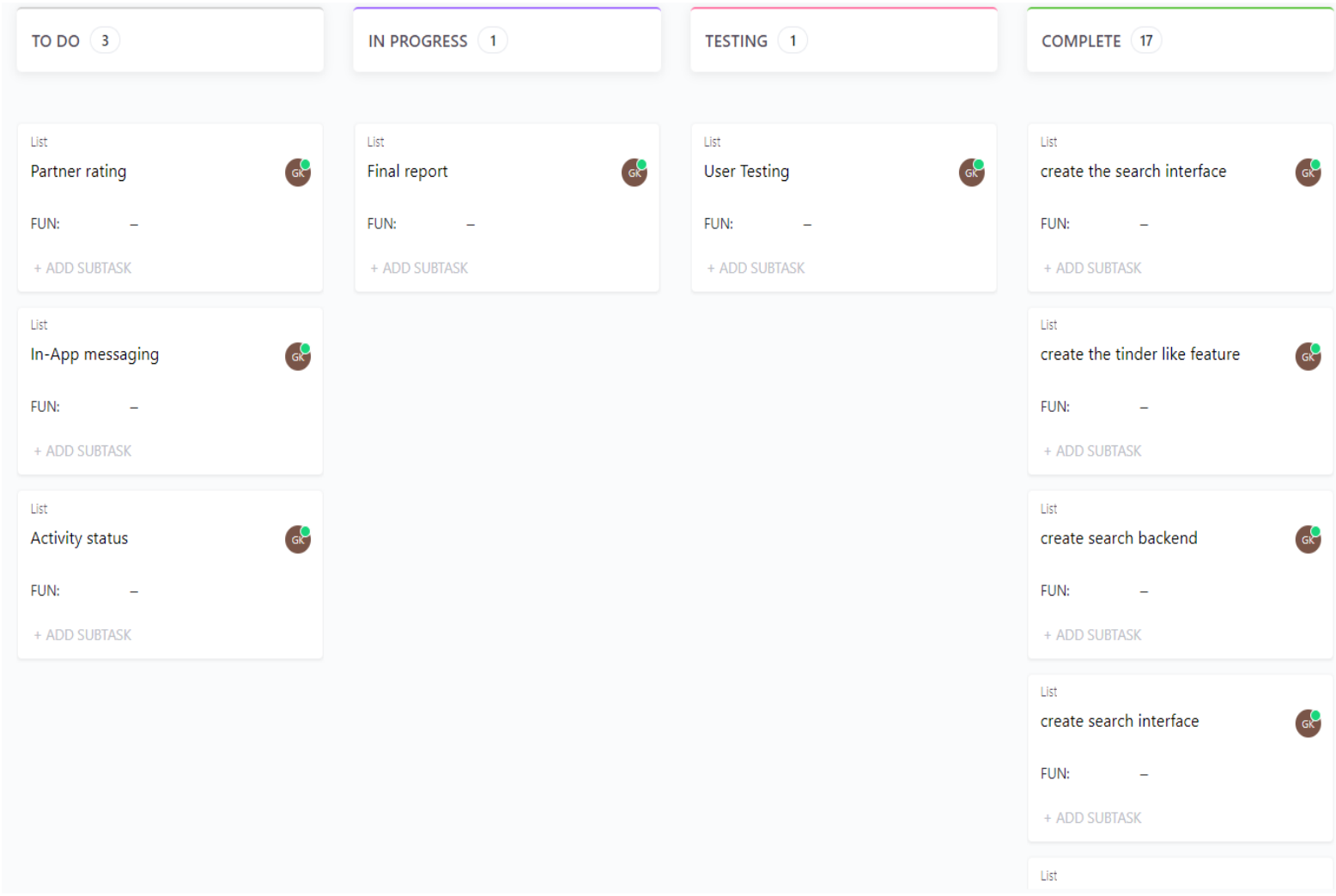


Figure 6.26: Kanban Board

This chapter details the implementation and testing process of the features, outlining the rationale behind each choice made and how the methodology, design, and requirements were utilized to achieve optimal results. The objective of the implementation process was to prioritize the features with greater importance over those that were less critical, all of which were developed and tested successfully. Despite encountering minor challenges during the development of some of the more complex features, the development and testing phase went smoothly, resulting in a successful outcome. The successful completion of this chapter is indicative of the project's success in accomplishing its main objective and paves the way for the evaluation process to commence.

Chapter 7: Evaluation

In this chapter, an assessment will be made of the project in relation to its primary objectives and requirements, as well as a comparison with similar existing applications to gauge its competitive edge. Additionally, an evaluation will be conducted on the methodology utilized in the project's development. The aim of this evaluation is to provide a comprehensive analysis of the project's performance in meeting its objectives and requirements, as well as its competitiveness relative to other similar applications.

7.1 Project Objectives Evaluation

The primary goal of this project was to develop a platform where individuals could discover suitable gaming teammates and establish meaningful friendships, as stated in chapter 1, which was accomplished successfully. To achieve this goal, a set of objectives were identified, including investigating various methodologies and practices to determine the most effective approach for creating the application, as achieved in chapter 3. Additionally, the requirements were gathered using appropriate methods such as research and a survey, as described in chapter 4. The research also entailed studying similar applications to identify areas for improvement, as covered in chapter 2. The implementation and testing phase, as discussed in chapter 6, were executed to verify that the application met the specified requirements. Upon evaluating the project's success, it can be observed that all objectives were successfully met. The application's design, features, and functionality proved to be effective in meeting the project's aim. Moreover, the application's implementation was aligned with the optimal practices and methodologies investigated earlier. In conclusion, the project was successful in delivering a platform that facilitates the formation of gaming partnerships and friendships, based on the achievement of the identified objectives.

7.2 Evaluation Against Existing Systems

To gain a comprehensive understanding of what was achieved in this project, The table below (refer to Table 7.1) will present a comparative analysis of the application developed in this project with the applications studied in chapter 2. The table aims to provide a clear perspective of the application's feature ranking, enabling readers to assess the project's accomplishments.

