Fyaz Qadir Ahmed Ikram 2021 BSc (Hons) ISD

TUS
Technological University of the Shannon: Midlands Midwest
Ollscoil Teicneolaíochta na Sionainne: Lár Tíre Iarthar Láir

An Android Mobile Application to Assist People with various disabilities and to connect them to an online buddy system

Fyaz Qadir Ahmed Ikram

Bachelor of Science (Honours) in
Internet Systems Development

2021

An Android Mobile Application to Assist People with various disabilities and connect them to an online buddy system

Fyaz Qadir Ahmed Ikram

K00237093

A Final Year Project submitted in partial fulfilment of the requirements of Technological University of Shannon: Midlands Midwest for the degree of Bachelor of Science (Honours) in Internet Systems Development.

Supervised by:

Pamela O'Brien

Month/Year

Ethical Declaration:

I declare that this project and document is wholly my own work except where I have made explicit reference to the work of others. I have read the Department of Information Technology Final Year Project guidelines and relevant institutional regulations, and hereby declare that this document is in line with these requirements.

I have discussed, agreed, and completed with whatever confidentiality or anonymity terms of reference were deemed appropriate by those participating in the research and dealt appropriately with any other ethical matters arising, in line with the TUS Research Ethics Guidelines for Undergraduate and Taught Postgraduate Programmes policy document.

____________________________ ____________________

Fyaz Qadir Ahmed Ikram [Date]

Acknowledgements

I would like to thank my supervisor, Pamela O’Brien, for giving me the opportunity to undertake this project. Her constant support and guidance were an invaluable resource throughout the project.

I would also like to thank my friends and classmates for all the help and support they have given me during this project and over the past four years.

Finally, I would like to thank my parents and my sister, Ikram, Nilofar, and Safaa who have been great to me throughout my whole life, and without whom none of this would have been possible.

Abstract

For the Final Year Project, the author has looked at the problem of social isolation and loneliness with people with intellectual and physical disabilities and decided to create a mobile application to have a buddy system wherein two parties (Buddies and people with disabilities) could interact and socialize with one another (Tough et al., 2017) ("Sage Journals", 2021) (Anderson, 2021) (Hodapp, 2020) (Raising Children Network, 2021).However, currently there are many e-buddy services online, but they all communicate via email, which in this day and age, is a lengthy process and could be very inefficient. All these services also do not provide any type of mobile application for buddying these types of parties. That is why the author has decided to create a Realtime Mobile Application which will help and assist people with various disabilities to overcome their solitude and loneliness and help them interact with peers or buddies on the mobile application. Any person (with or without disability) will be able to register themselves as a user and either get assistance or register as a buddy. Whenever a user register, they will be able to create a diary for availability to pair up with a buddy, which will get stored in a Realtime database. The app will also automatically gather the user's location from the geo location service API, which will allow both parties to arrange an appointment to meet up with the respective parties. The app would also incorporate a chat area for both parties to communicate between each other via text, or call.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
GLS	Geo-Location Services
SP	Shared Preferences
UI	User Interface
UE	User Experience
RDBMS	Relational Database Management System
P2P	Peer to Peer
iOS	iPhone Operating System
OS	Operating System
DBMS	Database Management System
AWS	Amazon Web Services
DB	Database
GPS	Global Positioning System
IDE	Integrated Development Environments

1 Introduction

The reason for this Project's development was to assist people with various disabilities (physical and mental) and help them socially interact with people without any disabilities using a real time buddy mobile application. The aim is to create an android mobile based application that would help people with disabilities to overcome their solitude and to interact with peers or buddies without much difficulty using a social online platform. This project will consist of an Android Application, a Firebase Database, a Chat Service, and a Geo-Location Service, while including accessibility features to cater for various (visual, hearing, etc.) impairments. As a huge issue among people with disabilities is social loneliness and solitude. The app will be able to buddy a person(s) with disabilities and a person(s) without disability together which will allow them to communicate via the mobile application.

The main objectives of this project were to develop a system capable of chat communication and that has the ability of pairing parties of people with and without disability together using Geo-Location Services, while helping people with disability overcome the loneliness and solitude. While working on this Project, it gave the author the opportunity to experience new technologies that he had not worked with before such as Android Studio.

The main functional objectives of this project are as follows:

- People with disability would be able to register and request for a buddy
- People without disability would be able to register and offer to be a buddy
- Both parties would be able to create their diary for availability
- Both parties would be able to go to the help page which would answer Frequently Asked Questions
- The app would contain accessibility features to cater for people with all types of disability (Hearing, Visual, Speech) Impairments
- The app would be able to identify the location of users using geo location services which allows users to find the buddies closest to them on a map
- The app would contain a chat area to communicate to different users via text or call
- The app would feature an online booking system via chat to meet up and socialize between users

- The app would be able to also provide an emergency bay that directly links to the local country's emergency service

The Authors reasoning for picking a project such as this one, is since his parents were former health professionals working in the area of helping people with disabilities. They would always consider the idea of creating a buddy program that could easily be accessible and efficient for its users. The Author is developing this application to help decrease the problem of social loneliness and exclusion among people with disabilities.

This project involved researching the topics of Social Isolation among people with disabilities to fully understand the area of research. The Author used this knowledge to pitch the idea of an Android Application to benefit people with disabilities. This project involved software elements to complete this application. Over the course of the year the Author experimented with different techniques and designs for the project to ensure the Author came out with the best solution for the user.

The main methodologies for research that will be used for this project are as follows:

- Internet - Will be used to research various aspects of the subject and identify further ideas for the mobile application.
- Library Resources (Books, Articles and Journals) - Will be used to research various aspects of subject and identify further ideas for the mobile application.
- Interviews - Will be used to better understand, and explore research subject opinions, behavior, experiences, and phenomenon and will be used to gather further requirements of the mobile application and understand different approaches of the subject.

Some research questions that the Author had in mind when researching about the area of research were as follows:

- What platform does the Author intend on developing this application?
- How can Geo Location be used to capture the nearest locations for their buddies?
- What is the framework of connecting google maps to the specific buddy nearest to another buddy?

- What database would be used to store the information on the application?
- What would the database be used to store (Kind of Data) on the mobile application?
- Would the data store on the mobile application hold any sensitive data (What would that data be)?
- What would the online chatbot incorporate and try to answer on the mobile application for its users?
- What security measures would be taken into consideration to verify people with and without disability?

This Document entails the steps the Author went through to bring the project to completion. The chapters include:

- Literature Review
- Analysis and Design
- Implementation
- Testing and Results
- Discussion and Conclusion

2 Literature Review

2.1 Introduction

What current applications exist and what services do they offer in regard to online buddying systems for people with disabilities?

Currently there are websites that provide various e-buddy services in Ireland such as St. Hilda’s and Best Buddies. These websites include friendships with peers, securing jobs, and living independently. The aspect of the buddy program on their websites are an element of their services and offer buddying services for people with intellectual developmental disabilities wherein people with and without disabilities can socialize and interact via emails. In this day and age buddying via email’s is very inefficient and slow. Although there are many social media platforms such as Snapchat, Facebook and Twitter which allow users to communicate and chat with different people there currently are no such online platforms which cater and accommodate people with various disabilities and provide a peer/buddy system to accommodate for people with various impairments. After conducting an interview with the Author’s mother, The Author has found that many people with both physical and mental disabilities do not prefer using platforms such as Facebook, Twitter, and Snapchat due to the fact that people with disabilities cannot communicate effectively and clearly to people without any disability. This demands the need to create an application that can open the window of opportunities for people with intellectual disabilities and people with physical disabilities, to communicate and socialize effectively online.

What would make the Authors application unique?

The Author has also looked and researched into if there are any mobile applications similar to these websites that provide these types of services, but currently no such applications exist. The Author has decided to create and provide these services as mentioned but would be implemented using a mobile application which would enable a wider audience as well as allowing people with all kinds of disability to interact and socialize with buddies in an efficient and enjoyable way. The mobile application would highlight various new technologies and features not implemented currently with any other online buddy systems, including geo-location services to connect both parties (People with/without disability) by nearest locations, chat area to communicate between both parties, one-on-one appointment arrangement system to meet up with the respective parties, and an emergency bay to connect the people with disability to the local emergency service in case of any predicaments.

2.2 Project Context

2.2.1 Disability

The main factor for this Android Application is developed around catering and helping people with disabilities buddy up with people without any disabilities. The social involvement of disabled people is inadequate and low; one of the reasons often overlooked but of great importance may lie in the disparate patterns of social interaction between the disabled and abled people (Liu et al., 2021, para.1). According to best buddy's annual survey, fifty-four per cent (54%) of the members with intellectual disability disorder showed that they feel more confident engaging in an online platform ("e-Buddies", 2021, para.6). The best buddy survey was conducted among college participants in the United States. The findings suggest that both the college students and the people with intellectual disabilities benefitted from the program. Both the groups reported that they enjoyed their experiences and engaging in friendship activities were mutually beneficial (Hardman & Clark, 2006, p.56-63).

There are two main categories of people with disabilities, which include people with physical and mental disabilities. Social Isolation and Exclusion is a huge concern for people not just with cognitive disabilities but also people with physical disabilities. According to Disabil Health J, People with a disability experienced loneliness, inadequate perceived social support, and social isolation at remarkably higher rates than people without any disability. People with physical disabilities are affected by loneliness and exclusion mainly due to the restriction caused by their limited movement, which affects them to interact or socialize with anyone. However, when it comes to people with mental disabilities, the main reason for their social isolation is their inability to communicate clearly and effectively.

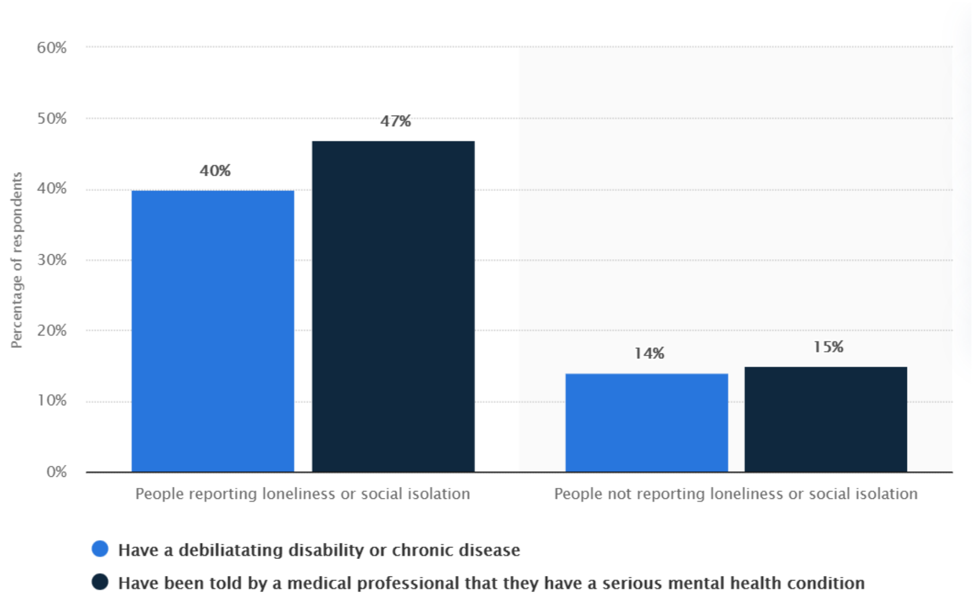
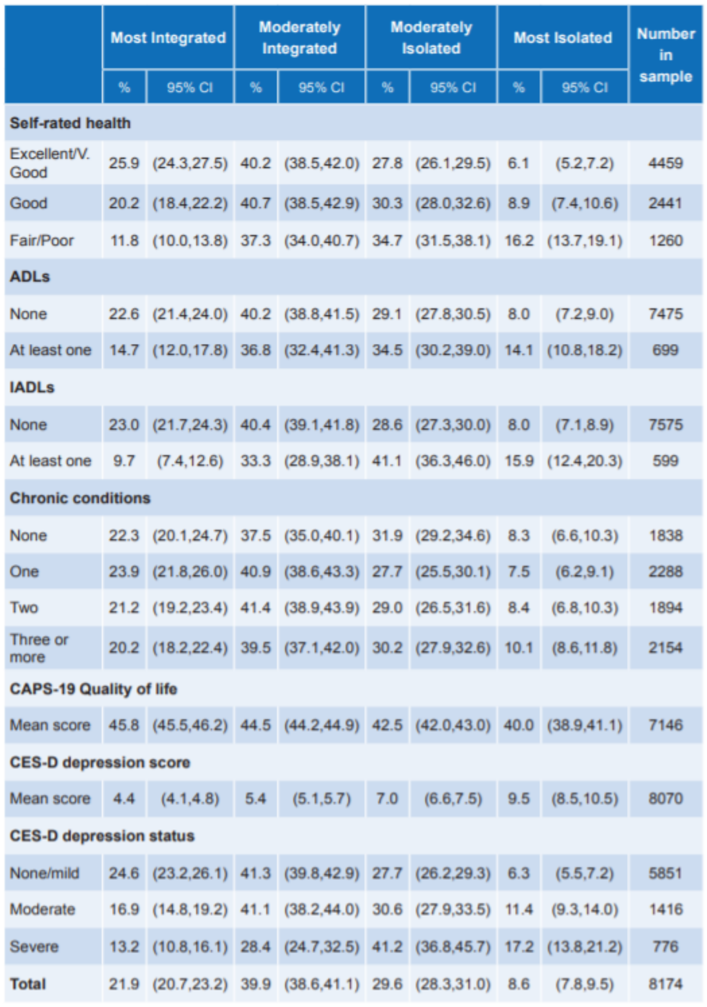


Figure 1 Proportion of people experiencing loneliness and social isolation in the U.S. in 2018, by physical or mental health condition

In a survey conducted by Statista in 2018, over 40% (2/5) of the US population who either have a debilitating disability or have been told by a medical professional that they have severe mental health conditions have reported that they suffer from social loneliness and isolation. In comparison, the people saying that they do not suffer from social isolation and loneliness in the US is significantly lower at a percentage of around 15% (3/20).



	Most Integrated		Moderately Integrated		Moderately Isolated		Most Isolated		Number in sample
	%	95% CI	%	95% CI	%	95% CI	%	95% CI	
Self-rated health									
Excellent/V. Good	25.9	(24.3,27.5)	40.2	(38.5,42.0)	27.8	(26.1,29.5)	6.1	(5.2,7.2)	4459
Good	20.2	(18.4,22.2)	40.7	(38.5,42.9)	30.3	(28.0,32.6)	8.9	(7.4,10.6)	2441
Fair/Poor	11.8	(10.0,13.8)	37.3	(34.0,40.7)	34.7	(31.5,38.1)	16.2	(13.7,19.1)	1260
ADLs									
None	22.6	(21.4,24.0)	40.2	(38.8,41.5)	29.1	(27.8,30.5)	8.0	(7.2,9.0)	7475
At least one	14.7	(12.0,17.8)	36.8	(32.4,41.3)	34.5	(30.2,39.0)	14.1	(10.8,18.2)	699
IADLs									
None	23.0	(21.7,24.3)	40.4	(39.1,41.8)	28.6	(27.3,30.0)	8.0	(7.1,8.9)	7575
At least one	9.7	(7.4,12.6)	33.3	(28.9,38.1)	41.1	(36.3,46.0)	15.9	(12.4,20.3)	599
Chronic conditions									
None	22.3	(20.1,24.7)	37.5	(35.0,40.1)	31.9	(29.2,34.6)	8.3	(6.6,10.3)	1838
One	23.9	(21.8,26.0)	40.9	(38.6,43.3)	27.7	(25.5,30.1)	7.5	(6.2,9.1)	2288
Two	21.2	(19.2,23.4)	41.4	(38.9,43.9)	29.0	(26.5,31.6)	8.4	(6.8,10.3)	1894
Three or more	20.2	(18.2,22.4)	39.5	(37.1,42.0)	30.2	(27.9,32.6)	10.1	(8.6,11.8)	2154
CAPS-19 Quality of life									
Mean score	45.8	(45.5,46.2)	44.5	(44.2,44.9)	42.5	(42.0,43.0)	40.0	(38.9,41.1)	7146
CES-D depression score									
Mean score	4.4	(4.1,4.8)	5.4	(5.1,5.7)	7.0	(6.6,7.5)	9.5	(8.5,10.5)	8070
CES-D depression status									
None/mild	24.6	(23.2,26.1)	41.3	(39.8,42.9)	27.7	(26.2,29.3)	6.3	(5.5,7.2)	5851
Moderate	16.9	(14.8,19.2)	41.1	(38.2,44.0)	30.6	(27.9,33.5)	11.4	(9.3,14.0)	1416
Severe	13.2	(10.8,16.1)	28.4	(24.7,32.5)	41.2	(36.8,45.7)	17.2	(13.8,21.2)	776
Total	21.9	(20.7,23.2)	39.9	(38.6,41.1)	29.6	(28.3,31.0)	8.6	(7.8,9.5)	8174

Table 1 Distribution of social isolation by key health and psychological wellbeing indicators

Furthermore, according to the findings from the Irish Longitudinal Study on Ageing (TILDA) as shown in Table 1, it displays the distribution of social isolation in relation to major health and psychological variables. A higher level of social isolation was linked to lower self-rated health. 16.2 percent (95 percent CI: 13.7-19.1) of participants with mediocre or poor self-rated health were socially isolated, compared to 6.1 percent (95 percent CI: 5.2-7.2) of those with outstanding or very high self-rated health (Loneliness, social isolation, and their discordance among older adults, 2021).

Considering all these factors, The Author has concluded the best way to socialize, interact, alleviate, and help people with disabilities overcome their solitude and loneliness is to create this buddy system using a mobile application.

2.2.2 Accessibility

The Author has defined that he would incorporate accessibility features into the mobile application. Accessibility is a critical feature in any application, but it is vitally important to integrate it into this project since it caters to and helps people with disabilities. Accessibility is the design of a mobile application for individuals with disabilities. The disabilities could include anything from hearing to mobility-related problems. The app will consist of accessibility features to cater to various hearing, visual, and speech impairments. According to Appoly, it is stated that around 15% of people globally suffer from some form of disability or impairment. Mobile applications need to implement accessibility features mainly because 1 in 7 people might not be able to use it because it would not regale any accessibility features ("The Importance of Accessibility in a Mobile App", 2021).

The Accessibility options that the Author has featured and make good use of within this mobile application are good mobility, clear vision, colour perception, and hearing, speech, cognition, and literacy assistance. These elements will be addressed and implemented by assigning assistance for keyboard interactions, configurable hand gestures and screen readers; the application would also support user customization within the mobile application allowing the user to be able to change settings depending on what impairments they may have. For people that may have low vision and hearing impairments customizable font sizes, magnification lenses, colour change customization, and high contrast settings would be used. The Author has included descriptive labels, screen readers, and audio descriptions of text for those people who are visually impaired (Accessibility and your app design, 2021).

The Author has included a good UI design with a modern and clear colour scheme that would enable the user to use and access the mobile application with ease and comfort. In terms of navigation within the mobile application, the Author has also created a straightforward workflow within the pages to get to the required page destination with the least number of clicks.