Features	This Application	GamerLink (GamerLink Inc, 2014)	Plink(DogApps, 2018)	Eblitz(eBlitz Ltd, 2021)	Epal(Epal Inc,2020)
Login Feature	**X**	**X**	**X**	**X**	**X**
Profile	**X**	**X**	**X**	**X**	**X**
In app messages	**-**	**X**	**X**	**X**	**X**
Skill criteria	**X**	**X**	**X**	**-**	**-**
Tinder(Tinder, 2013) like swipe feature	**X**	**-**	**X**	**X**	**-**
Gamer rating	**-**	**-**	**X**	**-**	**-**
Game stats	**X**	**X**	**X**	**X**	**-**
Free	**X**	**X**	**-**	**-**	**-**

Table 7.1: Comparison against existing application

By examining the table above, it becomes evident that this application outperforms both the Eblitz and Epal applications, as they lack many of the core features that are displayed in the table. Furthermore, a comparison between this application and the GamerLink app reveals that although both are missing two features, the application developed is superior since the omitted features are less crucial in accomplishing the intended goal. Conversely, GamerLink's absence of the Tinder-like swipe feature is a significant drawback for gamers, making the developed application a more appealing choice. Lastly, in contrast to the Plink application, which was previously identified as the best of the four applications in the literature review, the developed application offers all of Plink's core features but is also free, making it more accessible and user-friendly. In summary, the application developed can more than be considered competitive against the other applications mentioned if not superior to most of them.

7.3 Project Requirements Evaluation

This section will focus on evaluating the functional and non-functional requirements and will be divided into two sub-sections. To aid in the evaluation process and improve clarity, a table will be used to showcase the results. The table will provide a

comprehensive overview of the evaluation, making the process more understandable and accessible.

7.3.1 Functional Requirements

The table below (refer to Table 7.2) will be used for the evaluation process of the functional requirements that were going to be implemented. As mentioned before the aim of this project was to implement all of the features of higher priority than “May have in the future”.

Requirement	Implementation	Evaluation
Login/Sign up(Must Have)	**Implemented**	Login and signup were successfully implemented using Firebase Authentication.Users are able to create an account and then log in to their account by using an email address and a password.
Authentication(Must Have)	**Implemented**	Email authentication was successfully implemented, after the creation of the user’s account an email is being sent to the user in order to verify their email.
Password Reset(Could Have)	**Implemented**	Password reset feature successfully lets the users recover their account in case they forget their password just by sending them an email with instruction on how to reset it.
Database(Must Have)	**Implemented**	Firebase Database was successfully utilized, data can be stored, retrieved and deleted with no problem. It is also scalable and can handle a huge number of data.
Profile creation and customization(Should Have)	**Implemented**	This feature was successfully implemented and lets the user customize their profile by entering the appropriate information and giving access to the user to edit that at any time.
Require Discord username(Could Have)	**Implemented**	This feature was successfully implemented and provides the users with a way of communication, this feature is displayed in the user matches profile.
Avatar selection(Could Have)	**Implemented**	This feature was successfully implemented by giving the user a choice between 6 avatars in order to represent as a profile picture and can be seen in both the users profile.

Search functionality based on specific criteria(Must Have)	**Implemented**	This feature allows the user to select the search criteria and searches through the database to find all users that match that criteria and show them to the user, excluding the user from the search.
Friend matching(Must Have)	**Implemented**	This feature allows the user to add users to their matches by clicking the like button for the users shown in the search.
Matches list(Must Have)	**Implemented**	This feature successfully displayed the user matches list in the home page of the application.
Friend removing(Must Have)	**Implemented**	This feature allows the user to remove any match from the matches list of the home page.
User games(Should Have)	**Implemented**	This feature gives the user the ability to pick the games in both the search criteria and profile creation.
User rating(May have in the future)	**Not Implemented**	This feature wasn't implemented due to time constraints as it is not a core feature for the application but would definitely be considered for future implementation.
User rankings(Should Have)	**Implemented**	This feature allows the user to pick game rankings in both the search criteria and the profile customization.
Navigation Bar(Should Have)	**Implemented**	This feature allows the user to easily and efficiently navigate through the application improving the overall user experience.
Log out(Must Have)	**Implemented**	This feature allows the user to log out of their account at any time.
In-App messages(May have in the future)	**Not Implemented**	This feature wasn't implemented due to limited time, but other ways of communication were used such as providing both the users discord username and email address.
Activity status(May have in the future)	**Not Implemented**	This feature due to limited time and also because it doesn't provide much to the user wasn't implemented.

Table 7.2: Functional Requirements Evaluation

7.3.2 Non-Functional Requirements Evaluation

The table below (refer to Table 7.3) will be used to evaluate the Non-functional requirements for this project.

Requirement	Implementation	Evaluation
Android Compatibility(Must Have)	**Implemented**	By using API 26: Android 8.0 (oreo) the application is compatible with approximately 90.7% of android devices.
Reliability(Must Have)	**Implemented**	The application can perform consistently and accurately over time under different operating conditions, without failures, errors, or unexpected interruptions, and maintain the integrity and consistency of processed data.
Scalability(Could Have)	**Implemented**	The application can easily be scaled due to the database used which is Firebase which offers a number of services and plans to accommodate scalability.
Accessibility(Must Have)	**Implemented**	The application is designed and developed to be usable and navigable by all users, including those with disabilities or impairments, by providing appropriate interfaces such as using sp instead of dp for the font sizes because sp scales to the user's font and increases user experience for users with eyesight problems
Performance(Must Have)	**Implemented**	The application executes its tasks and operations efficiently, reliably, and responsively, within acceptable time and resource constraints, and meets the expected service levels, throughput, and response times, even under high loads or stress conditions.
Usability(Should Have)	**Implemented**	The application is easy to learn, use, and understand by the users, by providing clear and consistent interfaces, workflows, and feedback mechanisms, and minimizing cognitive load and errors, in order to enhance user satisfaction, productivity, and engagement.
Security(Should Have)	**Implemented**	The application protects its users from unauthorized access, and data breaches, by implementing appropriate security

		mechanisms, protocols, and policies.This was achieved by implementing email authentication, authorization, and password forget features.