2.2.3 Geo Location

Another key feature that this application would incorporate into this mobile application is Geo-Location Services. A Geolocation API is a communication interface between an application (Client-Side) and an application (Server Side), recognizing and retrieving information about the client's geographic location. Geolocation APIs are called when there is a need to call and configure a web/mobile-based application programmatically based on the user's location. Common use cases include geo-tracking and implementing user security checks (What is a Geolocation API and how is it used?, 2021).

The Geolocation Service API does not hold any information within the API; however, it can accumulate existing information from the machine being queried (What is a Geolocation API and how is it used?, 2021)

The Author has integrated Geolocation services within this mobile application to find, search and connect to the buddy nearest to another buddy based on the closest geographical location. The Geo-Location Services would be implemented using the Geo Location API on Google that provides Libraries to call and use these functions to integrate it with a mobile application. This would be a massive feature on the mobile application as it would allow users to pair up with buddies closest to them, which would make it more accessible and convenient for the user.

2.2.4 Chat Services

Another main characteristic that would embody this mobile application is a chat service that will allow its users to communicate with different parties (people with and without disabilities) together. Chat service is an online service or technology that enables text messages to be translated in real-time between users (What is Chat Service - Helpshift, 2021).

The Author emphasizes putting this feature as the focal point on the mobile application solely since this feature will aim to help people with disabilities socialize and buddy up with people without any disability.

The application would first prompt the user to sign up and register using his/her email on the mobile application, which will create an account for the user and request the person(s) to log in. Using the details that were stored for registering with the account, the chat service would then register them onto the chat service platform enabling the user to find their friends using the find friends area via search or map, which in turn allows the user to connect and chat to their buddies and peers effectively.

The chat service that this application would comprise within this mobile application would include two main implementations for online communication: calling and texting. These options for exchange give the user different choices based on their preferred mode of social interaction. The Author would also implement video calling as an extra mode of communication for the users using the mobile application.

3 Technologies Research

3.1 Mobile Application Development

In today's world, technology has become a critical factor in everyday life. One of the most widespread components of technology these days is the use of smartphones/hand-held devices. Smartphone users have significantly surpassed the amount of desktop/laptop users. In a survey of smartphone users around the globe by Statista, in 2021, over 3.9 million people use smartphones in Ireland; it is expected that by 2026, that number is expected to go up to 4.41 million, which is a considerable proportion of the Irish population (Ireland mobile internet users 2016-2026 | Statista, 2021). As time progresses, the popularity of smartphones grows worldwide as well. A 2021 survey showed that 7.1 billion users around the world use a smartphone today. According to DataReportal, that is roughly 73% of the world population (Forecast number of mobile users worldwide 2020-2025 | Statista, 2021). Even though smartphones are popular today, the two most established platforms are Google's Android Operating System and Apple's iPhone Operating System. Due to the following reasons, the Author has decided to develop a mobile-based application.

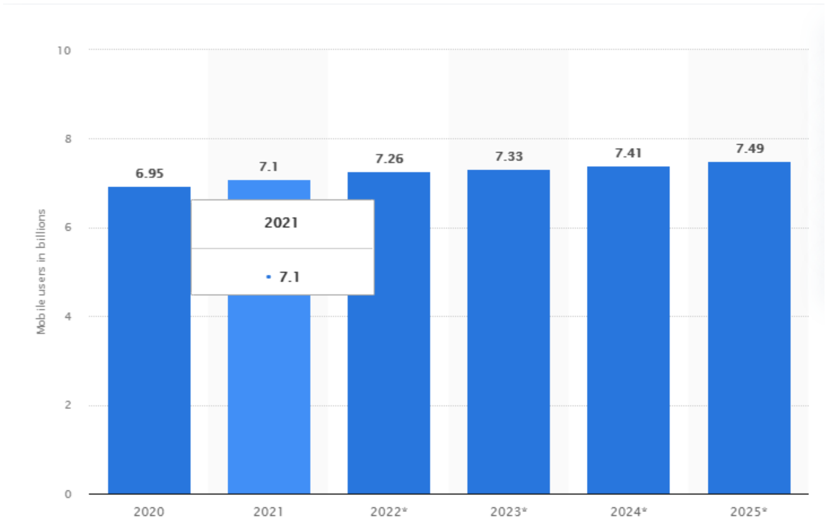


Figure 2 Number of mobile users worldwide from 2020 to 2025

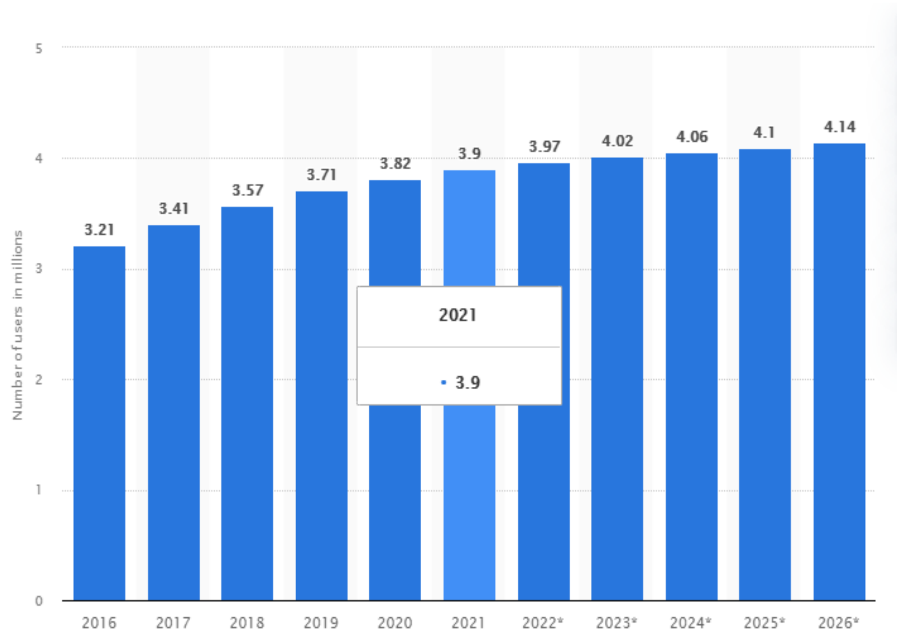


Figure 3 Number of mobile internet users in Ireland from 2016 to 2026

3.1.1 Android

Android is an operating system formed on the Linux Operating system and is a free, open-source platform for various mobile devices such as tablets, smartphones, and smart devices. Android ranks as the most used mobile application platform. According to Statcounter, in September 2021, Android held a 72.44% portion of the mobile phone market (Android Operating System Market Share Worldwide | Statcounter Global Stats, 2021). Android was founded by Open Handset Alliance, which Google and other Multinational Companies led. Android allows its developers to only develop in Android, which could be implemented and used across multiple devices powered by Android, making it efficient and easily accessible. The first pre-release of the Android Software Development Kit was released in 2007 by Google. Then in 2008, the stable version was released, which was for commercial use. Google then announced the next Android Version, 4.1 Jellybean, in 2012. Jellybean was an updated version for Android, which provided better UI in terms of Performance and Functionality (Android - Overview, 2021).

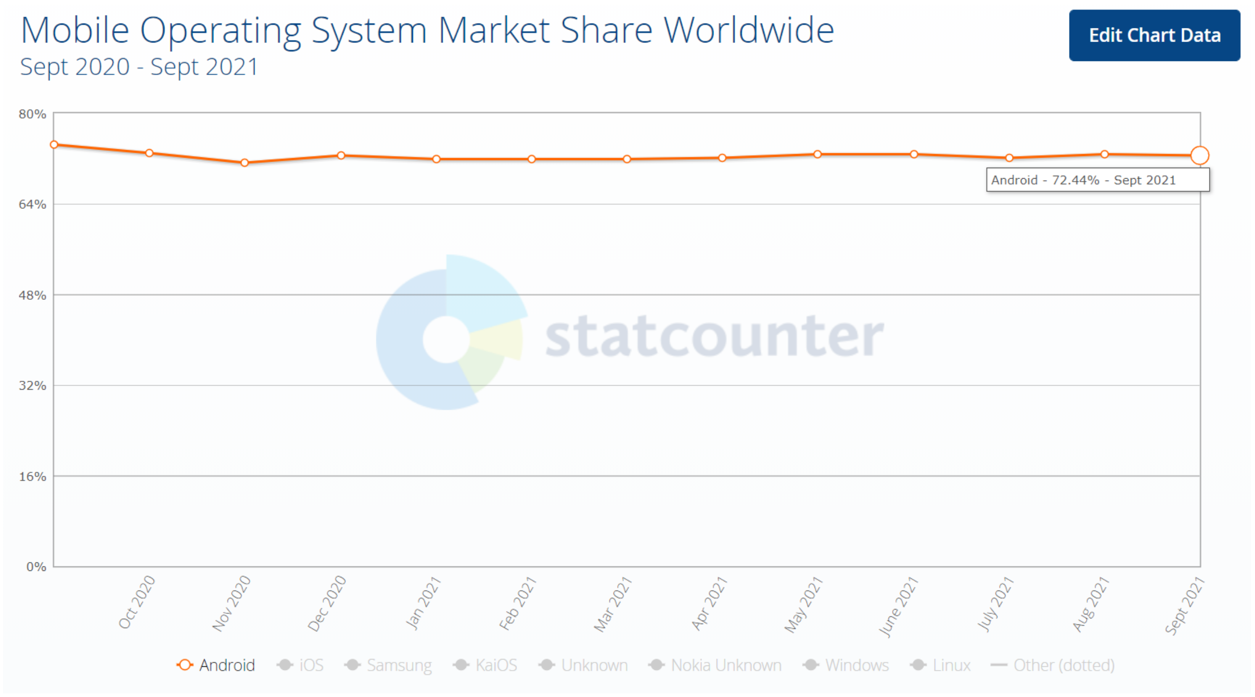


Figure 4 Android Operating System Market Share Worldwide

There are several reasons why Android Operating Systems have become consequently popular. Android Operating System provides an Open-Source Environment for its users. They have a prominent developer and Community Reach, have Increased Marketing, provide Inter-App Integrations, allow for Reduced Cost of Development, have a High Success Ratio, and supply a Rich Development Environment (Android - Overview, 2021).

Android is a powerful Operating System that provides and supports a vast number of features. Some main features and advantages of the Android Operating System include the following:

- UI Design – Provides a modern and intuitive user interface.
- Connectivity – Provides connections to various technologies, mainly Bluetooth, Wi-Fi, and NFC.
- Storage - SQLite is a software package that provides a Relational Database Management System.
- Media Support – Supports various media extensions, mainly MPEG-4 SP, MP3, WAV, JPEG, PNG, GIF, and BMP.
- Messaging – This Allows for MMS and SMS type messaging.

- Multi-Touch - Android has native support for multi-touch which recognizes more than one point of contact with the device surface for easy access.
- Multi-Tasking - Users can jump from one application to another and can also concurrently run various applications at the same time.
- Widgets - are resizable, so users can enlarge them to identify more content or shorten them to save lots of space.
- Multi-Language - Supports using different languages by internationalization.
- GCM - Google Cloud Messaging (GCM) is a service that lets developers send short messages to users on Android devices without having an exclusive sync solution.
- Wi-Fi Direct - A technology that lets apps discover and pair directly over a high-bandwidth P2P transmission.
- Android Beam - A NFC-based technology that allows users to share immediately by touching two NFC-enabled phones together.

(Android - Overview, 2021)

However, looking at the benefits and an immense number of features that Android provides, there are also many downfalls in using the Android Operating System. Some applications on the Android Market can contain viruses and bugs, a considerable amount of background processes can lead to the battery quickly draining, advertisements would always be on display with these applications, they contain a high device fragmentation, and they have low-security policies on their Operating Systems (The Pros and Cons of Android, 2021) (Advantages and Disadvantages of Android & iOS, 2021).

3.1.2 iOS

Apple iOS is an Operating System for iPhones, iPad and other smart apple devices based on the Mac Operating system. iOS ranks as the second most used mobile application platform, right behind the Android Operating System. According to Statcounter, in September 2021, Apple iOS held a 26.75% portion of the mobile phone market (iOS Operating System Market Share Worldwide | Statcounter Global Stats, 2021). Apple Operating System was first released in June 2007 by Apple. Apple released their iPhone Software Development Kit in 2008, which allowed developers to create applications for the Apple platform. Apple published their iOS version 12 in 2018, which paved advancements in design and functionality (Apple iOS, 2021).

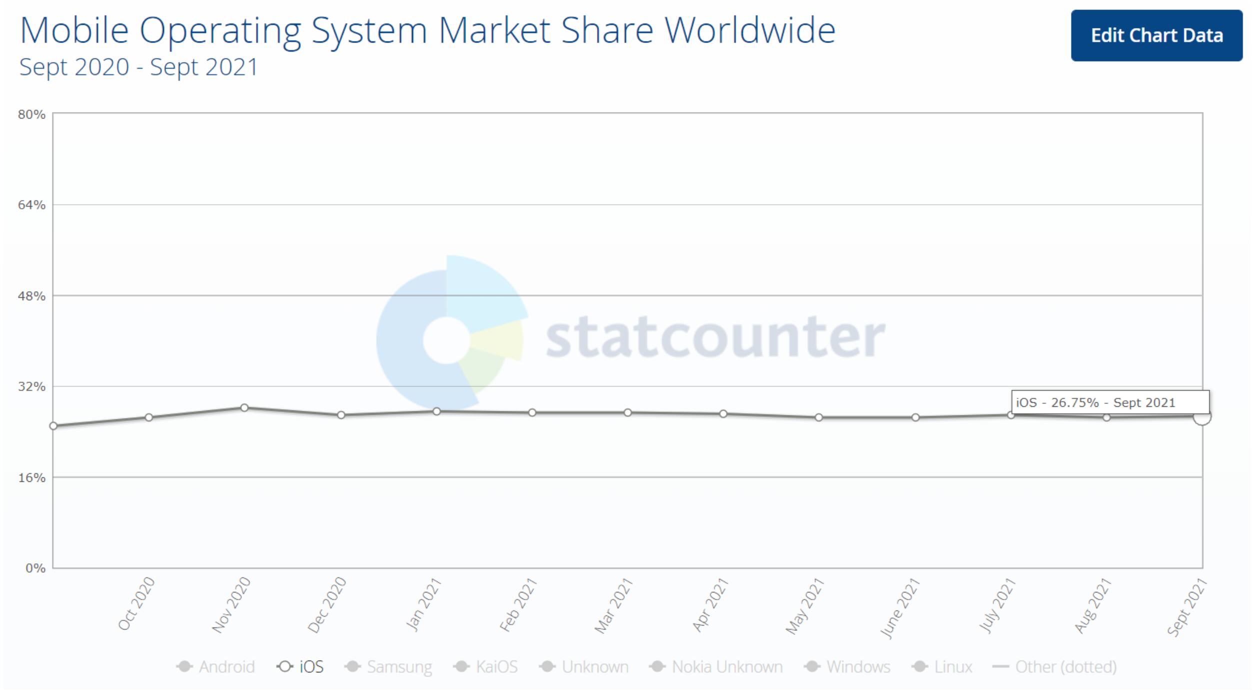


Figure 5 iOS Operating System Market Share Worldwide

There are multiple reasons why the Apple Operating System has become so popular and globally recognized. According to Investopedia, Apple sold nearly 218 million iPhones within the year 2018. Estimates show that iOS products have made around $1 Trillion of Revenue for Apple. iOS Operating System provides higher revenue, Higher return in investments, takes less time to develop, offers a fluid user experience to consumers, and has better security (Apple iOS, 2021) (Top Reasons to Choose the iOS Platform for Mobile App Development, 2021).

There are many features that the iOS Operating System provides and supports. Some of the features and advantages of using an iOS Operating system are as follows:

- UI Design – Provides a modern and fluid user interface.
- Connectivity – Provides connections to a variety of technologies, with the main ones being Bluetooth and Wi-Fi.
- Storage – A Solid-State flash memory, which stores all the data.

- Media Support – Supports various media extensions, with the main ones being H.264, MPEG-4.
- iMessaging – Allows for MMS and SMS type messaging
- Multi-Touch - Android has native support for multi-touch which recognizes more than one point of contact with the device surface for easy access.
- Multi-Tasking - Users can jump from one application to another and can also concurrently run various applications at the same time.
- Apple Pair – Allows the user to pair and send data to other apple users.

Nonetheless, with all the benefits and features that the iOS Operating System provides, there are also disadvantages to the iOS Operating System, which are it is not compatible with any other types of platforms or devices, it has no widget support for iOS apps, does not provide NFC and is not radio built in. The app size is usually too big, consuming too much storage and space (Advantages and Disadvantages of Android & iOS, 2021) (Advantages and Disadvantages of iOS | IOS App Development Service, 2021).

3.1.3 Preference of Platform

The Author, after looking and researching the Android and iOS Operating Systems. The Author has decided to approach this project and develop this application using the Android OS. The main reason for choosing the Android OS is because it has a wider market share compared to iOS OS, which in turn allows the Author to reach a wider audience. Android OS also provides an open-source environment in comparison to iOS OS, which is a closed source. There is also more customization allowed when developing on Android OS compared to customizing on iOS OS, which does not offer much customization on their platform. The Author would also be using the Android OS over the iOS OS due to the higher market share which would increase the target audiences using the mobile application with Android compared to iOS applications (Android vs iOS - Which mobile platform is better in 2020? 2021).

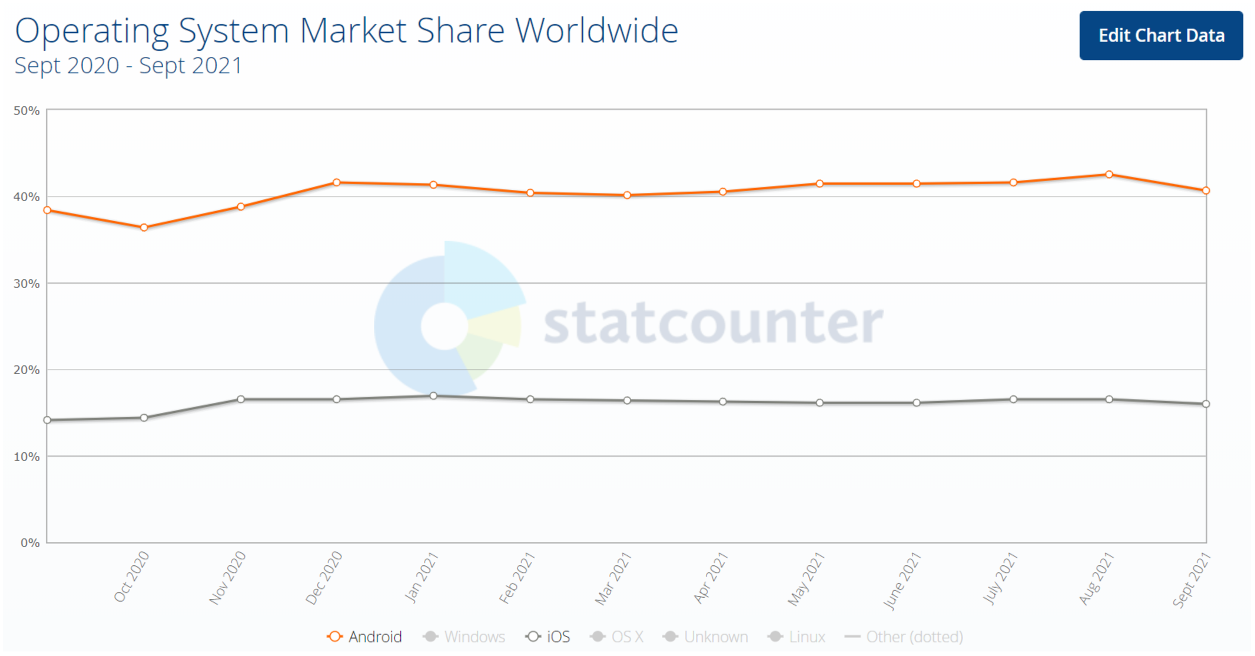


Figure 6 Android Operating System Market Share Worldwide vs iOS Operating System Market Share Worldwide

3.1.4 Java versus Kotlin

There are two main programming languages used in developing an Android Application on Android Studio: Java and Kotlin.

Java is a Programming Language developed in 1995 and emerged in 1995 by James Gosling at Sun Microsystems, which Oracle then acquired in 2009. Java is an object-oriented, open-source and general-purpose language. Java also works on most platforms and devices (Kotlin vs Java: the 12 differences you should know, 2021).

Kotlin is also a Programming Language but is very new compared to Java, as it was first introduced in 2016. Kotlin also supports various Libraries and Frameworks made from Java that are compatible with Kotlin Projects. Kotlin is also known as the Updated Java Programming Language for Android Development which allows for cleaner, more accessible, and faster compilations. It contains a mixture of functional and Object-oriented Programming (Kotlin vs Java: the 12 differences you should know, 2021).

Java and Kotlin are now the two most prominent Programming Languages for Android Development. The Author will conduct and present the main key differences between the two Languages and then select which language would be used to develop the mobile application. The main differences between Java and Kotlin Programming are as follows:

- Code – Regarding code, Kotlin requires less code and is very concise, whereas, in Java, it requires more lines of code than Kotlin.
- Functional Programming – In aspects of Functional Programming, Kotlin contains a combination of both functional and object-oriented programming, but in comparison, Java is only limited to object-oriented programming.
- Primitive Data Types – In Kotlin, once a primitive data type is initiated, it would be treated as an object. In contrast, in Java, variables of primitive types are not considered objects but are defined as Java data types.
- Wildcard Types – In Kotlin, it does not provide any form of wildcards, but on the other hand, Java offers and provides wildcards.
- Public Fields – Kotlin does not offer public fields, whereas public fields are available in Java.
- Data Classes – Kotlin imparts a more effortless path to create classes to store data by using the “data” keyword in the class definition. Still, in Java, Developers must initialize the fields to store the data, such as using the getter and setter functions and other functions.
- Casts – Smart cast features cater cast Checks by Kotlin, which automatically manage redundant cast while on the contrary, Developers much check the variable types per the operation.
- Checked Exceptions – Checked exceptions are not supported on Kotlin, whereas it allows for checked exceptions in Java.
- Null Safety – In Kotlin, it is impossible to attribute null values to variables or objects by default. Still, Java allows NullPointerException, enabling developers to assign a null value to any variable.
- Extension Functions – Kotlin permits the developers to extend the functionality of classes without trying to inherit from a class. In contrast, Java requires creating new classes and inherit the functions from the parent class to extend the functionality of an existing class.

- Coroutines Support – Kotlin provides coroutines assistance. Coroutines are stackless, which permit developers to write code, suspend the execution and resume it later. Java, on the other hand, enables the formation of several background threads when handling lengthy operations.
- Implicit Conversions – Kotlin does not support implicit widening conversions, while Java allows for implicit conversions, which means that's developers do not need to perform explicit conversions.

(Kotlin vs Java: the 12 differences you should know, 2021).

3.1.5 Preference of Programming Language

Making comparisons on both the Java and Kotlin Programming Languages for Android Development. The Author has culminated to develop this mobile application using the Java Programming Language. The Author has decided to go with Java over Kotlin for many reasons, for the first reason being that Java has faster and cleaner builds compared to Kotlin. (Lastovetska, 2021), also, there is a broader community for Java than Kotlin; hence it offers more resources such as articles, websites, journals, and books. Another reason why the Author has concluded to develop this mobile application in Java is due to his proficiency in this programming language.

3.2 Databases

Databases are a huge part of any mobile application. A database is a structured gathering of different information, usually stored on a computer. A Database Management System (DBMS) generally runs a database. A DBMS is an interface that connects the database with the users or applications to retrieve, update, create or delete data allowing them to organize and manage the information. The database, the DBMS, and the applications that use them all work together and form a whole database system (What is a database? 2021).

There are many types of databases that are used, but the most popular of which are as follows:

- Relational Databases – In the 1980s, relational databases became the standard. A relational database's items are structured into tables with columns and rows. The most effective and versatile approach to access structured data is through using a relational database.

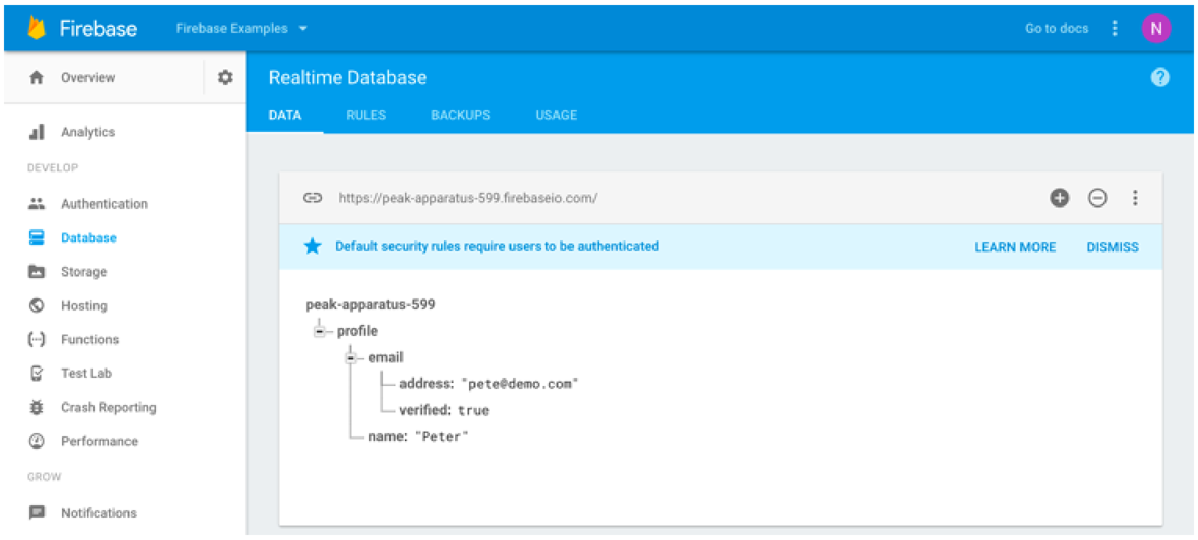
- NoSQL Databases – Unstructured and semi-structured data can be stored and manipulated in a NoSQL or non-relational database. As online applications became more frequent and complicated, NoSQL databases became increasingly popular.
- Cloud Databases – A cloud database is a structured or unstructured collection of data stored on a non-public or public cloud computing platform. Traditional and database as a service are the two sorts of cloud database models (DBaaS). A service provider with DBaaS handles administrative and maintenance tasks.

(What is a database? 2021).

Android Mobile Applications can use various databases to implement and use when developing their applications. The Author has chosen to research the three most popular databases: Firebase, MySQL Lite, and AWS DynamoDB. After looking in-depth at the three databases, the Author would then decide on which database would be used when developing the mobile application.

3.2.1 Firebase Cloud Database

Firebase is a real time database feature that is very efficient and easy to use. It is a NoSQL cloud-based database which stores and sync's data. The data always remains available and synced to the developer even when the application is offline. Firebase is also stored in a JSON format (Firebase Realtime Database | Firebase Documentation, 2021) (Khawas and Shah, 2018)
Advantages and Disadvantages of Firebase (Firebase Pros and Cons: When You Should and Shouldn't Use Firebase | OSDB, 2021).



https://peak-apparatus-599.firebaseio.com/

Default security rules require users to be authenticated

```
peak-apparatus-599
  profile
    email
      address: "pete@demo.com"
      verified: true
    name: "Peter"
```

Figure 7 Firebase Database Structure of app

The Firebase Real-Time database provides many key capabilities and advantages such as its running in real-time which permits for data synchronization for any changes in the data which could be received within milliseconds up the update. Also, Firebase applications remain active even when the application is offline due to the SDK requesting the data to disk. Firebase in addition is accessible from various client devices by directly allowing it to connect to client devices by security and data validation rules. Firebase further authorizes to scale across multiple devices databases by subscribing to a free blaze plan on Firebase. Firebase is also very easy to configure for android mobile applications and is not complex. Looking at figure 7 we can also see that Firebase has a very structured and ordered layout when it comes to displaying the data (Firebase Realtime Database | Firebase Documentation, 2021) (Khawas and Shah, 2018)
Advantages and Disadvantages of Firebase (Firebase Pros and Cons: When You Should and Shouldn't Use Firebase | OSDB, 2021).

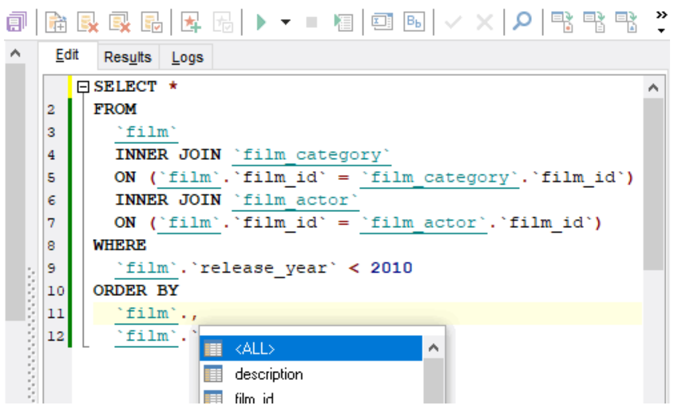
However, with all the capabilities and advantages that Firebase has. Some disadvantages of Firebase include it not having the ability to manage immense and complex queries, limited data migration transfer, and it being very android centered (Firebase Pros and Cons: When You Should and Shouldn't Use Firebase | OSDB, 2021).

3.2.2 MySQL Lite Database

MySQL Lite is a C-Language Library that performs a small, fast, high reliability and whole featured Structured Query Language (SQL) database engine. The most popular type of database used in Android development is the MySQL Lite. For Android Development, MySQL Lite is usually the default database (SQLite Home Page, 2021)

One advantage of MySQL Lite is that the database is very light and compact. Also, it offers better performance for writing and reading operations which is 35% faster than the default file system. There is also no need for installation needed which makes it very easy to use. It also offers excellent reliability by continuously updating data so little or no data is lost in the event of a crash or power failure. MySQL Lite is accessible through a wide variety of third-party tools and is also portable across multiple 32-bit and 64-bit operating systems (SQLite Advantages and Disadvantages - javatpoint, 2021).

Nevertheless, MySQL Lite has some downfalls as well. MySQL Lite is usually used to handle low to medium-sized traffic HTTP requests. The database size is restricted to only 2GB in most cases which could be a nuisance if the application being developed requires more data to be stored. MySQL Lite also requires developers to have in-depth knowledge of using SQL and understanding its syntax. (SQLite Advantages and Disadvantages - javatpoint, 2021).



```
SELECT *
FROM
  `film`
  INNER JOIN `film_category`
  ON (`film`.`film_id` = `film_category`.`film_id`)
  INNER JOIN `film_actor`
  ON (`film`.`film_id` = `film_actor`.`film_id`)
WHERE
  `film`.`release_year` < 2010
ORDER BY
  `film`.,
  `film`.`
```

Figure 8 MySQL Lite Query

3.2.3 AWS DynamoDB Database

Amazon DynamoDB is a wholly managed, serverless NoSQL database built to run high-performance applications of any size. Amazon DynamoDB comes with built-in encryption, automated backups, memory caching, and data expertise tools (Fast NoSQL Key-Value Database – Amazon DynamoDB – Amazon Web Services, 2021).

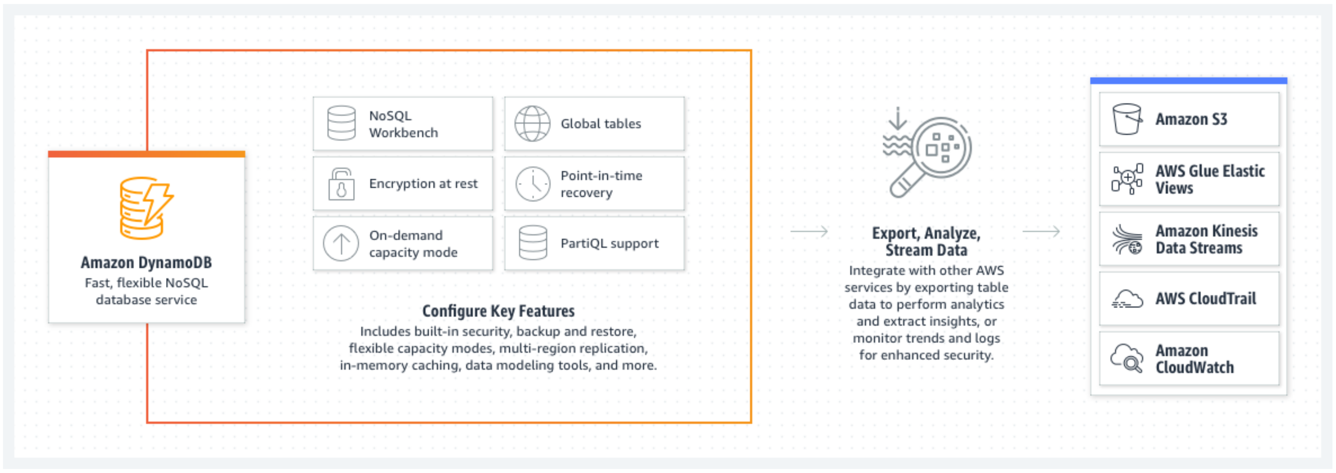

Amazon DynamoDB
Fast, flexible NoSQL database service
NoSQL Workbench
Global tables
Encryption at rest
Point-in-time recovery
On-demand capacity mode
PartiQL support
Configure Key Features
Includes built-in security, backup and restore, flexible capacity modes, multi-region replication, in-memory caching, data modeling tools, and more.
Export, Analyze, Stream Data
Integrate with other AWS services by exporting table data to perform analytics and extract insights, or monitor trends and logs for enhanced security.
Amazon S3
AWS Glue Elastic Views
Amazon Kinesis Data Streams
AWS CloudTrail
Amazon CloudWatch

Figure 9 Features of Amazon DynamoDB

Some key advantages of using the Amazon DynamoDB database is that it offers and provides scalability, which means that users can store infinity quantities of data according to their needs. Amazon DynamoDB also allows for Data Replication, managed across multiple availability zones in a region or across multiple areas. It is a schemeless database, which is exemplary for IoT, mobile and gaming. It is also very secure and monitors and alerts Database Threat Detections, Customizable traffic filtering and more (DynamoDB, 2021).

Yet, Amazon DynamoDB being some powerful and efficient as a database, there can be disadvantages that come with them. Some workloads are not suitable for Amazon DynamoDB, for instance, when menu-items transactions are required, when complex queries are essential and when real-time data on necessary data is needed (DynamoDB, 2021).

3.2.4 Preference of Database

Evaluating and inspecting the three different databases that the Author has proposed, The Author will use the Firebase Real-Time Database for many reasons. The first reason is that Firebase allows for more easy sharing of data, and secondly because it can support various real-time features like notifications, chats, and real-time feeds. Another reason why the Author has purposed on using Firebase as the database used to develop this application is it allows for integration with other tools, such as Google Ads, Data Studio and Google Analytics, which in turn guarantees better user experience and smarter marketing feedbacks such as collecting information on the target audiences using the mobile application and also marketing information to see how many people have viewed and used the mobile application within different timeframes (Firebase Realtime Database | Firebase Documentation, 2021) (Khawas and Shah, 2018).

3.3 Technology for Geo-Location Services

As mentioned previously, a critical factor that this application would incorporate would be using the Geo-Location API to allow users to connect to buddies by nearest locations. There are many ways this feature can be achievable, but the Author would use device-based collections. A device-based collection is banked on using cellular networks and GPS, which allows it to be more accurate in areas with higher population density since there is a narrower triangulation distance. The accuracy, on the other hand, decreases when the population density falls. There are generally delays or pauses in data in these situations. This, in turn, escalates the margin for error. People can use (and be used by) these services to find people's general location via GPS-tower-device triangulation as long as location-based services are enabled and have a GPS chip and a mobile network signal. However, users must allow location detection on each device and each application for this feature to work (What is Geolocation? How it Works & Why it Matters | Gravitate, 2021).

There are various Geolocation API's Properties that can be used to get the user's location on the map, such as the getCurrentPosition(), watchPosition(), and clearWatch(). The Author will use the getCurrentPosition() property from the Geo-Location API mainly because the mobile application will only need one position to get the user's base location and not continuously change location as the user is travelling to different places. The getCurrentPosition() property fetches the device locations data and stores it in latitude, longitude and accuracy into a position object which can then be used in a callback function. It is also possible to add arguments to specify the accuracy, timeout value and the maximum age to cache position data (What is a Geolocation API and how is it used? 2021).

3.4 Technology for Accessibility Features

Another prominent feature the Author has built into this mobile application is using accessibility features to cater to people with disabilities. The Author has increased text visibility and used extensive and straightforward controls for people with visual impairments to see clearly, which would be achieved by going into the UI design on Android Studio and making adjustments. Secondly, the Author has implemented a screen reader for people having difficulties looking and interacting with online content; an Audio reader is also incorporated into this mobile application to help people with visual impairments. On-screen text and fingerprint gestures will be implemented to help people with any speech and cognitive impairments. These features are all implemented using the Accessibility service that Android Studio provides. The Author creates a button on the mobile application that would directly link the user to the device settings on the mobile device and allow the user to get accessibility features on the mobile application. The accessibility service, which connects to the android-based accessibility menu, will also enable users with disabilities to take screenshots, lock screens, open google assistant, open quick settings and notifications, control volume, and control brightness by allowing the user to use hand gestures and easily accessible buttons. Another prominent feature implemented into the mobile application is the Dark/White mode toggle button, enabling users to toggle between these two different themes and access their preferred version of themes based on their preference. Last but not least, the Author has designed a workflow before the development phase to plan the least number of clicks to get to each page to provide the most enjoyable experience for the user (The Mobile Accessibility Landscape - Level Access, 2021) (Use the Accessibility Menu - Android Accessibility Help, 2021).

3.5 Technology for Chat Services

The chat service for messaging via the mobile application between buddies is also an important feature that would be consolidated within application. The Author would do this using Android Studio with first creating the Messaging User Interface, then creating the view which connects the messages with the UI design. After that the Author would create an activity to allow for communication. Once all these steps are completed the Messaging system will work and allow users/buddies to interact and communicate via each other via messaging. After that the Author would then implement the calling features that would enable users/buddies to call other buddies and speak with them, this would be done by granting permissions to allow the application to gather the user's contact details, followed by implementing the connection service, to handle all the common calling scenarios such as starting, holding, and ending calls. After the Author would incorporate mandatory chat features included in all mobile applications chat services such as offline support, link previews, commands, reactions, attachments, edit messages and threads. (Android chat tutorial: How to build a messaging UI - Sendbird, 2021) (Build a calling app | Android Developers, 2021) (Android Chat Getstream, 2021).