Table 7.3: Non-Functional Requirements Evaluation

7.4 Methodology Evaluation

The Kanban agile methodology was selected for the project due to its ability to accommodate changes and focus on continuous delivery of value to the customer. The approach emphasized on visualizing the workflow and limiting work in progress, ensuring a steady and efficient delivery of value. Tasks were prioritized and broken down into smaller, manageable pieces to ensure the delivery of the most important features first.

One of the key benefits of using Kanban was its flexibility, which allowed the team to adjust their plans and focus on the most valuable work. This approach enabled for changes to be handled more effectively and allowed for the incorporation of new ideas that arose during the development process or the improvement of existing ones.

Although some modifications had to be made to suit the single-developer environment, the Kanban methodology's overall impact on the project was positive. By prioritizing the most important features and breaking down work into manageable pieces, the single developer was able to deliver value frequently, providing a continuous stream of updates and enhancements. The feedback and metrics from each feature development and testing also played a crucial role in improving the process, increasing efficiency and ensuring the final product met the intended goal.

This chapter encompasses multiple evaluations aimed at assessing the success of the project's objectives and requirements. Firstly, the application was evaluated against the project's objectives to determine if the set goals were achieved. The functional and non-functional requirements were also assessed to confirm that they were implemented successfully. Secondly, a comparison was made between the developed application and those researched in the literature review chapter to establish its ranking in relation to the other applications. Finally, the project's methodology was evaluated, highlighting the benefits it offered to the development process. By conducting these evaluations, a comprehensive understanding of the project's success and areas for improvement was gained.

Chapter 8: Potential Improvements and Conclusion

This chapter provides a comprehensive summary of the entire project and outlines potential areas for future improvement of the application. Three critical potential developments are highlighted that could significantly enhance the application's functionality and overall user experience.

8.1 Potential Improvements

While the application can compete with existing similar applications, it still has significant room for improvement. There are several potential developments that could transform this application into an innovative solution that could help millions of people connect with gaming partners worldwide. The implementation of an in-app messaging feature could significantly enhance the application's usability, allowing users to communicate with one another directly within the platform. This feature would not only increase engagement but also eliminate the need for external communication platforms. Another crucial feature that could improve the user experience is a user rating system. This system would enable users to rate others they have matched with in the past, providing valuable information to avoid potentially toxic or fraudulent users. Lastly, enhancing the profile customization feature by allowing users to upload pictures instead of just selecting avatars would help users have a better understanding of the individuals they are communicating and matching with.

8.2 Conclusion

This project aimed to tackle the issue of social isolation and limited social connections among gamers, which has been exacerbated by various factors such as COVID-19, shyness, and social awkwardness. To address this problem, an Android application was developed to assist gamers in finding gaming partners based on their preferences in games, gender, and rankings. The application achieved this by providing users with a search feature, allowing them to search for potential matches based on the specified criteria. Users could then add a user to their matches list if they were interested or skip the user by pressing the dislike button. As the application successfully solved the problem it was intended to address, all of the aims and objectives outlined in the first chapter, as well as the requirements gathered in Chapter 4, were met.

Throughout the course of the project, a number of valuable skills, experiences and knowledge were gained. These included the ability to plan and manage a software development project, the knowledge of the importance of user-centered design, and the practical experience in using various programming languages and tools such as Java, Android Studio and Google Firebase. Additionally, I was able to improve my problem-solving skills, critical thinking as well as learn about the importance of testing and debugging during software development. Moreover, I gained experience in conducting research and gathering requirements, as well as in writing technical documentation. Overall, a range of new knowledge, skills and experiences were gained through the course of the project, all of which will be useful in future software development projects.

This chapter serves as a reflection on the entire project, highlighting the remarkable achievements and valuable lessons learned during the project's development. Additionally, it identifies potential areas for improvement that could not be implemented due to time constraints.

References

Turner, A. (2023). *Android vs. Apple Market Share: Leading Mobile OS*. https://www.bankmycell.com/blog/android-vs-apple-market-share/

Kowal, M., Conroy, E., Ramsbottom, N., Smithies, T. D., Toth, A. J., & Campbell, M. (2021). Gaming Your Mental Health: A Narrative Review on Mitigating Symptoms of Depression and Anxiety Using Commercial Video Games. JMIR Serious Games, 9(2), e26575–e26575. https://doi.org/10.2196/26575

Perry, R., Anders Drachen, Kearney, A., Kriglstein, S., Nacke, L. E., Rafet Sifa, Wallner, G., & Johnson, D. (2018). Online-only friends, real-life friends or strangers? Differential associations with passion and social capital in video game play. Computers in Human Behavior, 79, 202–210. https://doi.org/10.1016/j.chb.2017.10.032

Prochnow, T., Hartnell, L., & Patterson, M. S. (2021). Depressive symptoms, developing friendships, and social support through online gaming: a mixed-method analysis of online gaming network communication. Mental Health and Social Inclusion, 25(3), 243-253. https://www.emerald.com/insight/content/doi/10.1108/MHSI-02-2021-0011/full/pdf?casa_token=vElAIbdfO7wAAAAA:_eyhLP0zKItSgGIIGT8k7W891596isUhoR4tXsMT9THJownPjcvCH8BbKrvI2AwI1Zz19z7BOTs_4hV0bccK1_Q0i0-IBIvrRrPjn2pLugnFXYRuEqPO

GamerLink Inc. (2014). GamerLink LFG: Teams & Friends. Google.com. https://play.google.com/store/apps/details?id=gg.gamerlink.gamerlink