3.6 Conclusion

This research review's purpose is to help the reader understand different aspects posed by the research on social loneliness and isolation for people with both physical and mental disabilities and also to find the best approaches on how to build a mobile application to successfully cater for an application to allow people with various disabilities to communicate effectively to people without any disabilities. This is significant because people suffering from loneliness and social isolation do not have such services or applications to help them overcome their solitude. Most of the research was conducted from various websites and articles that gave an in-depth insight into the research context and technological research. After conducting this research, the Author has made critical decisions on how the application would be developed. The Author would develop this application on Android platform using Java and XML and incorporating Firebase Real-Tine Database and other technologies such as accessibility, geolocation, and chat services into the mobile application. More research and testing are required to better understand why there are still no applications developed to cater to these kinds of services. It is essential to conduct more studies on the results and why people with disabilities have high social loneliness and solitude rates.

4 Analysis and Design

4.1 System Overview

This project is comprised of various designs that incorporate multiple software aspects which in turn require separate phases of development. An overview of the project can be seen in the figure below.

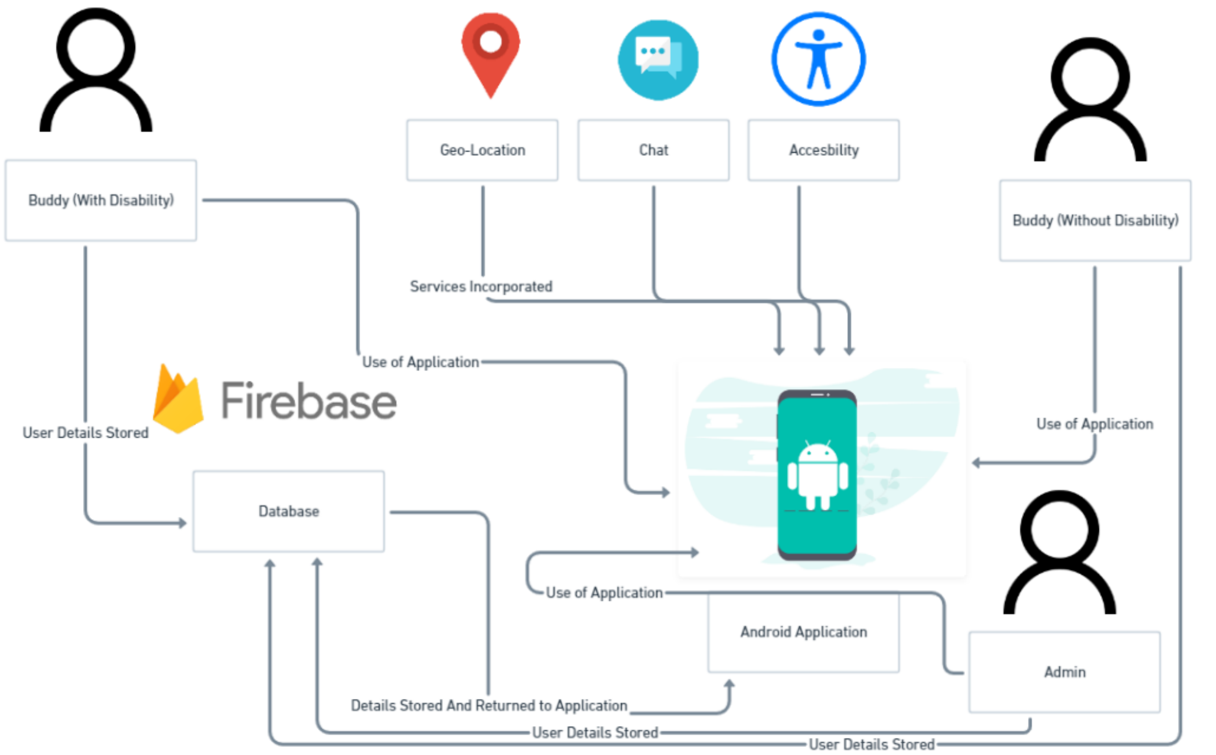


Figure 10 System Overview

This system would be developed using software components within this android mobile application. Each aspect of the project is essential to make the project function and operate successfully. The system contains a Firebase database and an Android application that would incorporate Geo-location, Chat and Accessibility features. The buddy with or without a disability can register on the application and then request the buddy to enable their location service, which gets stored in the Firebase real-time database. Then after registering, the buddy would be able to log in using the details provided to sign up by the system retrieving the information stored from Firebase. The buddy would then be able to add information to their profile for other buddies to read. The application would also let buddies view a map of people using the mobile application and locate the buddies closest to them and send an invite to pair up with the buddies. After the buddy accepts the invite, the other buddy would then be able to chat via the mobile application and make calls to communicate with each other. For the buddy having disabilities, the mobile application would allow the buddy to access various accessibility features to their liking and aid to use the application more effectively and clearly, for people with different hearing, visual and speech impairments.

4.2 Non-Functional Requirements

The Author has also taken into consideration the following non-functional requirements to develop this mobile application which are:

- Availability – The Author has decided for this application to be operational 24/7 and have minimum idle time by reducing the number of bugs within the application
- Performance – The Author has also considered allowing multiple users access this application simultaneously and having efficient and fast response times when using the mobile application
- Security – The Author has also implemented security and safety requirements to secure the users confidential data in the cloud using firebase cloud database and authentication
- Usability – The Author has also incorporated a friendly and ease of use interface that allows the users to seamlessly interact with the mobile application

4.3 Functional Requirements

The Author has implemented and taken into consideration the following functional requirements to develop this mobile application which are:

- Sign up as a buddy (Person with or without disability – 2 different roles)
- Login
- Logout
- View Personal Profile
- View Other Buddies Profile
- Edit Profile
- Add Diary to Profile
- Find a Buddy via search on contact list

- Search for a Buddy via Geo-Location Services
- Chat with a buddy via message
- Chat with a buddy via call
- Enable Accessibility Features
- Accept Buddy Requests
- Decline Buddy Requests
- Find Suggested Buddies near you
- View Frequently Asked Questions
- Add Attachments and Book Appointments via chat
- File Complaints and Report Abuse
- View Admin Dashboard
- View Reports and Abuse Complaints
- View and Remove Users

4.4 Android Application Wireframe Prototypes

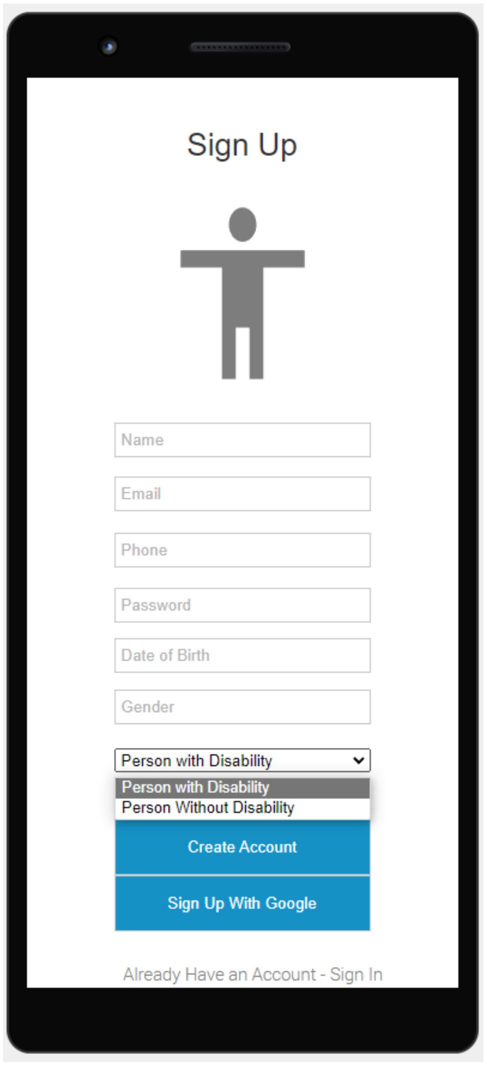


Figure 11 Sign Up Prototype Design

Figure 11:

This is the mobile application's sign-up screen. It will allow the user to sign up with their details to create an account, the user can also sign up with a google account if they have one. If the user already has an account the user can click on the "Already Have an Account – Sign in" Link which would redirect the user to login.

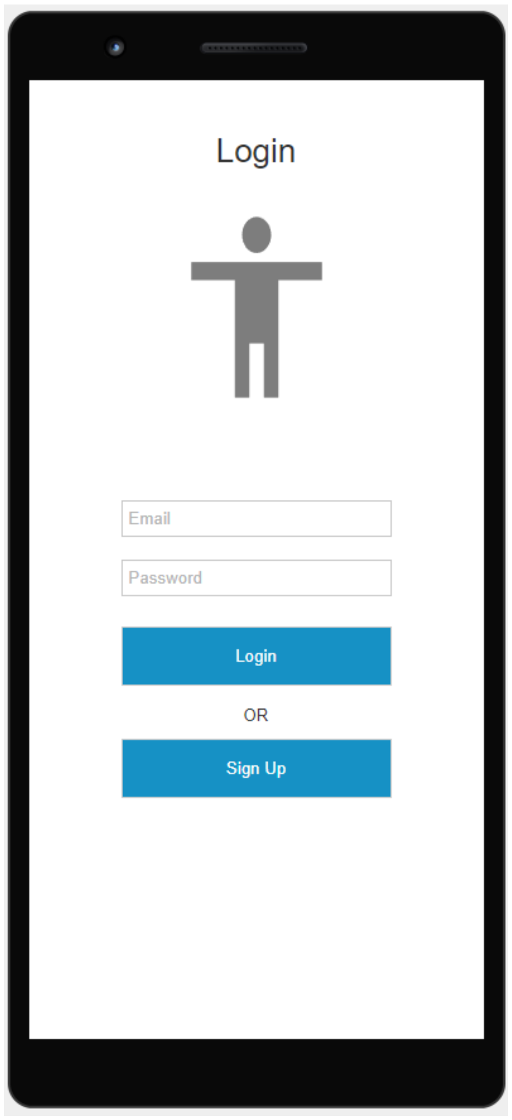


Figure 12 Sign in Prototype Design

Figure 12:

This is the mobile application's login screen. It will allow users to login with their credentials from either creating an account previously within the application or signing in with their google accounts.

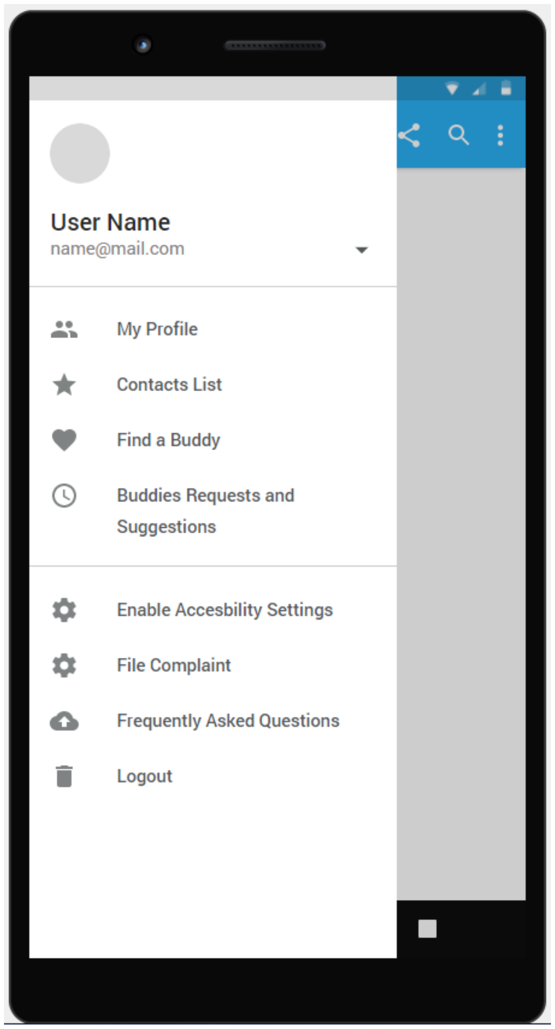

User Name
name@mail.com
My Profile
Contacts List
Find a Buddy
Buddies Requests and Suggestions
Enable Accesbility Settings
File Complaint
Frequently Asked Questions
Logout

Figure 13 Navigation Menu Prototype Design

Figure 13:

This is the mobile application's navigation pane screen. This screen allows users to view the different pages within the mobile application and navigate to the required pages when clicking on the specific links which redirects them to the specific page.

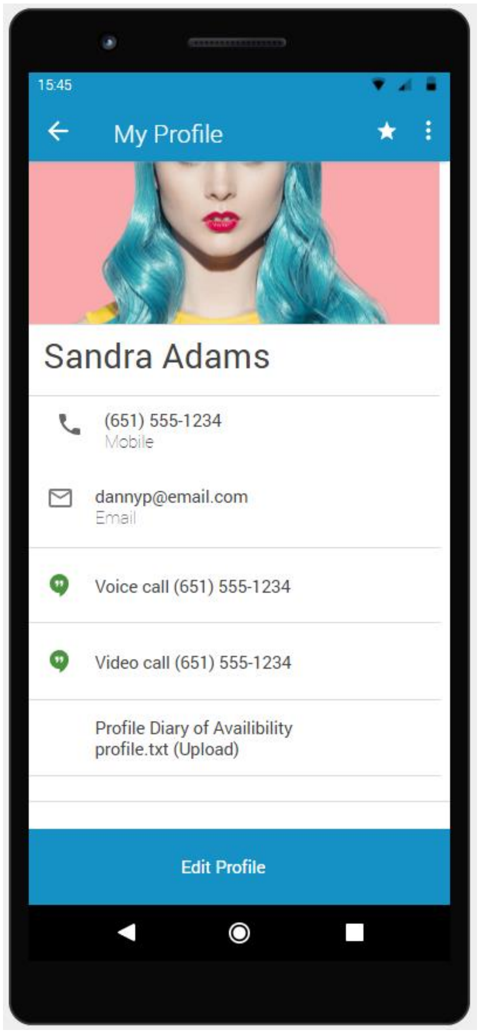


Figure 14 Personal Profile Page Prototype Design

Figure 14:

This is the mobile application's personal profile screen. Here users can view their profiles and edit their contact information and also upload a brief txt diary file of their personal diary and information.

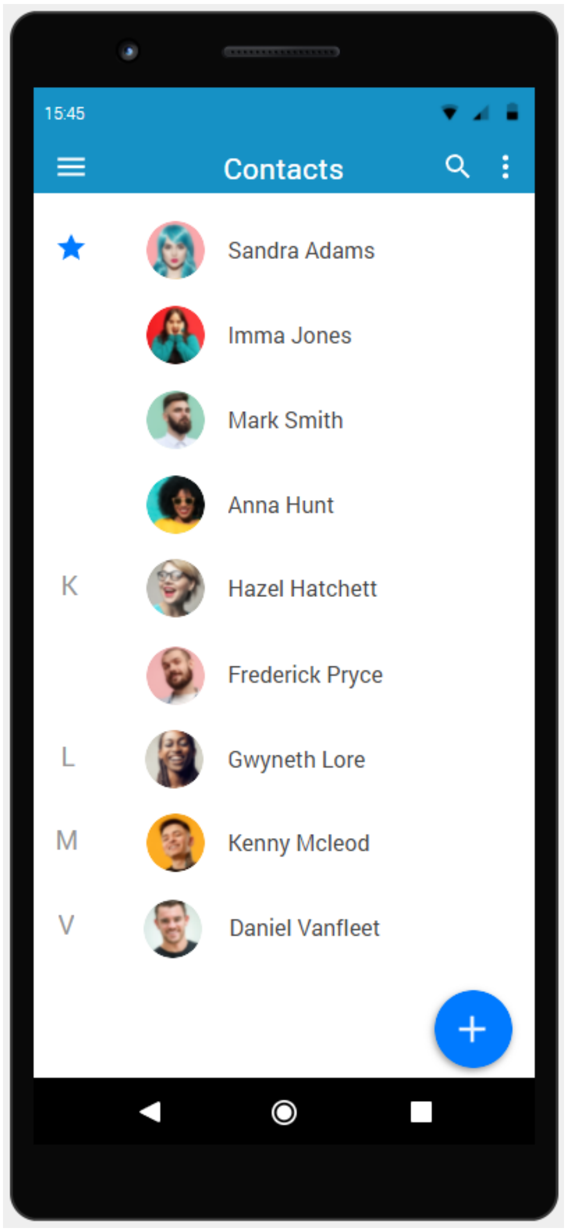
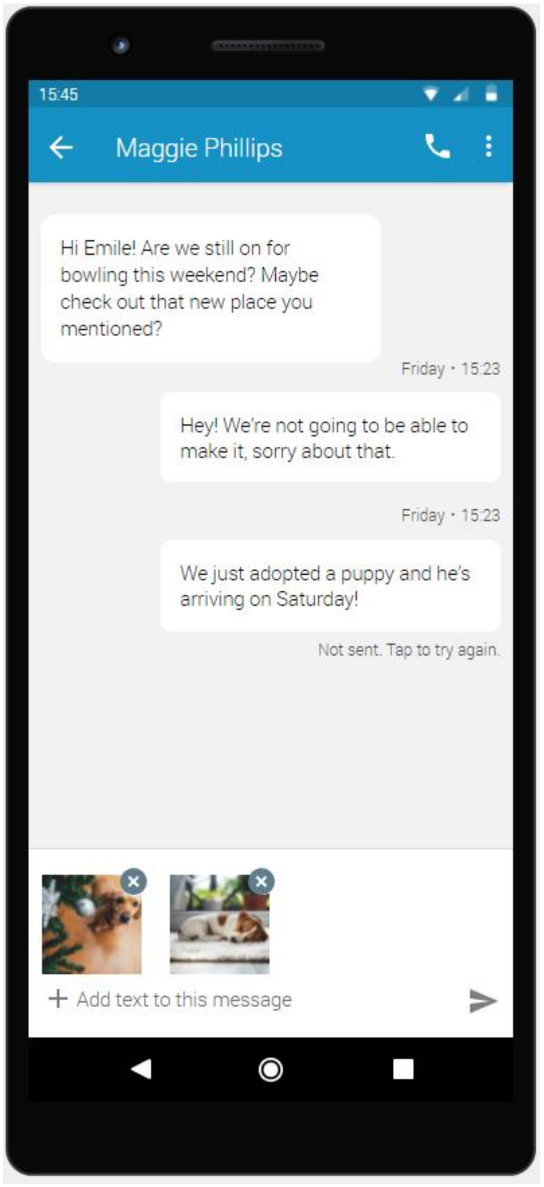


Figure 15 Buddies Contacts List Prototype Design

Figure 15:

This is the mobile application's contact list screen. Here users can see a list of their buddies they have connected with on the mobile application. If the user clicks on a buddy from the list it would link them to the chat for that particular buddy.



Hi Emile! Are we still on for bowling this weekend? Maybe check out that new place you mentioned?

Friday • 15:23

Hey! We're not going to be able to make it, sorry about that.

Friday • 15:23

We just adopted a puppy and he's arriving on Saturday!

Not sent. Tap to try again.

Figure 16 Chat Page Prototype Design

Figure 16:

This is the mobile application's chat screen. This allows the user to communicate with a particular buddy and attach photos, files and arrange appointments etc. The user can also call the buddy if he/she wishes to. If the user clicks the name of the buddy it would link them to their profiles.

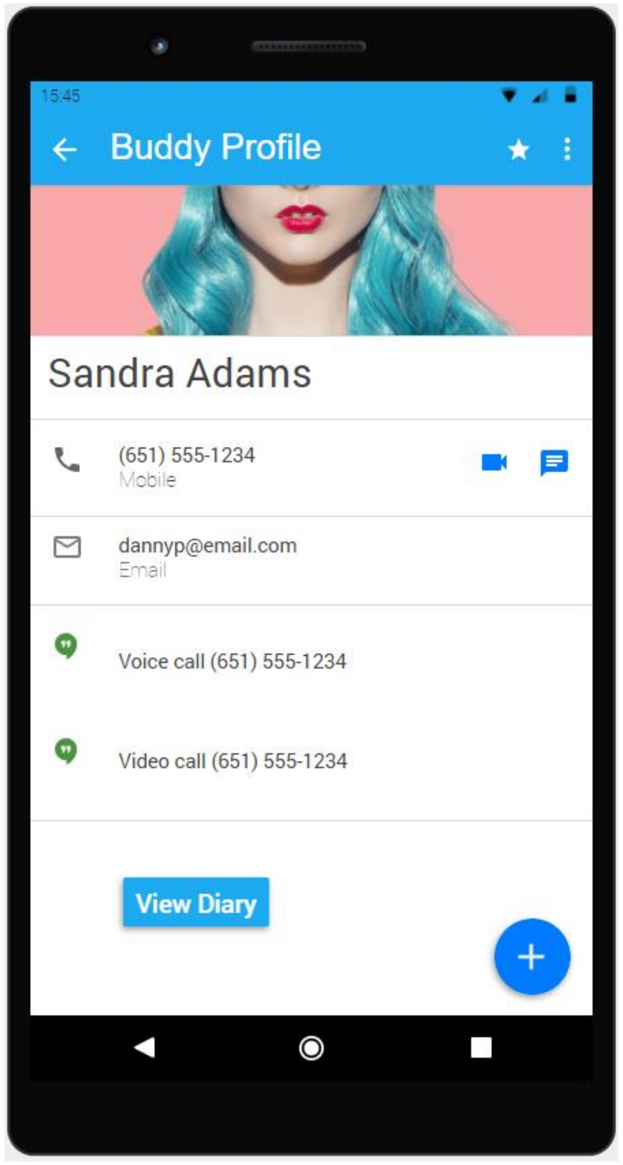


Figure 17 Buddy Profile Page Prototype Design

Figure 17:

This is the mobile application's buddy profile screen. Here users can view other buddy profiles and view a diary file of other buddy's personal diary and information.

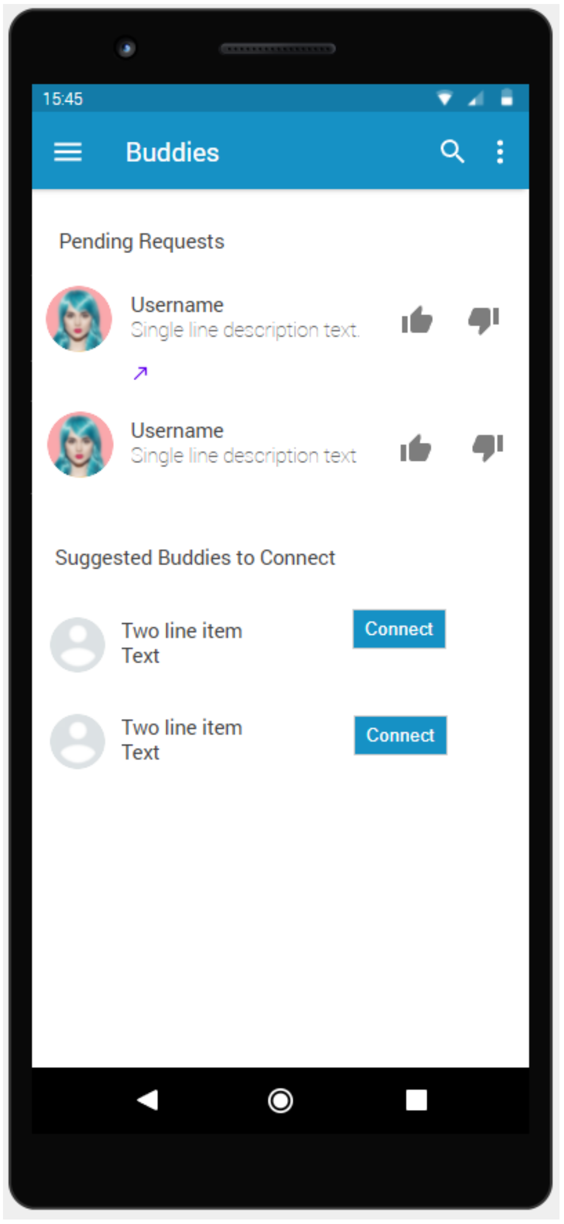


Figure 18 Buddies Suggestions List and Requests Page Prototype Design

Figure 18:

This is the mobile application's pending requests and suggested buddy's screen. Here the users can view a list of pending requests and accept or decline the request to connect with them. It also suggests buddies to connect to by closest distance to you.

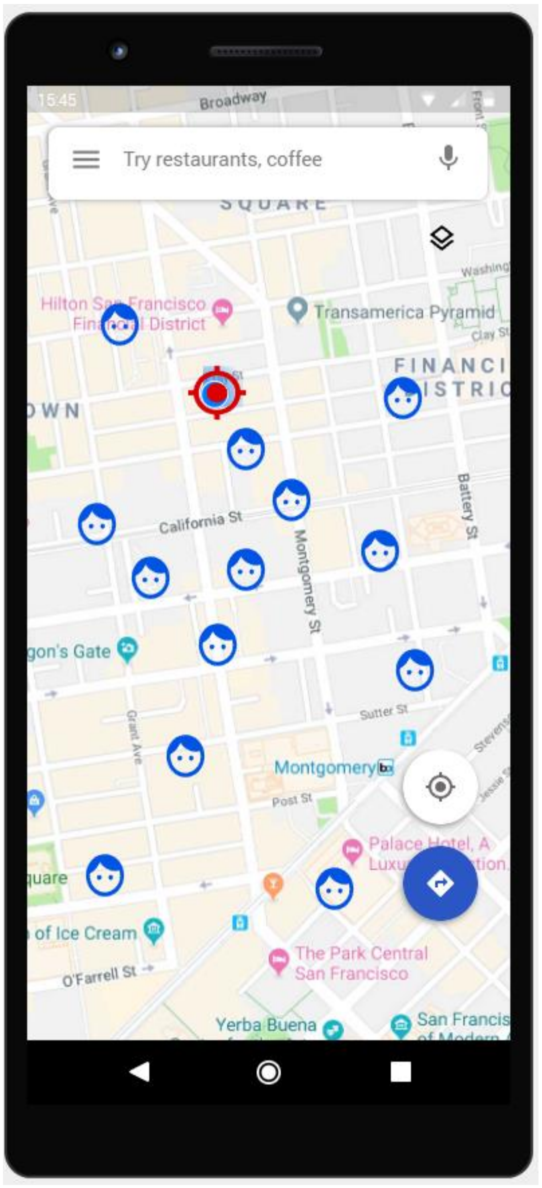
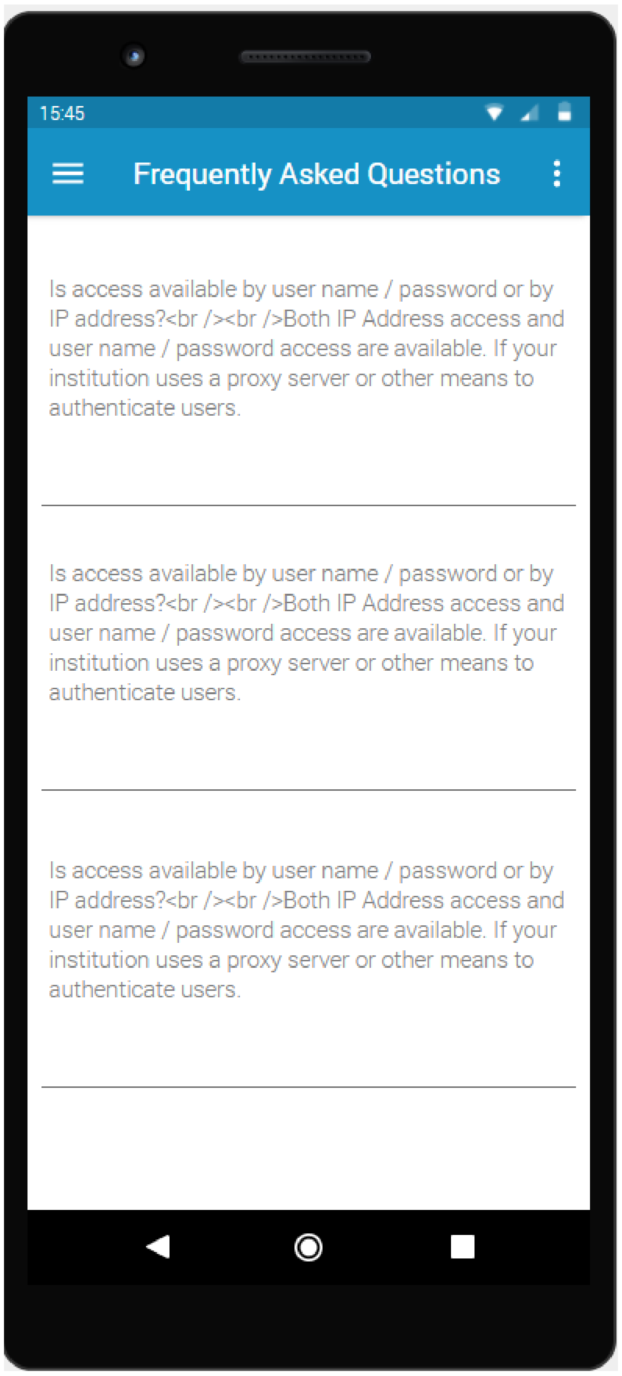


Figure 19 Google Maps Nearest Buddies Page Prototype Design

Figure 19:

This is the mobile application's google maps nearest buddy's screen. Here users can view a map and see other buddies around the world using the application and connect with the buddies closest to their location.



Is access available by user name / password or by IP address?<br /><br />Both IP Address access and user name / password access are available. If your institution uses a proxy server or other means to authenticate users.

Is access available by user name / password or by IP address?<br /><br />Both IP Address access and user name / password access are available. If your institution uses a proxy server or other means to authenticate users.

Is access available by user name / password or by IP address?<br /><br />Both IP Address access and user name / password access are available. If your institution uses a proxy server or other means to authenticate users.

Figure 20 Frequently Asked Questions Page Prototype Design

Figure 20:

This is the mobile application’s frequently asked questions screen. Here the users can view a list of frequently asked questions from multiple users which could be found here.

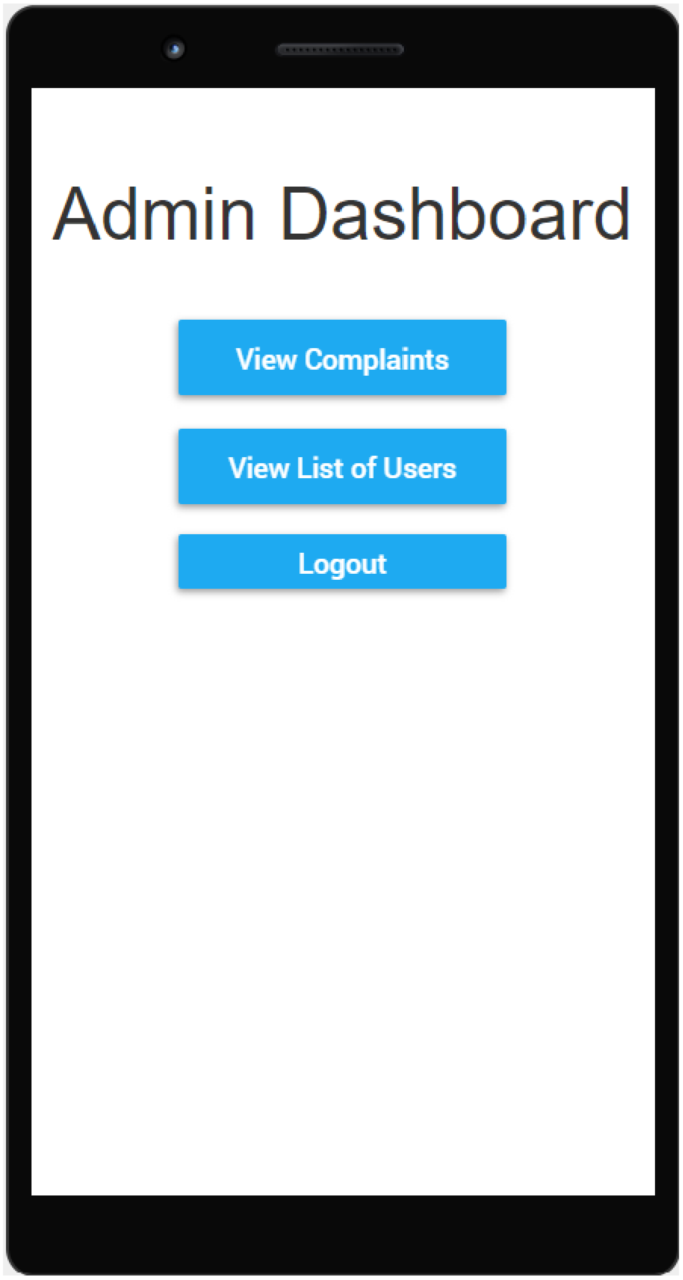


Figure 21 Admin Dashboard Page Prototype Design

Figure 21:

This is the mobile application's admin dashboard screen. Here the admin can either view buddies' complaints or view a list of users by clicking on the buttons.

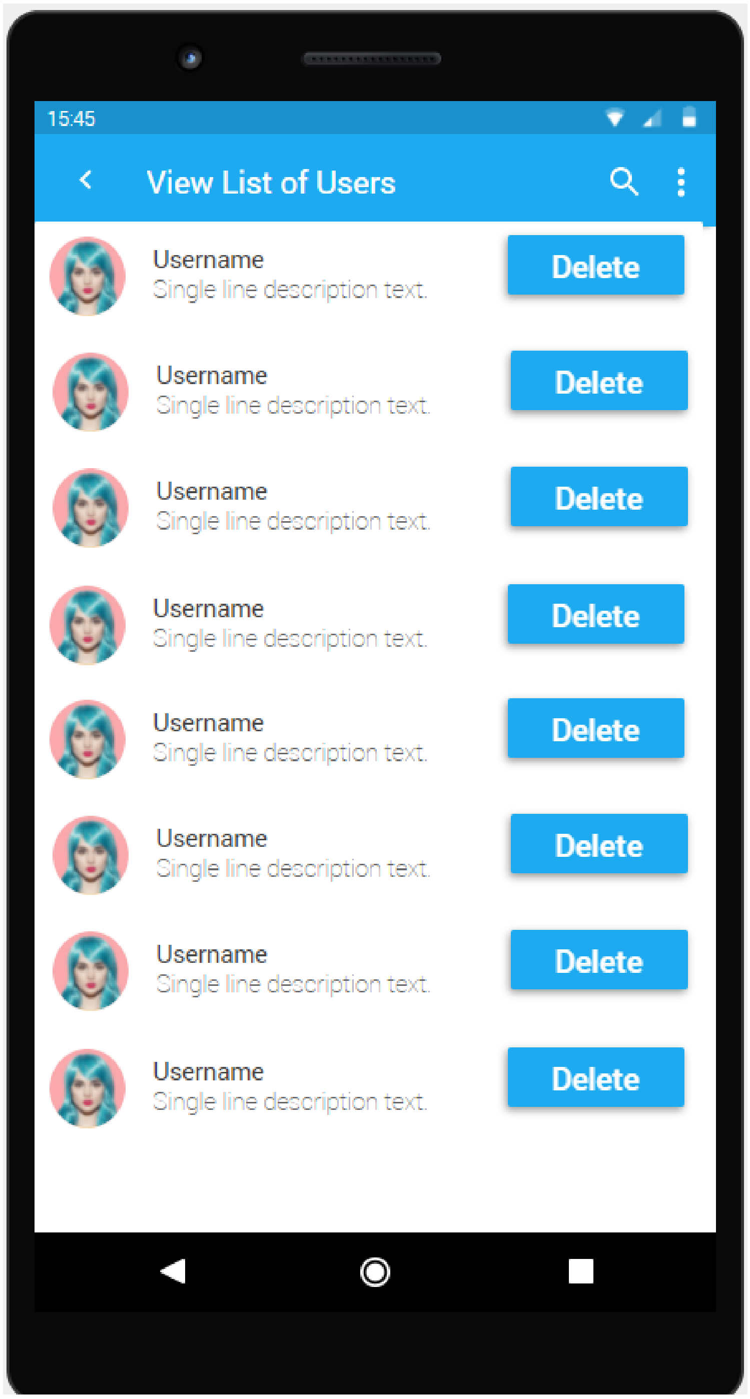

15:45
View List of Users
Username
Single line description text.
Delete
Username
Single line description text.
Delete
Username
Single line description text.
Delete
Username
Single line description text.
Delete
Username
Single line description text.
Delete
Username
Single line description text.
Delete
Username
Single line description text.
Delete
Username
Single line description text.
Delete

Figure 22 List of Users Page Prototype Design

Figure 22:

This is the mobile application's list of user's screens. Here the admin views a list of users and can delete a user.