DogApps. (2018). Plink: Team up, Chat & Play. Google.com. https://play.google.com/store/apps/details?id=tech.plink.PlinkApp

eBlitz Ltd. (2021). eBlitz - Find Gaming Friends. Google.com. https://play.google.com/store/apps/details?id=gg.eblitz.eblitzapp

Tinder. (2013). Tinder - Dating Made Easy. Google.com. https://play.google.com/store/apps/details?id=com.tinder

Epal Inc. (2020). Epal: Play games, Meet friends. Google.com. https://play.google.com/store/apps/details?id=com.epal.android

Ahmed, A., Ahmad, S., Ehsan, N., Mirza, E., & Sarwar, S. Z. (2010). Agile software development: Impact on productivity and quality. 2010 IEEE International Conference on Management of Innovation & Technology. https://doi.org/10.1109/icmit.2010.5492703

Cohen, D., Lindvall, M., & Costa, P. (2004). An introduction to agile methods. Adv. Comput., 62(03), 1-66. https://books.google.co.uk/books?hl=en&lr=&id=N-06uoJ9iSsC&oi=fnd&pg=PA1&dq=importance+of+choosing+the+correct+development+methodology&ots=l7gU3RXoZi&sig=iW1yzW6TdbJnKyKz_mdR-c8iTaA&redir_esc=y#v=onepage&q=importance%20of%20choosing%20the%20correct%20development%20methodology&f=false

Geambaşu, C., Jianu, I., Jianu, I., & Gavrilă, A. (2011). INFLUENCE FACTORS FOR THE CHOICE OF A SOFTWARE DEVELOPMENT METHODOLOGY. Accounting and Management Information Systems, 10(4), 479–494. https://core.ac.uk/download/pdf/6261795.pdf

Ahmad, M. O., Markkula, J., & Oivo, M. (2013). Kanban in software development: A systematic literature review. 2013 39th Euromicro Conference on Software Engineering and Advanced Applications. https://doi.org/10.1109/seaa.2013.28

McCormick, M. (2012). Waterfall vs. Agile methodology. MPCS, N/A, 3. http://www.mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf

L. Wait Rising, & Janoff, N. (2000). The Scrum software development process for small teams. IEEE Software, 17(4), 26–32. https://doi.org/10.1109/52.854065

MoSCoW Prioritization. (2022, December 21). Productplan.com. https://www.productplan.com/glossary/moscow-prioritization/

Electronic Arts. (2022). EA SPORTSTM FIFA 23 - Official Site. Electronic Arts Inc.; Electronic Arts. https://www.ea.com/en-gb/games/fifa/fifa-23

Level Infinite. (2018). PUBG MOBILE. Google.com. https://play.google.com/store/apps/details?id=com.tencent.ig

Riot Games Inc. (2020). League of Legends: Wild Rift. Google.com. https://play.google.com/store/apps/details?id=com.riotgames.league.wildrift

Google Forms. (2019). Google.com. https://docs.google.com/forms/u/0/?tgif=d

Malan, R., & Bredemeyer, D. (2001). Functional requirements and use cases. Bredemeyer Consulting.https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=aceaa41855c38aebe7c823e60e94b39506a92b99

Chung, L., Nixon, B. A., Yu, E., & Mylopoulos, J. (2012). Non-functional requirements in software engineering (Vol. 5). Springer Science & Business Media. https://books.google.com.cy/books?hl=en&lr=&id=MNrcBwAAQBAJ&oi=fnd&pg=PR15&dq=functional+requirements&ots=g2LdH5pSeN&sig=RMBZ06eHaEhhSB3wApw0CYMuTOg&redir_esc=y#v=onepage&q=functional%20requirements&f=false

Oluwatosin, H. S. (2014). Client-server model. IOSR Journal of Computer Engineering, 16(1), 67-71. https://www.researchgate.net/profile/Shakirat-Sulyman/publication/271295146_Client-Server_Model/links/5864e11308ae8fce490c1b01/Client-Server-Model.pdf

Hura, G. S. (1995). Client-server computing architecture: an efficient paradigm for project management. In Proceedings for Operating Research and the Management Sciences (pp. 146-152). IEEE.https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=523924

Peter Zsolt Bodrogi. (2003). Chromaticity contrast in visual search on the multi-colour user interface. Displays, 24(1), 39–48. https://doi.org/10.1016/s0141-9382(02)00070-7

Firebase. (2012). Firebase. https://firebase.google.com/

Vecteezy. (2023). Vecteezy. https://www.vecteezy.com/members/jemastock

Darejeh, A., & Singh, D. (2013). A review on user interface design principles to increase software usability for users with less computer literacy. Journal of computer science, 9(11), 1443.https://scholar.google.com/scholar?hl=en&as_sdt=0,5&qsp=1&q=%22software+usability%22+user+interface+design+principles&qst=br

Android Developers. (2014).Android mobile App Developer tools. Retrieved from https://developer.android.com

Uizard. (2021). https://app.uizard.io/prototypes/pbdMbYWn17UdMXP3Ay5v

Draw.io - free flowchart maker and diagrams online. Flowchart Maker & Online Diagram Software. (2012). https://app.diagrams.net/

Clickup. clickup.com. (2017). https://app.clickup.com/9005038910/v/b/8cbvz9y-108

Google. (2012). Firebase realtime database. Google. https://firebase.google.com/docs/database

Appendices

Appendix A: Project Initiation Document

(Next Page)

School of Computing
Final Year Engineering Project

Project Initiation Document

Giannis Konstantinidis

Teammate finder Android Application

1. Basic details

Student name:	Giannis Konstantinidis
Draft project title:	Teammate finder Android Application
Course and year:	Computer science, third year
Project supervisor:	Dr. Elisavet Andrikopoulou
Client organisation:	*N/A*
Client contact name:	N/A

2. Degree suitability

The course I'm currently studying is computer science which includes a lot of programming, problem solving and critical thinking which I will need to use in order to complete my final year project. This makes me believe that the project that I chose is more than suitable for a computer science degree.