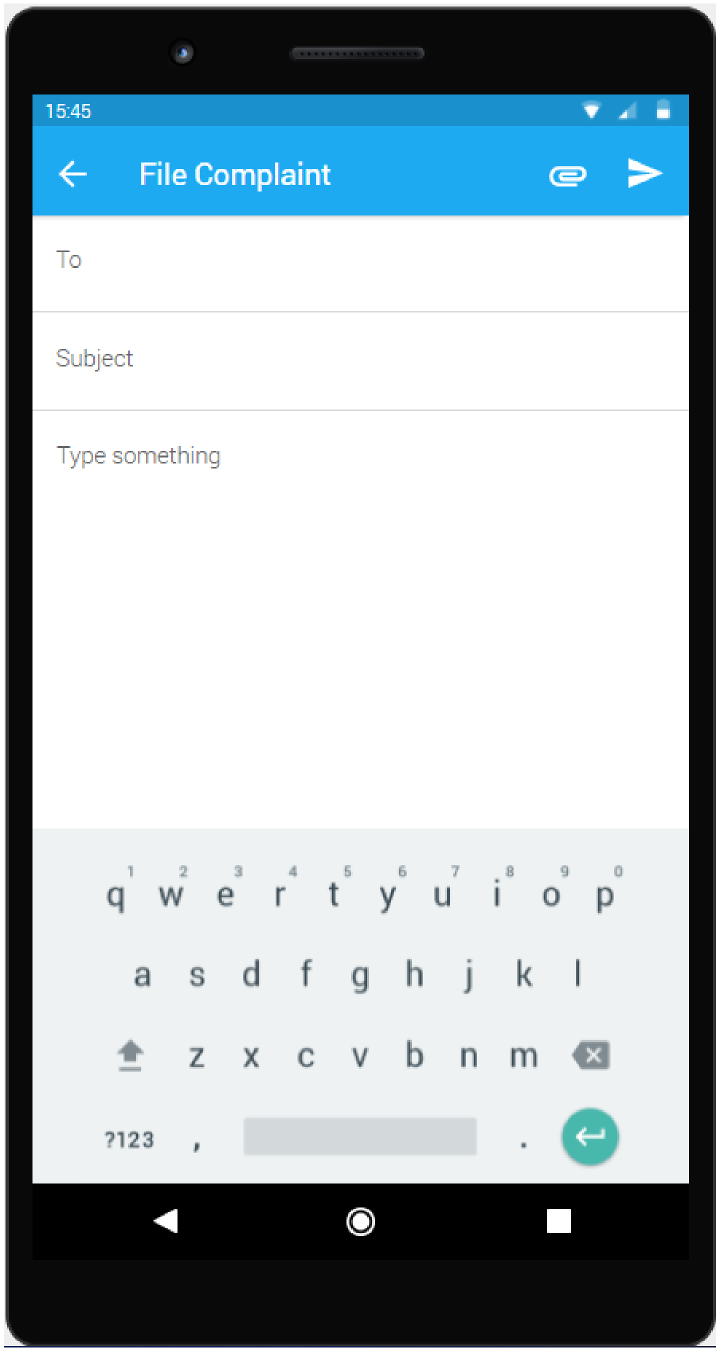


Figure 23 File Complaint Page Prototype Design

Figure 23:

This is the mobile application’s file complaint screens. Here the user can file a complaint and report abuse on the application.

4.5 Use Case Diagram

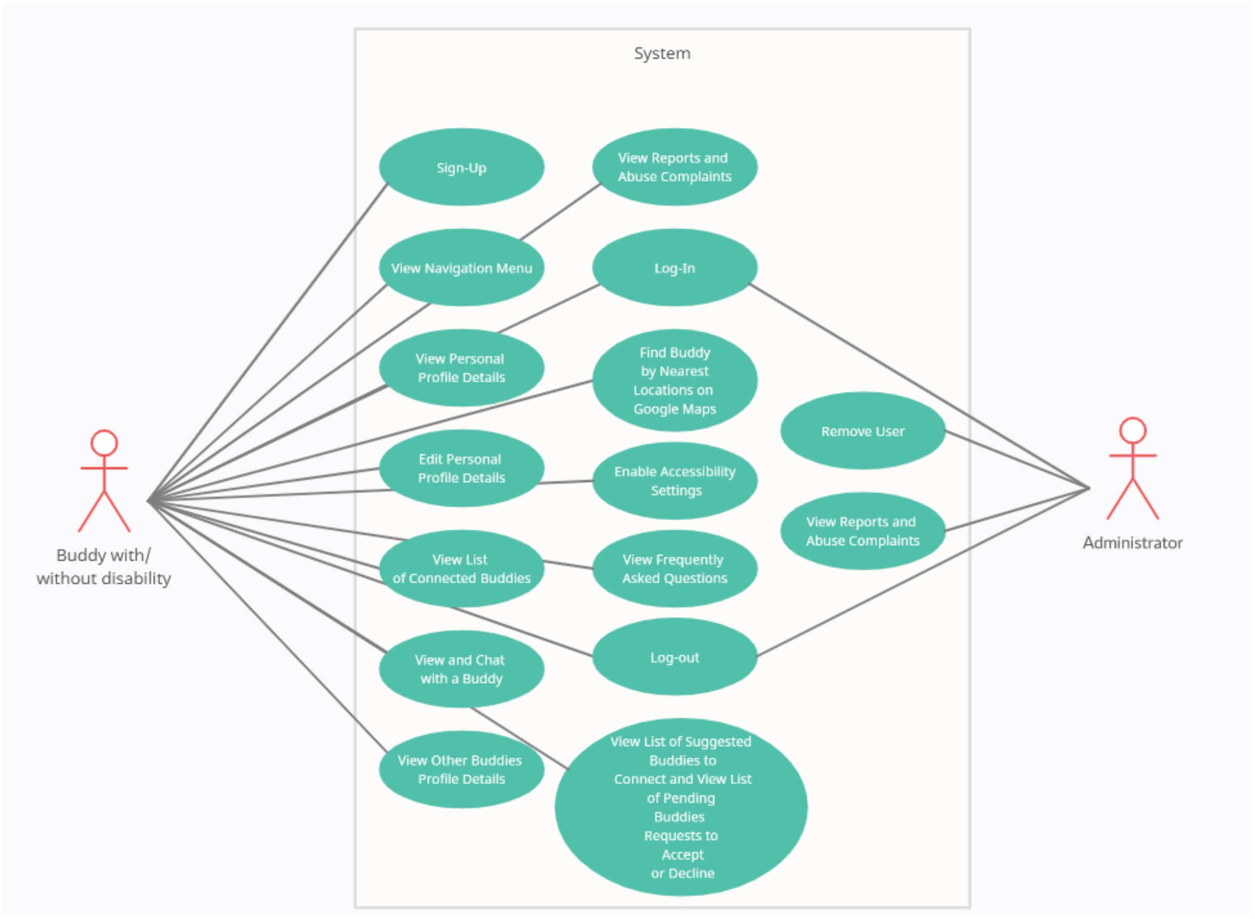

System
Sign-Up
View Reports and Abuse Complaints
View Navigation Menu
Log-In
View Personal Profile Details
Find Buddy by Nearest Locations on Google Maps
Remove User
Edit Personal Profile Details
Enable Accessibility Settings
View Reports and Abuse Complaints
View List of Connected Buddies
View Frequently Asked Questions
View and Chat with a Buddy
Log-out
View Other Buddies Profile Details
View List of Suggested Buddies to Connect and View List of Pending Buddies Requests to Accept or Decline
Buddy with/ without disability
Administrator

Figure 24 Use Case Diagram

Here the Author has designed a Use Case Diagram to show an overview of the use cases and system that would be defined within the next chapter. In the use case diagram, we have the system and use case operations within the mobile application system. The participating actors within this system are the Buddy with/without disability and the Administrator, which link to various functions/use cases within the system, as shown in the figure above.

4.6 Use Cases

Use Case Name:	Sign Up
Participating Actor:	Buddy with & without disability
Entry Conditions:	1.The Buddy has accessed the system to create an Account. 2.The Welcome Screen function of the system has been invoked.
Flow of Events:	1. The system responds by displaying the welcome screen which displays a welcome message and 1.1 FOR each new buddy the system requests the buddy to sign up and enter a name, email, phone, password, date of birth, gender, and their role. 1.2. The buddy then enters their details into the system to sign up. 2. The system responds by checking if the email and other credentials are valid. If the credentials are valid the system adds the details to the database and the account will be created.
Alternative Flow of Events:	2.1 The system responds by displaying an error message on the sign-up page and prompts the buddy to enter valid details.
Exit Conditions:	The Buddy has successfully created an account.

Table 2 Sign-Up Use Case

Use Case Name:	Log-In
Participating Actor:	Buddy with & without disability, Admin
Entry Conditions:	1.The Buddy/Admin has accessed the system to login to their Account.
Flow of Events:	1. The system responds by displaying the login screen which displays two fields prompting the buddy/admin to login with their credentials. 1.1 FOR each returning buddy/admin the system requests the buddy/admin to enter their email and password to login to the system. 2. Buddy/Admin enters their email and password. 3. The system responds by checking if the information provided in the fields are valid and if the information is correct the system logs in the buddy/admin and brings them to their contacts buddy list page for buddies or dashboard page for the admin.
Alternative Flow of Events:	3.1 The system responds by displaying an error message saying that the credentials are invalid and prompts the buddy/admin to login again.
Exit Conditions:	The Buddy/Admin has successfully logged in to their account.

Table 3 Log-In Use Case

Use Case Name:	View Navigation Menu
Participating Actor:	Buddy with & without disability
Entry Conditions:	1.The Buddy has successfully logged into their account. 2. The View List of Connected Buddies function has been invoked.
Flow of Events:	1. The system responds by displaying the View List of Connected Buddies Screen and allows the buddy to click on the nav icon to display the navigation menu pane. 2. Buddy clicks on the nav menu icon. 3. The system responds by displaying the navigation menu with several links to different pages on the system.
Alternative Flow of Events:	2.1 The buddy backs out of the navigation bar by closing it by pressing the x icon in the menu.
Exit Conditions:	The buddy has successfully viewed the Navigation menu of the system.

Table 4 View Navigation Menu Use Case

Use Case Name:	View Personal Profile Details
Participating Actor:	Buddy with & without disability
Entry Conditions:	1. The buddy is already a member and logged into the system. 2. The View Navigation Menu function of the system has been invoked.
Flow of Events:	1. The system displays the navigation menu pane. 2. The buddy selects the My Profile link in the navigation menu to view the buddy's personal profile. 3. The system then displays a screen which shows all the details of the buddy's personal profile. 4. The buddies can then view their details of their profiles.
Alternative Flow of Events:	2.1 The buddy backs out of the navigation menu pane to close the navigation bar.
Exit Conditions:	The buddy has successfully viewed their personal profile details.

Table 5 View Personal Profile Details Use Case

Use Case Name:	Edit Personal Profile Details
Participating Actor:	Buddy with & without disability
Entry Conditions:	1. The buddy is already a member and logged into the system. 2. The View Personal Profile Details function of the system has been invoked.
Flow of Events:	1. The system then displays a screen showing the buddy personal profile details. 2. The buddy clicks on a specific detail on their profile. 3. The system prompts the buddy to update that specific detail information. 4. The buddies updates the information and clicks save. 5. The system then updates the information and displays the information in the profile screen.
Alternative Flow of Events:	
Exit Conditions:	The buddy has successfully edited their profile details.

Table 6 Edit Personal Profile Details Use Case

Use Case Name:	View List of Connected Buddies
Participating Actor:	Buddy with & without disability
Entry Conditions:	1. The buddy is already a member and logged into the system. 2. The View Navigation Menu function of the system has been invoked.
Flow of Events:	1. The system displays the navigation menu pane. 2. The buddy selects the View List of Connected Buddies link in the navigation menu to view a list of the Buddies Connections. 3. The system then displays a screen which shows a list of connected buddies that the buddy has connected to.
Alternative Flow of Events:	2.1 The buddy backs out of the navigation menu pane to close the navigation bar.
Exit Conditions:	The buddy has successfully view the list of their connected buddies.

Table 7 View List of Connected Buddies Use Case

Use Case Name:	View and Chat with a Buddy
Participating Actor:	Buddy with & without disability
Entry Conditions:	1. The buddy is already a member and logged into the system. 2. The View List of Connected Buddies function of the system has been invoked.
Flow of Events:	1. The system displays a list of connected buddies. 2. The buddy selects a specific buddy to view and chat with them. 3. The system then displays a screen showing the specific buddy and the chat which allows buddies to send messages and make calls to that specific buddy. 4. The buddy then views and send messages to that specific buddy.
Alternative Flow of Events:	
Exit Conditions:	The buddy has successfully viewed and send messages to their buddies.

Table 8 View and Chat with a Buddy Use Case

Use Case Name:	View Other Buddies Profile Details
Participating Actor:	Buddy with & without disability
Entry Conditions:	1. The buddy is already a member and logged into the system. 2. The View and Chat with a Buddy function of the system has been invoked.
Flow of Events:	1. The system then displays a screen showing the specific buddy and the chat which allows buddies to send messages and make calls to that specific buddy. 2. The buddy then clicks the specific buddy's name at the top of the page. 3. The system then displays the profile details of that specific buddy along with their diary.
Alternative Flow of Events:	
Exit Conditions:	The buddy has successfully viewed other buddies profile details.

Table 9 View Other Buddies Profile Details Use Case

Use Case Name:	View List of Suggested Buddies to Connect and View List of Pending Buddies Requests to Accept or Decline
Participating Actor:	Buddy with & without disability
Entry Conditions:	1. The buddy is already a member and logged into the system. 2. The View Navigation Menu function of the system has been invoked.
Flow of Events:	1. The system displays the navigation menu pane. 2. The buddy selects the Buddies Requests and Suggestions link in the navigation menu to view a list of recommended buddies and pending requests for the buddy. 3. The system then displays a screen which shows a list of recommended buddies and pending requests for the buddy. 4. The buddies can then view and connected to the suggested buddies on the list and also accept or decline pending requests from other buddies to connect. 5. After accepting or declining a request the systems will then notify the other buddy on whether the buddy had accepted or declined the request.
Alternative Flow of Events:	2.1 The buddy backs out of the navigation menu pane to close the navigation bar.
Exit Conditions:	The buddy has successfully viewed a list of suggested buddies to connect and viewed a list of pending requests from other buddies to accept or decline their requests to connect.

Table 10 View List of Suggested Buddies to Connect and View List of Pending Buddies Requests to Accept or Decline Use Case

Use Case Name:	Find Buddy by Nearest Locations on Google Maps
Participating Actor:	Buddy with & without disability
Entry Conditions:	1. The buddy is already a member and logged into the system. 2. The View Navigation Menu function of the system has been invoked.
Flow of Events:	1. The system displays the navigation menu pane. 2. The buddy selects the Find a Buddy link in the navigation menu to view FAQ's regarding the system. 3. The system then displays a screen which shows a map of buddy using the system and see the closest buddies based on nearest location. 4. The buddies can then select the buddies closest to them and connect with them from the map view.
Alternative Flow of Events:	2.1 The buddy backs out of the navigation menu pane to close the navigation bar.
Exit Conditions:	The buddy has successfully found and connected to another buddy from the map view by closest location.

Table 11 Find Buddy by Nearest Locations on Google Maps Use Case

Use Case Name:	Enable Accessibility Settings
Participating Actor:	Buddy with & without disability
Entry Conditions:	1. The buddy is already a member and logged into the system. 2. The View Navigation Menu function of the system has been invoked.
Flow of Events:	1. The system displays the navigation menu pane. 2. The buddy selects the Enable Accessibility Settings link in the navigation menu to enable accessibility features for people with various impairments . 3. The system navigates the buddy to the phones device settings and allows them to enable accessibility features to help them navigate on the system better. 4. The buddy selects the accessibility features that they wish to enable for the system and click on the back button once selecting the features to enabled. 4. The system the returns the buddy back to the screen that they previously were using.
Alternative Flow of Events:	2.1 The buddy backs out of the navigation menu pane to close the navigation bar.
Exit Conditions:	The buddy has successfully enabled Accessibility Features.

Table 12 Enable Accessibility Settings Use Case

Use Case Name:	View Frequently Asked Questions
Participating Actor:	Buddy with & without disability
Entry Conditions:	1. The buddy is already a member and logged into the system. 2. The View Navigation Menu function of the system has been invoked.
Flow of Events:	1. The system displays the navigation menu pane. 2. The buddy selects the View Frequently Asked Questions link in the navigation menu. to view FAQ’s regarding the system. 3. The system then displays a screen which shows the most Frequently Asked Questions regarding the Application.
Alternative Flow of Events:	2.1 The buddy backs out of the navigation menu pane to close the navigation bar.
Exit Conditions:	The buddy has successfully viewed the Frequently Asked Questions Screen of the system.

Table 13 View Frequently Asked Questions Use Case

Use Case Name:	Log-out
Participating Actor:	Buddy with & without disability, Admin
Entry Conditions:	1. The buddy/admin is already a member and logged into the system. 2. The View Navigation Menu/Admin Dashboard function of the system has been invoked.
Flow of Events:	1. The system displays the navigation menu pane/admin dashboard. 2. The buddy/admin selects the logout link in the navigation menu/admin dashboard to sign out of the system. 3. The system logs the buddy/admin out of the systems and displays the welcome screen.
Alternative Flow of Events:	2.1 The buddy/admin backs out of the navigation menu pane/admin dashboard button to close the navigation bar/logout button.
Exit Conditions:	The buddy/admin has successfully logged out of the system.

Table 14 Log-out Use Case

Use Case Name:	View Reports and Abuse Complaints
Participating Actor:	Admin
Entry Conditions:	1. The admin is already a member and logged into the system. 2. The View Admin Dashboard view of the system has been invoked.
Flow of Events:	1. The system displays the admin dashboard page and prompts the user to either view a list of users or view complaints. 2. The admin selects the view complaints button. 3. The system displays a list of complaints on the mobile application. 4. The admin views the list of complaints.
Alternative Flow of Events:	N/A
Exit Conditions:	The admin has successfully viewed a list of complaints written by users.

Table 15 View Reports and Abuse Complaints Use Case

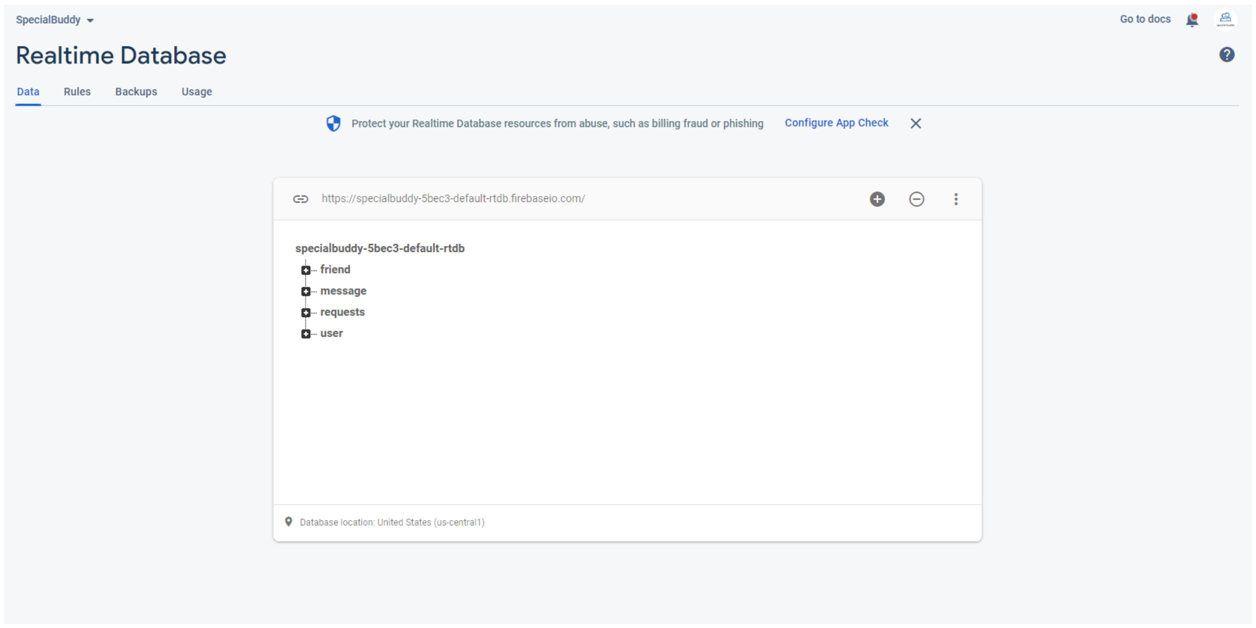
Use Case Name:	View and Remove User
Participating Actor:	Admin
Entry Conditions:	1. The admin is already a member and logged into the system. 2. The View Admin Dashboard view of the system has been invoked.
Flow of Events:	1. The system displays the admin dashboard page and prompts the user to either view a list of users or view complaints. 2. The admin selects the view a list of user's button. 3. The system displays a list of users on the mobile application. 4. The admin views the list of users and clicks the delete button for one of the users. 5. The system then deletes the user from the mobile application.
Alternative Flow of Events:	N/A
Exit Conditions:	The admin has successfully viewed and deleted users from the mobile application.

Table 16 Remove User Use Case

Use Case Name:	File Complaints and Report Abuse
Participating Actor:	Buddy with & without disability
Entry Conditions:	1. The buddy is already a member and logged into the system. 2. The View Navigation Menu function of the system has been invoked.
Flow of Events:	1. The system displays the navigation menu pane. 2. The buddy selects the file complaints link in the navigation menu to file a complaint on the mobile application. 3. The system displays a for asking the user to fill details about the complaint. 4. The user fills out the details and clicks the send button. 5. The system responds by displaying a message saying that the issue would be dealt with as soon as possible.
Alternative Flow of Events:	2.1 The buddy backs out of the navigation menu pane to close the navigation bar.
Exit Conditions:	The buddy has successfully filed a complaint on the system.

Table 17 File Complaints and Report Abuse Use Case

4.7 Database Design


SpecialBuddy
Go to docs
Realtime Database
Data
Rules
Backups
Usage
Protect your Realtime Database resources from abuse, such as billing fraud or phishing
Configure App Check
https://specialbuddy-5bec3-default-rtdb.firebaseio.com/

specialbuddy-5bec3-default-rtdb
- friend
- message
- requests
- user

Database location: United States (us-central1)

Figure 25 Database Diagram

The database is hosted using Firebase. The Firebase Realtime Database allows the user to easily control log-in and sign-up using Firebase Authentication, and holding the user, messaging, request, and friends' data using Firebase's real-time database. The user data table holds all the data in relation to the user using the application. It stores the all the information of the buddy such as the name, email, phone, diary, image, location, and their role within the application. The friend's data table holds all of the connected buddies that a particular user has on the mobile application. The message table holds all the data from messages sent to different buddies from different buddy accounts. The requests table then holds all the data on various requests sent from different buddies using the application. The Author also intends on using the Analytics provided from Google Firebase to identify patterns within the data to improve and build the application further.

4.8 Logo

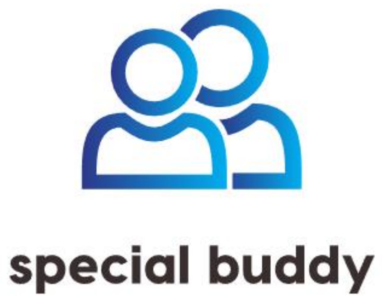


Figure 26 Application Logo

As shown in Figure 11, the Author has created and designed this logo icon for this mobile application. The two-person image in the icon shows a relationship between two different people (people with and without disabilities) as one, showing a close bond and friendship. The Author has also decided to use a gradient of light and dark blue to represent freedom, intuition, imagination, and inspiration. The reason why the Author had chosen to use the colour blue was mainly due to the fact that blue is a colour that is appealing to all audiences and is visually engaging to both people with and without disabilities.

5 System Implementation

5.1 Tools and Technologies

To complete this project, various software technologies were used, such as Programming Languages and IDE's. this section gives an overview of the tools that were used to develop this mobile android application.

5.1.1 Programming Languages

In the development of this Project, the programming language that was used to develop this android mobile application was the following:

5.1.1.1 Java

Java was the programming language used to develop this mobile application mainly due to the fact there were many resources and materials to build mobile applications in Java that the Author could reference. Implementing this project using the Java programming language was easier as it has been the default language for android development for years. The Author being proficient in coding in Java also made the development of this mobile application easier to develop.

5.1.2 Integrated Development Environments (IDE's)

5.1.2.1 Android Studio

Android Studio is an IDE based on the IntelliJ Interface, which is used for Android application development. The reason for the Author choosing this IDE to develop this mobile android application is that it provides comprehensive tools and features such as pre-installed packages and Android Emulators, which allows the Author to test the mobile application on a physical Android phone, which enables the development of mobile applications to be more accessible and effortless to develop (Android Studio features | Android Developers, 2021).

5.2 Design Quality and Decisions

5.2.1 Android Application

The design from the design and analysis chapter stayed comparatively the same. Although there were some small changes in the design such as using different icons and some color changes for multiple design components. Using new technologies that the Author had not previously used such as Google Maps API, and Firebase, had allowed the Author to both gain and expand their knowledge in the use and development of Android Application Development. Below the Author has attached images of the final design implementation for this Project.

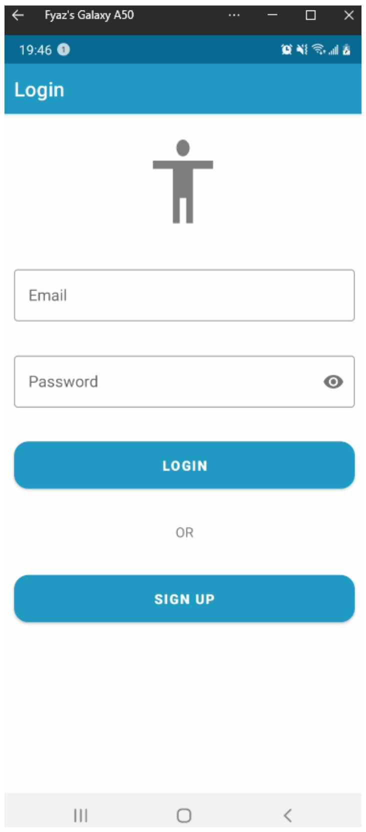


Figure 27: Login

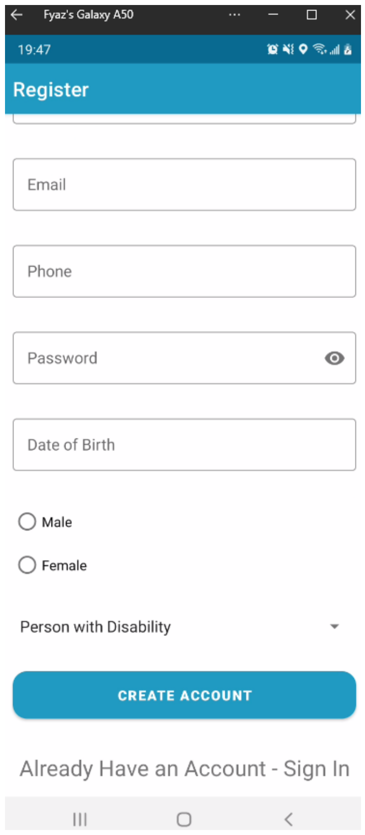


Figure 28: Register

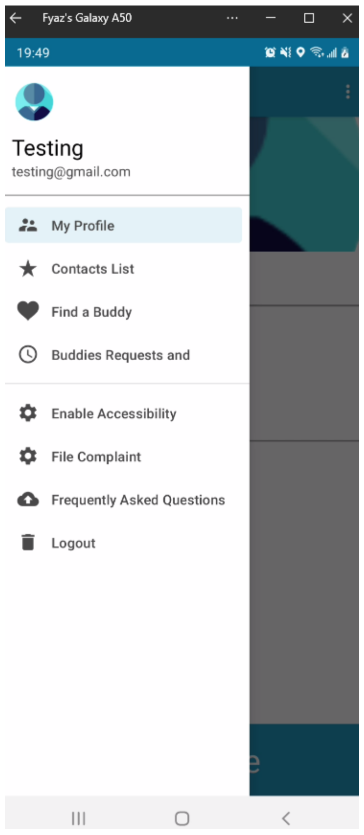


Figure 29: Nav Menu

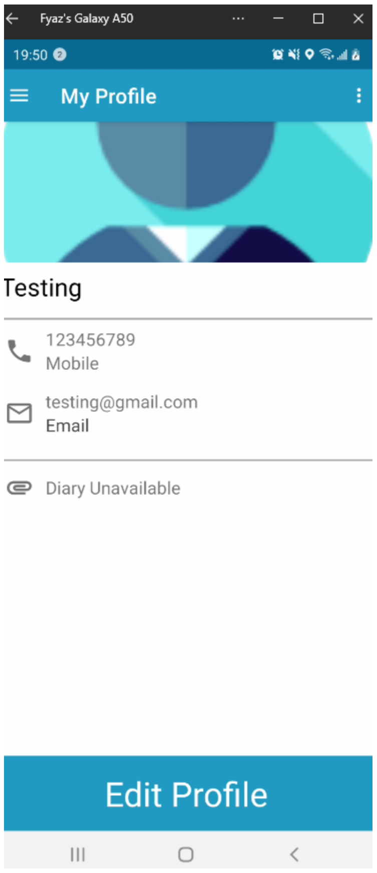


Figure 30: View Personal Details

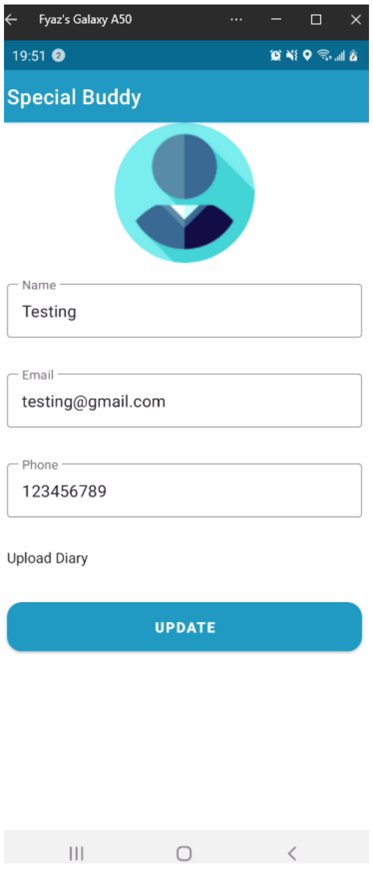


Figure 31: Edit Personal Details

Fyaz's Galaxy A50
19:56
Contacts List
man
19:55
I'm good wbu?

Figure 32: Contact List with Search

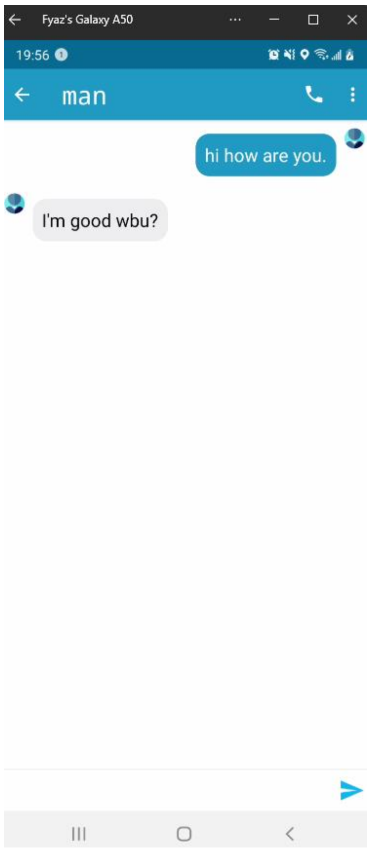


Figure 33: Chat Between Users

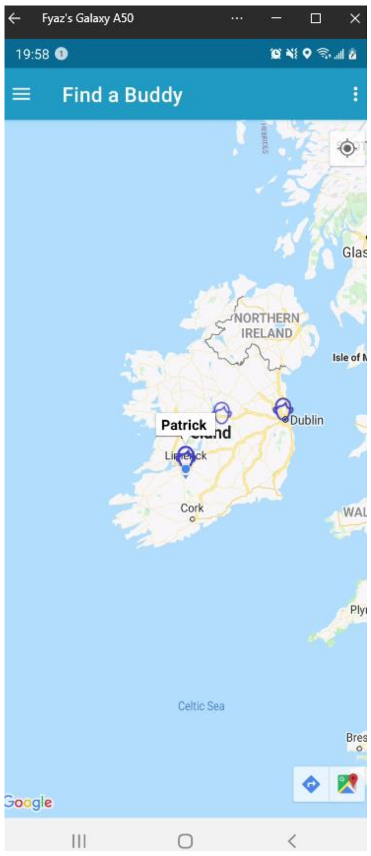
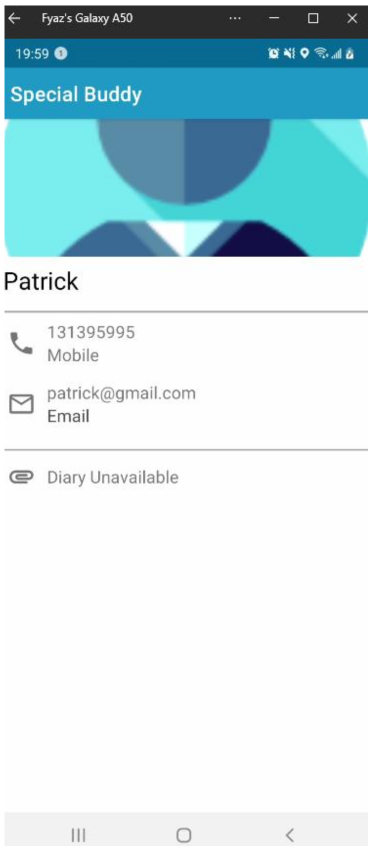


Figure 34: Find Users via Google Maps



Patrick
131395995
Mobile
patrick@gmail.com
Email
Diary Unavailable

Figure 35: View Other Buddies Profile

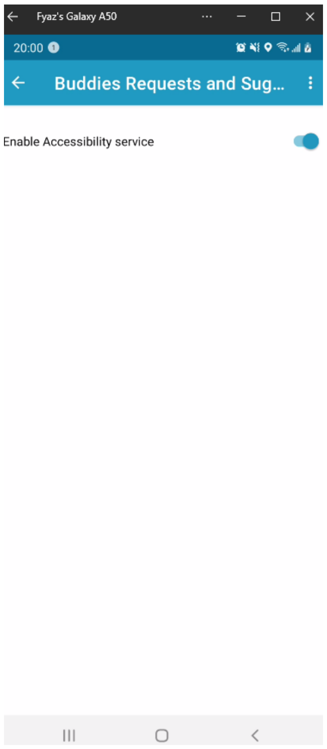
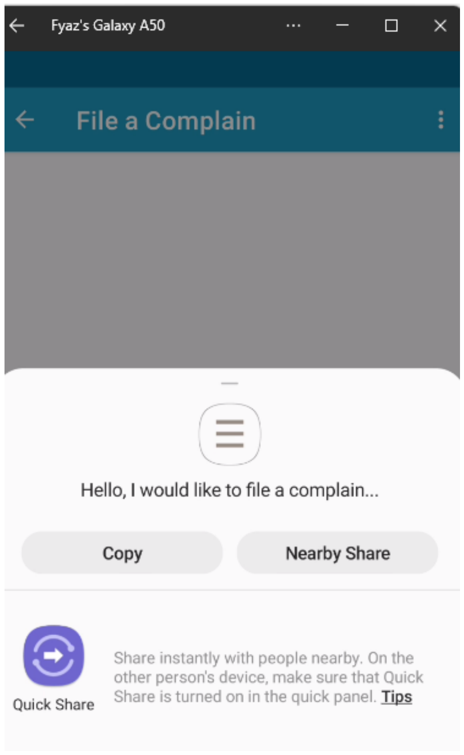


Figure 36: Enable Accessibility Service



Hello, I would like to file a complain...

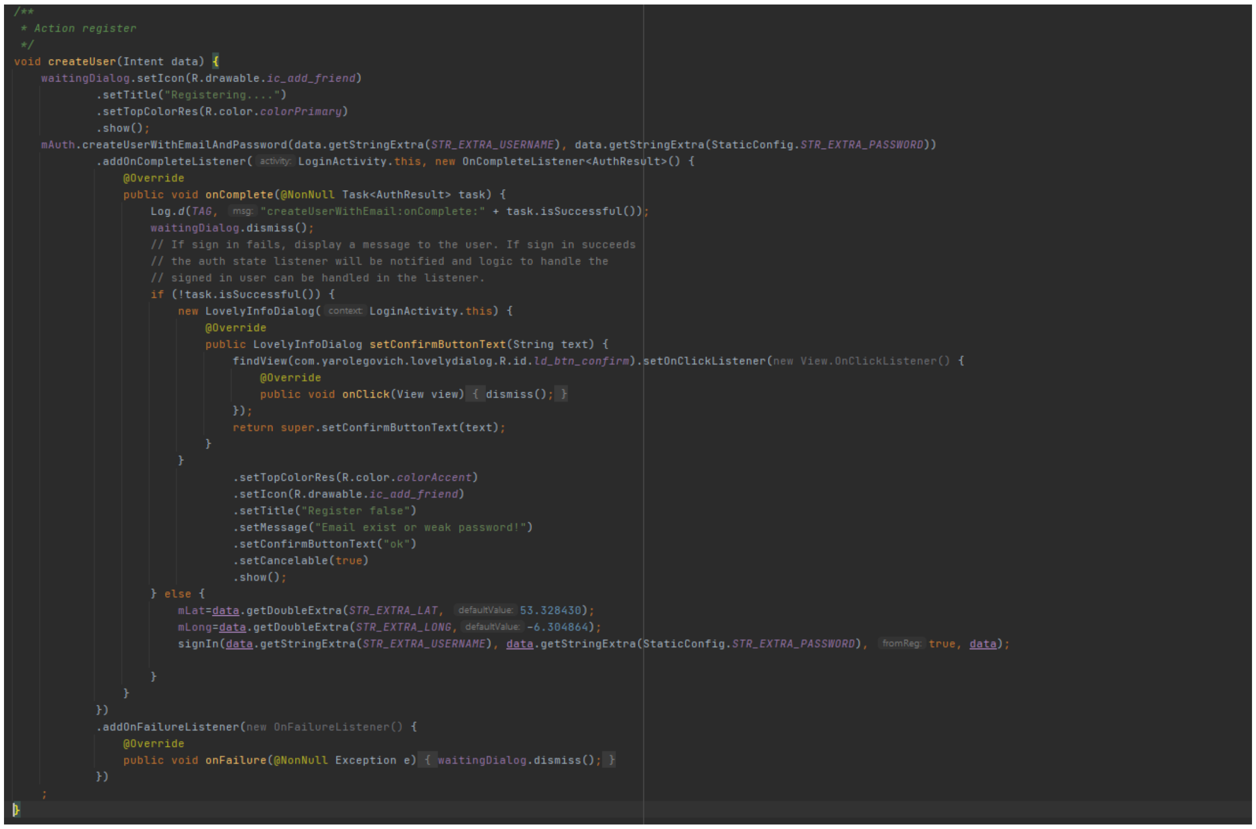
Share instantly with people nearby. On the other person's device, make sure that Quick Share is turned on in the quick panel. Tips