3. Outline of the project environment and problem to be solved

*For engineering **projects without a client**:*	*For projects **with a client**:*	*For **research** or **study** projects:*
The problem that I will be trying to solve is finding gaming partners for people that have no friends to play games with. The reason that the project I chose is worth working on is because a lot of people, myself included, have a hard time finding decent teammates to play with, which I believe can be solved through my project.		

4. Project aim and objectives

The overall aim for the project is to help every gamer find a suitable teammate/teammates with whom they can enjoy playing video games with.
The objectives that will help me achieve that are:

- The literature review
- Real-time matching feature
- Gender option feature
- Region option feature
- Game option feature
- Game rank option feature
- Age option feature
- Social media tags feature

5. Project constraints

The project constraints that i might face are:

- Not enough survey answers
- Poor time management

6. Facilities and resources

The platform I’m going to use in order to build my software is Android studio and the resources that will help me do that are the library,the web and my supervisor.

7. Log of risks

Description	Impact	Mitigation/Avoidance
COVID-19 outbreak means I cannot get into a lab for usability testing	Cause delays to the project timeline	Get in while I can, prioritise lab tasks in time. Make an alternate test plan that does not need the lab.
Software failure means that android studio stops function hence i won't be able to develop my application	Cause delays to the project timeline	Try to develop the application while the android studio is still functioning.
No users for user testing	Cause delays to the project timeline	Talk to more people or email more people that might be willing to help me with testing.
Supervisor is unavailable due to health or other reasons	Won't be able to get feedback on my progress	Get feedback on my work.
Personal health issues	Won't be able to do much work.	Try to do as much work as possible while being healthy.
Mental breakdown	Won't be able to do much work.	Not leave everything till the last minute, and try to stay according to the schedule.

8. Project deliverables

The artefact that is going to be developed is an android application.
The documents that are going to be produced are a project report.

9. Project approach

The development methodology I am going to use is agile and more specifically the Kanban software development methodology. The reason I chose this methodology is because it is best suited for small teams, and provides planning flexibility due to the fact that it is only focused on the work that's actively in progress.

The research I will have to do is on the existing applications that are similar to mine that are currently available on play store.

10. Project plan

- Write project report
- Survey
- literature review
- requirements gathering
- analysis
- Software development
- user testing and testing
- write-up

The skills I will require to complete this project is java knowledge and some basic database knowledge.

I will need access to other people during my survey, other than that I don't think I will need any other access to people.

11. Supervisor Meetings

My supervisor and I have arranged weekly meetings every Tuesday at 11 a.m. - 11:30 a.m.
The reason why we scheduled weekly meetings is so that if we miss a week it won’t matter that much because we would have already had a meeting the previous week.

12. Legal, ethical, professional, social issues (mandatory)

The data that is going to be collected is going to be anonymized and protected by the UoP secure network.I will receive consent from the participants for the data collection and they will be able to withdraw from the study at any point they want.

Appendix A: Ethics certificate

Certificate of Ethics Review

Project title: Teammate Finder

Name:	Giannis Konstantinidis	**User ID**:	954123	**Application date**:	20/10/2022 12:16:12	**ER Number**:	TETHIC-2022-103938

You must download your referral certificate, print a copy and keep it as a record of this review.

The FEC representative(s) for the **School of Computing** is/are **Haythem Nakkas, Dalin Zhou**

It is your responsibility to follow the University Code of Practice on Ethical Standards and any Department/School or professional guidelines in the conduct of your study including relevant guidelines regarding health and safety of researchers including the following:

- University Policy
- Safety on Geological Fieldwork

It is also your responsibility to follow University guidance on Data Protection Policy:

- General guidance for all data protection issues
- University Data Protection Policy

Which school/department do you belong to?: **School of Computing**
What is your primary role at the University?: **Undergraduate Student**
What is the name of the member of staff who is responsible for supervising your project?: **Elisavet Andrikopoulou**
Is the study likely to involve human subjects (observation) or participants?: Yes
Will you gather data about people (e.g. socio-economic, clinical, psychological, biological)?: No
Will you gather data from people about some artefact or research question (e.g. opinions, feedback)?: Yes
Will the study involve National Health Service patients or staff?: No
Do human participants/subjects take part in studies without their knowledge/consent at the time, or will deception of any sort be involved? (e.g. covert observation of people, especially if in a non-public place): No
Will you collect or analyse personally identifiable information about anyone or monitor their communications or on-line activities without their explicit consent?: No
Does the study involve participants who are unable to give informed consent or are in a dependent position (e.g. children, people with learning disabilities, unconscious patients, Portsmouth University students)?: Yes
Are drugs, placebos or other substances (e.g. food substances, vitamins) to be administered to the study participants?: No
Will blood or tissue samples be obtained from participants?: No
Is pain or more than mild discomfort likely to result from the study?: No
Could the study induce psychological stress or anxiety in participants or third parties?: No
Will the study involve prolonged or repetitive testing?: No
Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?: No
Are there risks of significant damage to physical and/or ecological environmental features?: No
Are there risks of significant damage to features of historical or cultural heritage (e.g. impacts of study techniques, taking of samples)?: No
Does the project involve animals in any way?: No
Could the research outputs potentially be harmful to third parties?: No
Could your research/artefact be adapted and be misused?: Yes
Will your project or project deliverables be relevant to defence, the military, police or other security organisations and/or in addition, could it be used by others to threaten UK security?: No

Supervisor Review

As supervisor, I will ensure that this work will be conducted in an ethical manner in line with the University Ethics Policy.

Supervisor comments: **##supervisorComments##**

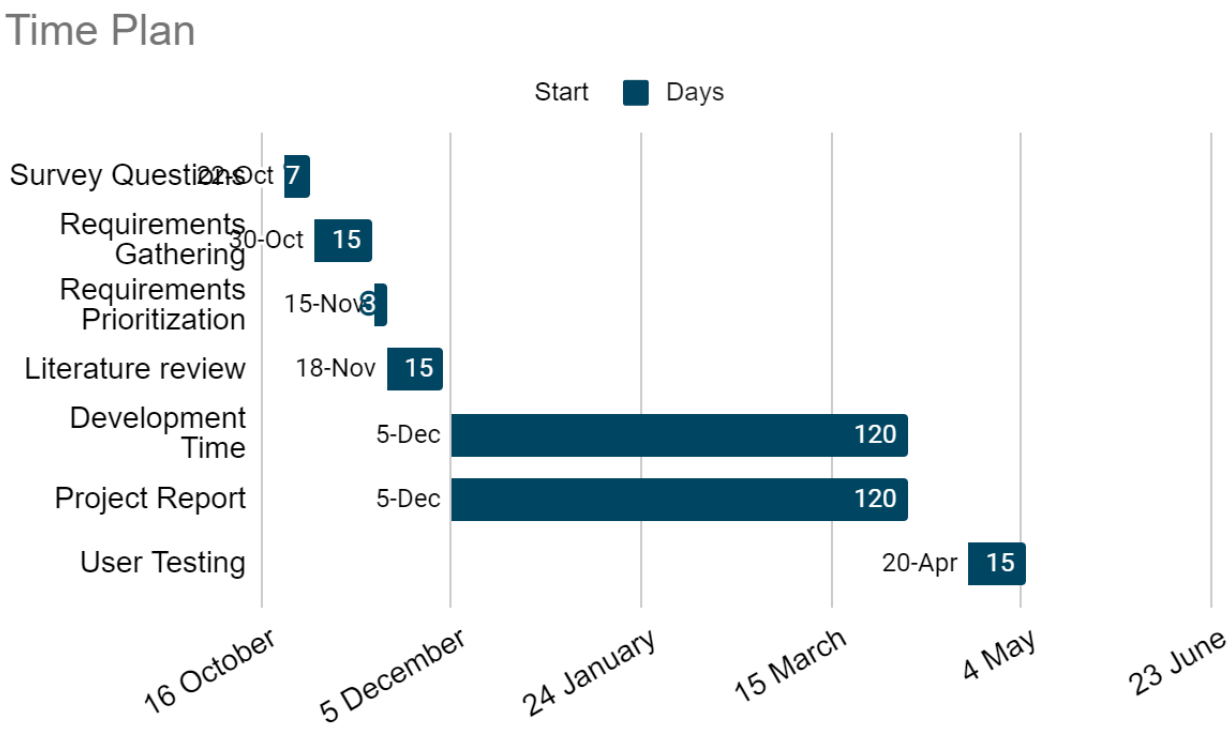
Supervisor's Digital Signature: **##supervisorSig##** Date: **##supDate##**

Faculty Ethics Committee Review

Ethics Rep comments: **##ethicsRepComments##**

Faculty Ethics Committee Member's Digital Signature(s): **##repSig##** Date: **##repDate##**

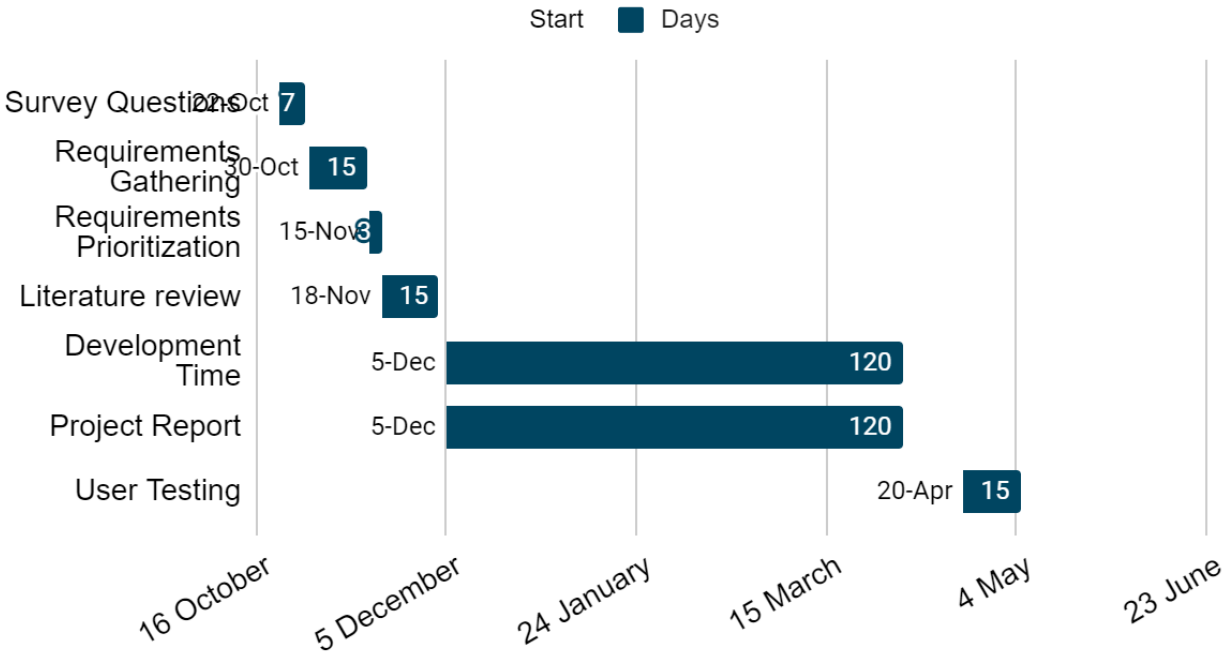
Appendix B: Gantt chart



Appendix B : Gantt Chart

(Next Page)

Time Plan


Start
Days
Survey Questions
22-Oct
7
Requirements Gathering
30-Oct
15
Requirements Prioritization
15-Nov
3
Literature review
18-Nov
15
Development Time
5-Dec
120
Project Report
5-Dec
120
User Testing
20-Apr
15
16 October
5 December
24 January
15 March
4 May
23 June

Appendix C : Ethics form

(Next Page)

Certificate of Ethics Review

Project title: Teammate Finder

Name:	Giannis Konstantinidis	**User ID:**	954123	**Application date:**	21/10/2022 10:47:07	**ER Number:**	TETHIC-2022-104025

You must download your referral certificate, print a copy and keep it as a record of this review.

The FEC representative(s) for the **School of Computing** is/are **Elisavet Andrikopoulou, Kirsten Smith**

It is your responsibility to follow the University Code of Practice on Ethical Standards and any Department/School or professional guidelines in the conduct of your study including relevant guidelines regarding health and safety of researchers including the following:

- University Policy
- Safety on Geological Fieldwork

It is also your responsibility to follow University guidance on Data Protection Policy:

- General guidance for all data protection issues
- University Data Protection Policy

Which school/department do you belong to?: **School of Computing**
What is your primary role at the University?: **Undergraduate Student**
What is the name of the member of staff who is responsible for supervising your project?: **Elisavet Andrikopoulou**
Is the study likely to involve human subjects (observation) or participants?: Yes
Will you gather data about people (e.g. socio-economic, clinical, psychological, biological)?: No
Will you gather data from people about some artefact or research question (e.g. opinions, feedback)?: Yes
Confirm whether and explain how you will use participant information sheets and apply informed consent.: I will collect data through a survey
Confirm whether and explain how you will maintain participant anonymity and confidentiality of data collected: The data will be protected by the university of portsmouth
Will the study involve National Health Service patients or staff?: No
Do human participants/subjects take part in studies without their knowledge/consent at the time, or will deception of any sort be involved? (e.g. covert observation of people, especially if in a non-public place): No
Will you collect or analyse personally identifiable information about anyone or monitor their communications or on-line activities without their explicit consent?: No
Does the study involve participants who are unable to give informed consent or are in a dependent position (e.g. children, people with learning disabilities, unconscious patients, Portsmouth University students)?: No
Are drugs, placebos or other substances (e.g. food substances, vitamins) to be administered to the study participants?: No
Will blood or tissue samples be obtained from participants?: No
Is pain or more than mild discomfort likely to result from the study?: No
Could the study induce psychological stress or anxiety in participants or third parties?: No
Will the study involve prolonged or repetitive testing?: No
Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?: No
Are there risks of significant damage to physical and/or ecological environmental features?: No
Are there risks of significant damage to features of historical or cultural heritage (e.g. impacts of study techniques, taking of samples)?: No
Does the project involve animals in any way?: No
Could the research outputs potentially be harmful to third parties?: No
Could your research/artefact be adapted and be misused?: Yes
Identify any risks associated. How do you plan to minimise risks?: People could disguise themselves as

somebody else and use the application. The measure i'm going to take against that are strict verification of identity

Will your project or project deliverables be relevant to defence, the military, police or other security organisations and/or in addition, could it be used by others to threaten UK security?: No

Please read and confirm that you agree with the following statements: I confirm that I have considered the implications for data collection and use, taking into consideration legal requirements (UK GDPR, Data Protection Act 2018 etc.), I confirm that I have considered the impact of this work and and taken any reasonable action to mitigate potential misuse of the project outputs, I confirm that I will act ethically and honestly throughout this project

Supervisor Review

As supervisor, I will ensure that this work will be conducted in an ethical manner in line with the University Ethics Policy.

Supervisor comments:

Supervisor's Digital Signature: **elisavet.andrikopoulou@port.ac.uk** Date: **14/12/2022**

Faculty Ethics Committee Review

Ethics Rep comments:

Faculty Ethics Committee Member's Digital Signature(s): **elisavet.andrikopoulou@port.ac.uk** Date: **14/12/2022**

Appendix D : Survey Results

(Next Page)

47 responses

Link to Sheets

Not accepting responses

Message for respondents

This form is no longer accepting responses

Summary | Question | Individual

Do you understand everything mentioned above?

Copy

47 responses

Required Questions

Are you an adult?

Copy

47 responses

Required questions

Do you want to participate in the survey?

Copy

47 responses

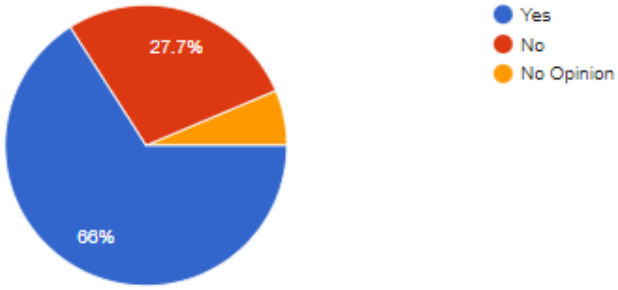
Goodbye, thank you for your time but you're not eligible to take part in this survey. If you would like to find out more please email me at up954123@myport.ac.uk or elisavet.andrikopoulou@port.ac.uk.

Survey

Should there be a gender choice in the teammate search criteria?

Copy

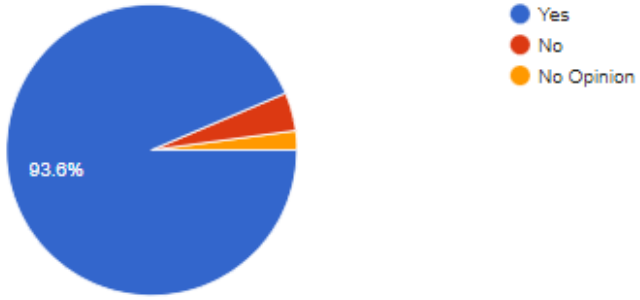
47 responses



Should there be a skill level choice in the teammate search criteria?

Copy

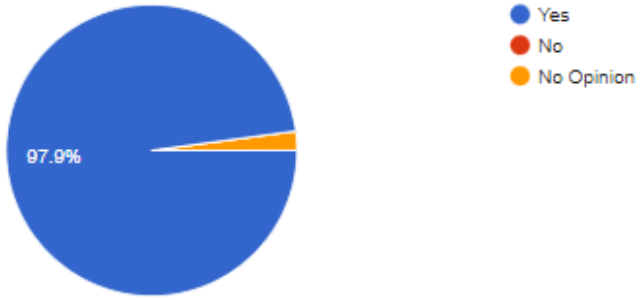
47 responses



Should there be a Login Feature?

Copy

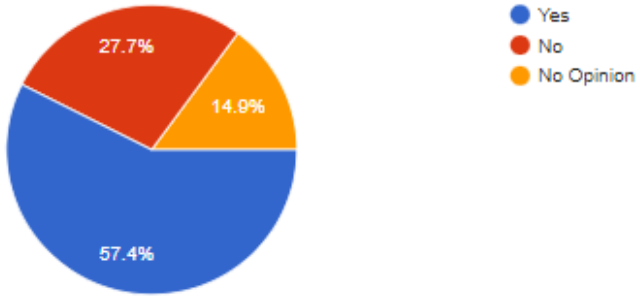
47 responses



Should the app require unique usernames?

Copy

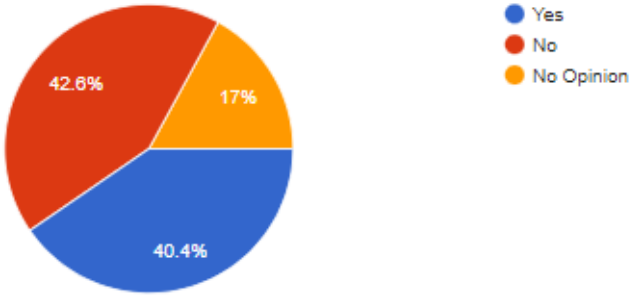
47 responses



Should there be real time teammate matching?

Copy

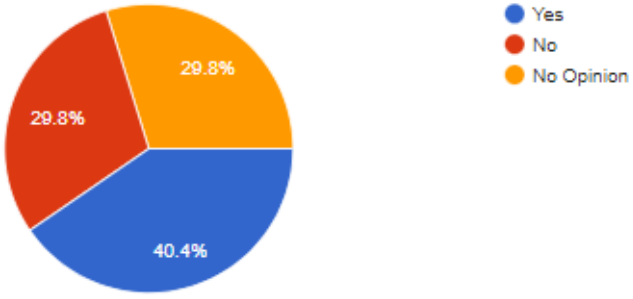
47 responses



Would you like to have a choice of the interface colour?

Copy

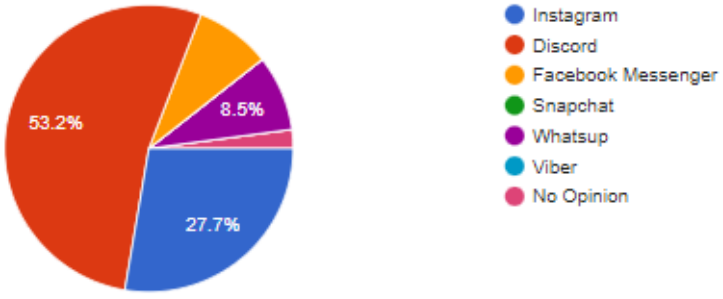
47 responses



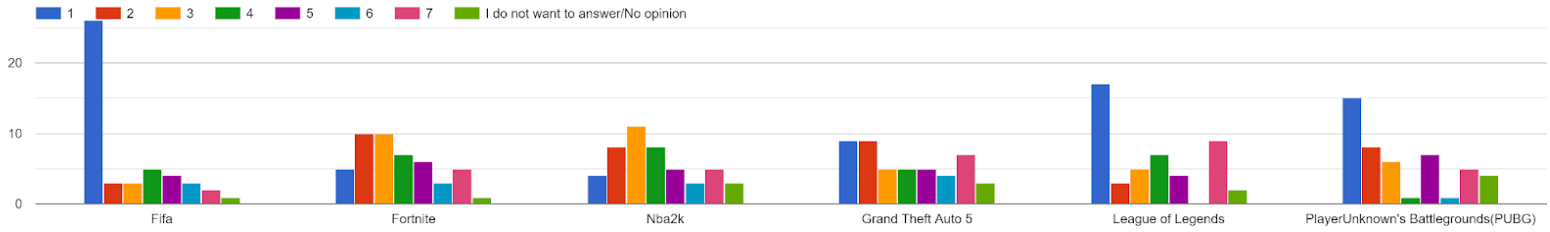
Social media preference to chat with you teammate

Copy

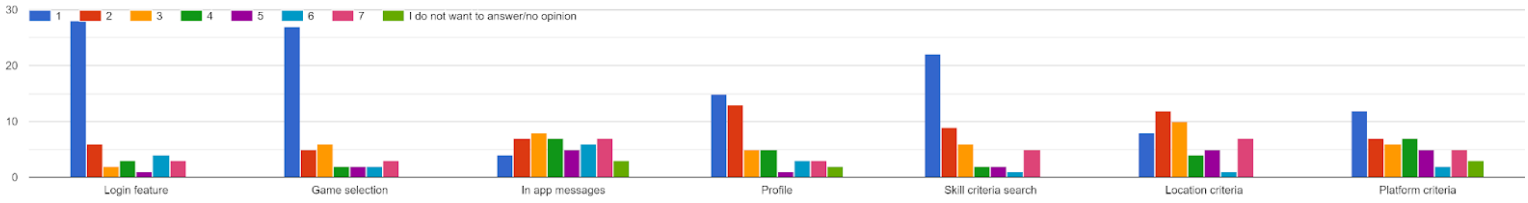
47 responses



Rank these games from 1-7(1 being the highest/most important and 7 being the lowest/least important)



Rank these features from 1-7(1 being the highest/most important and 7 being the lowest/least important)



Rank these features from 1-7(1 being the highest/most important and 7 being the lowest/least important)