Figure 37: File a Complaint

5.3 Project Management

5.3.1 Authentication Set-Up

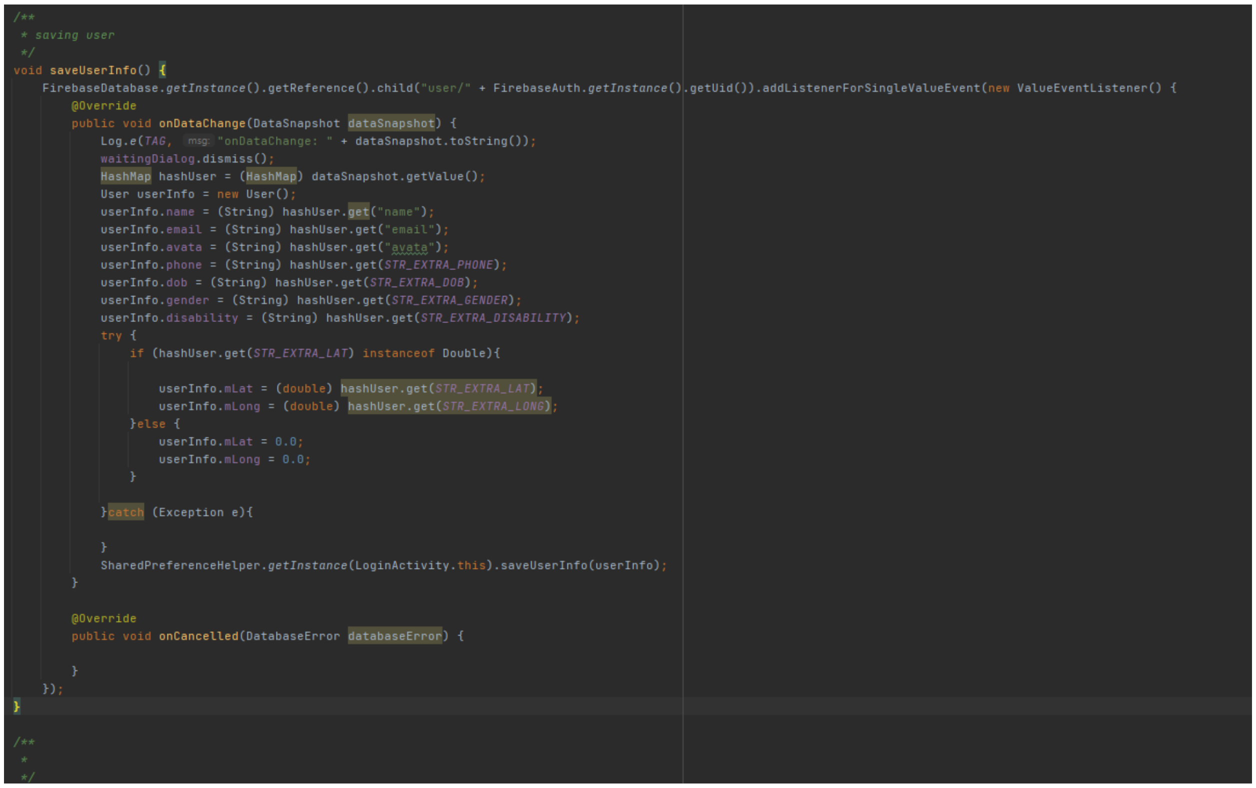
Authentication set up involved three steps which was the Login and Register Account for users on the special buddy mobile application and also setting up Authentication via Firebase.



```
/**
 * Action register
 */
void createUser(Intent data) {
    waitingDialog.setIcon(R.drawable.ic_add_friend)
            .setTitle("Registering....")
            .setTopColorRes(R.color.colorPrimary)
            .show();
    mAuth.createUserWithEmailAndPassword(data.getStringExtra(STR_EXTRA_USERNAME), data.getStringExtra(StaticConfig.STR_EXTRA_PASSWORD))
            .addOnCompleteListener( activity: LoginActivity.this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    Log.d(TAG,  msg: "createUserWithEmail:onComplete:" + task.isSuccessful());
                    waitingDialog.dismiss();
                    // If sign in fails, display a message to the user. If sign in succeeds
                    // the auth state listener will be notified and logic to handle the
                    // signed in user can be handled in the listener.
                    if (!task.isSuccessful()) {
                        new LovelyInfoDialog( context: LoginActivity.this) {
                            @Override
                            public LovelyInfoDialog setConfirmButtonText(String text) {
                                findView(com.yarolegovich.lovelydialog.R.id.ld_btn_confirm).setOnClickListener(new View.OnClickListener() {
                                    @Override
                                    public void onClick(View view) { dismiss(); }
                                });
                                return super.setConfirmButtonText(text);
                            }
                        }
                                .setTopColorRes(R.color.colorAccent)
                                .setIcon(R.drawable.ic_add_friend)
                                .setTitle("Register false")
                                .setMessage("Email exist or weak password!")
                                .setConfirmButtonText("ok")
                                .setCancelable(true)
                                .show();
                    } else {
                        mLat=data.getDoubleExtra(STR_EXTRA_LAT,  defaultValue: 53.328430);
                        mLong=data.getDoubleExtra(STR_EXTRA_LONG,  defaultValue: -6.304864);
                        signIn(data.getStringExtra(STR_EXTRA_USERNAME), data.getStringExtra(StaticConfig.STR_EXTRA_PASSWORD),  fromReg: true, data);

                    }
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) { waitingDialog.dismiss(); }
            })
    ;
}
```

Figure 38: Register Account

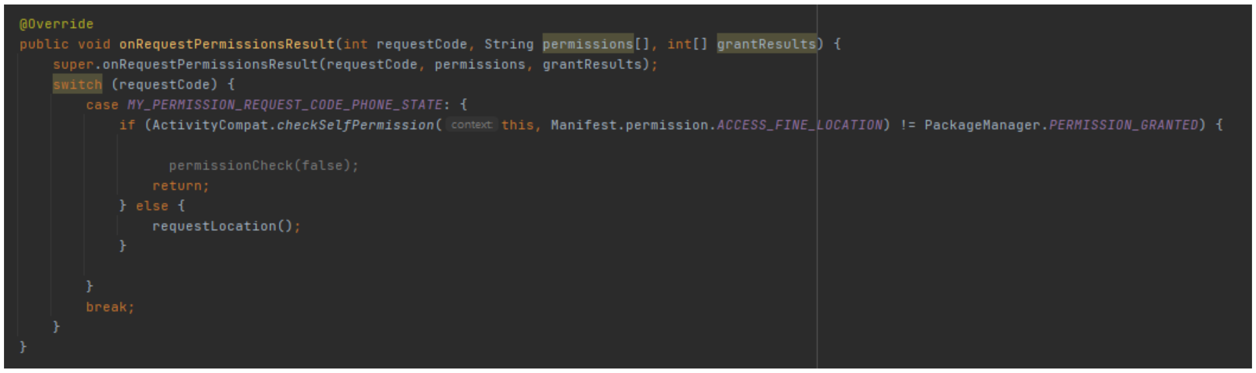
Firstly, the Author had coded a method to create a new user on the mobile application. As we can see from the figure above, If the details are all correctly entered, and the sign-up button is pressed dialog message would appear and successfully register the account and login the user automatically to their account. If an email already exists on the mobile application or if password is weak a dialog would appear and block the user from registering as a user.



```java
/**
 * saving user
 */
void saveUserInfo() {
    FirebaseDatabase.getInstance().getReference().child("user/" + FirebaseAuth.getInstance().getUid()).addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            Log.e(TAG,  msg: "onDataChange: " + dataSnapshot.toString());
            waitingDialog.dismiss();
            HashMap hashUser = (HashMap) dataSnapshot.getValue();
            User userInfo = new User();
            userInfo.name = (String) hashUser.get("name");
            userInfo.email = (String) hashUser.get("email");
            userInfo.avata = (String) hashUser.get("avata");
            userInfo.phone = (String) hashUser.get(STR_EXTRA_PHONE);
            userInfo.dob = (String) hashUser.get(STR_EXTRA_DOB);
            userInfo.gender = (String) hashUser.get(STR_EXTRA_GENDER);
            userInfo.disability = (String) hashUser.get(STR_EXTRA_DISABILITY);
            try {
                if (hashUser.get(STR_EXTRA_LAT) instanceof Double){

                    userInfo.mLat = (double) hashUser.get(STR_EXTRA_LAT);
                    userInfo.mLong = (double) hashUser.get(STR_EXTRA_LONG);
                }else {
                    userInfo.mLat = 0.0;
                    userInfo.mLong = 0.0;
                }

            }catch (Exception e){

            }
            SharedPreferenceHelper.getInstance(LoginActivity.this).saveUserInfo(userInfo);
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });
}

/**
 *
 */
```

Figure 39: Saving User Info

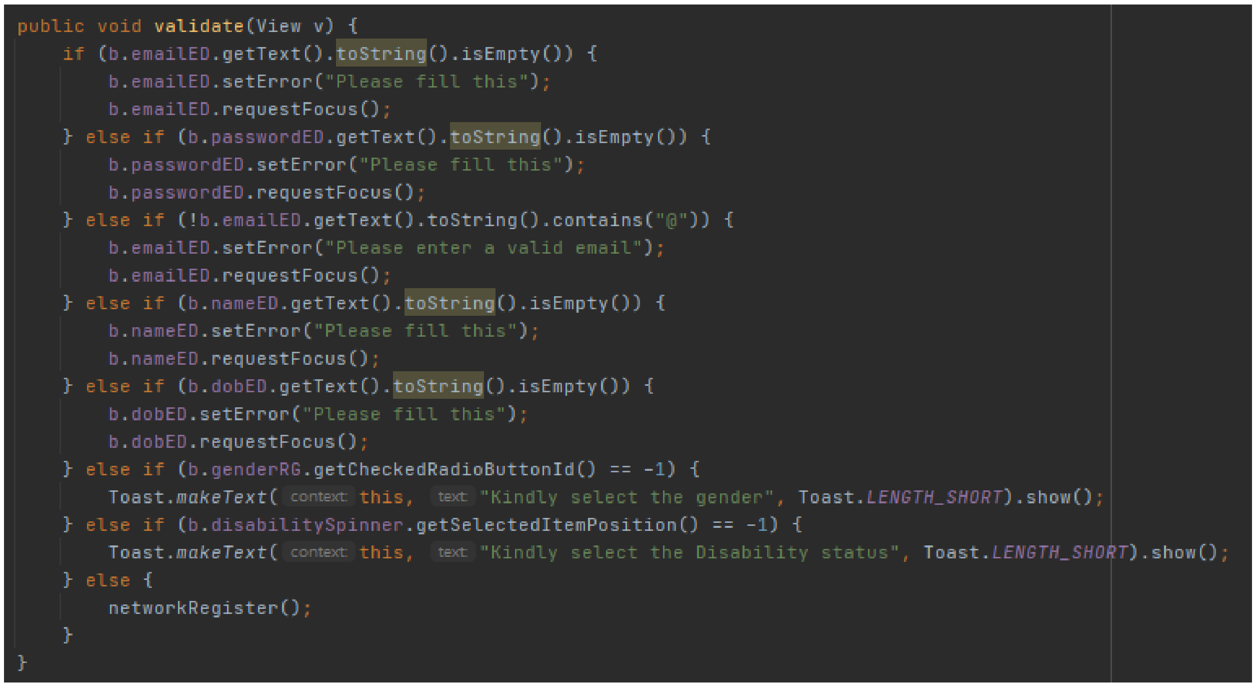
As shown above in Figure 39, The Author had created a method to save the user's information such as their name, email, image, phone, date of birth, gender, disability status, and location after the user is successfully registered to the application.



```java
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    switch (requestCode) {
        case MY_PERMISSION_REQUEST_CODE_PHONE_STATE: {
            if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {

                  permissionCheck(false);
                return;
            } else {
                requestLocation();
            }

        }
        break;
    }
}
```

Figure 40: Requesting Location Services from User

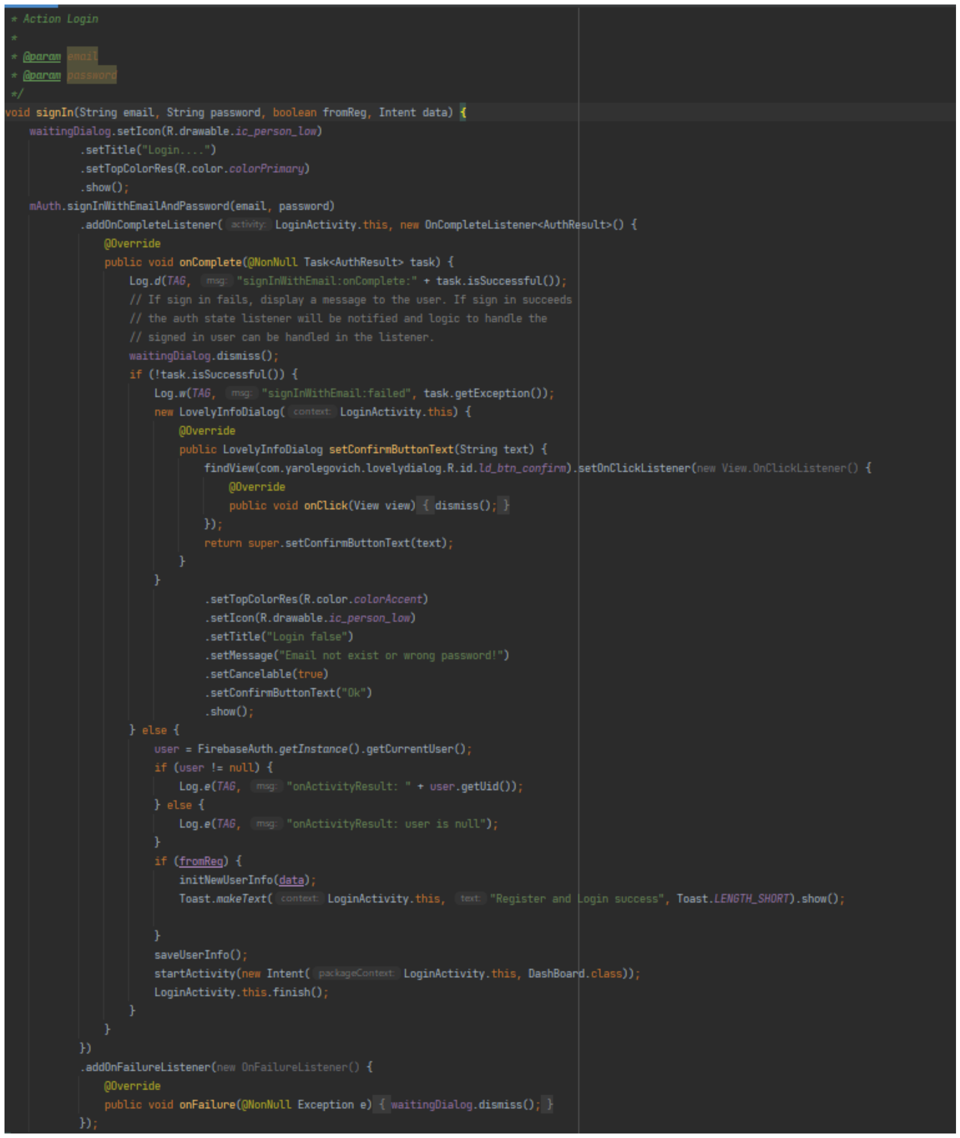
All of the information needed when signing up requires the user to input the related details except for the location. The location for the particular user when signing up is collected by sending a permissions request to the user to allow the app to collect the location information of the user. As you can see from the figure above, we set the ACCESS_FINE_LOCATION to request the user to share their location information. Once the user accepts the permissions the app would set their location on the find a buddy map.



```java
public void validate(View v) {
    if (b.emailED.getText().toString().isEmpty()) {
        b.emailED.setError("Please fill this");
        b.emailED.requestFocus();
    } else if (b.passwordED.getText().toString().isEmpty()) {
        b.passwordED.setError("Please fill this");
        b.passwordED.requestFocus();
    } else if (!b.emailED.getText().toString().contains("@")) {
        b.emailED.setError("Please enter a valid email");
        b.emailED.requestFocus();
    } else if (b.nameED.getText().toString().isEmpty()) {
        b.nameED.setError("Please fill this");
        b.nameED.requestFocus();
    } else if (b.dobED.getText().toString().isEmpty()) {
        b.dobED.setError("Please fill this");
        b.dobED.requestFocus();
    } else if (b.genderRG.getCheckedRadioButtonId() == -1) {
        Toast.makeText( context: this,  text: "Kindly select the gender", Toast.LENGTH_SHORT).show();
    } else if (b.disabilitySpinner.getSelectedItemPosition() == -1) {
        Toast.makeText( context: this,  text: "Kindly select the Disability status", Toast.LENGTH_SHORT).show();
    } else {
        networkRegister();
    }
}
```

Figure 41: Validation Check for Register

The Figure 41 above shows the method for validation checks for each information required when signing up. If a particular field is empty a message would display on the application notifying the user to fill out a particular missing field.



```
 * Action Login
 *
 * @param email
 * @param password
 */
void signIn(String email, String password, boolean fromReg, Intent data) {
    waitingDialog.setIcon(R.drawable.ic_person_low)
            .setTitle("Login....")
            .setTopColorRes(R.color.colorPrimary)
            .show();
    mAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener( activity: LoginActivity.this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    Log.d(TAG,  msg: "signInWithEmail:onComplete:" + task.isSuccessful());
                    // If sign in fails, display a message to the user. If sign in succeeds
                    // the auth state listener will be notified and logic to handle the
                    // signed in user can be handled in the listener.
                    waitingDialog.dismiss();
                    if (!task.isSuccessful()) {
                        Log.w(TAG,  msg: "signInWithEmail:failed", task.getException());
                        new LovelyInfoDialog( context: LoginActivity.this) {
                            @Override
                            public LovelyInfoDialog setConfirmButtonText(String text) {
                                findView(com.yarolegovich.lovelydialog.R.id.ld_btn_confirm).setOnClickListener(new View.OnClickListener() {
                                    @Override
                                    public void onClick(View view) { dismiss(); }
                                });
                                return super.setConfirmButtonText(text);
                            }
                        }
                                .setTopColorRes(R.color.colorAccent)
                                .setIcon(R.drawable.ic_person_low)
                                .setTitle("Login false")
                                .setMessage("Email not exist or wrong password!")
                                .setCancelable(true)
                                .setConfirmButtonText("Ok")
                                .show();
                    } else {
                        user = FirebaseAuth.getInstance().getCurrentUser();
                        if (user != null) {
                            Log.e(TAG,  msg: "onActivityResult: " + user.getUid());
                        } else {
                            Log.e(TAG,  msg: "onActivityResult: user is null");
                        }
                        if (fromReg) {
                            initNewUserInfo(data);
                            Toast.makeText( context: LoginActivity.this,  text: "Register and Login success", Toast.LENGTH_SHORT).show();

                        }
                        saveUserInfo();
                        startActivity(new Intent( packageContext: LoginActivity.this, DashBoard.class));
                        LoginActivity.this.finish();
                    }
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) { waitingDialog.dismiss(); }
            });
```

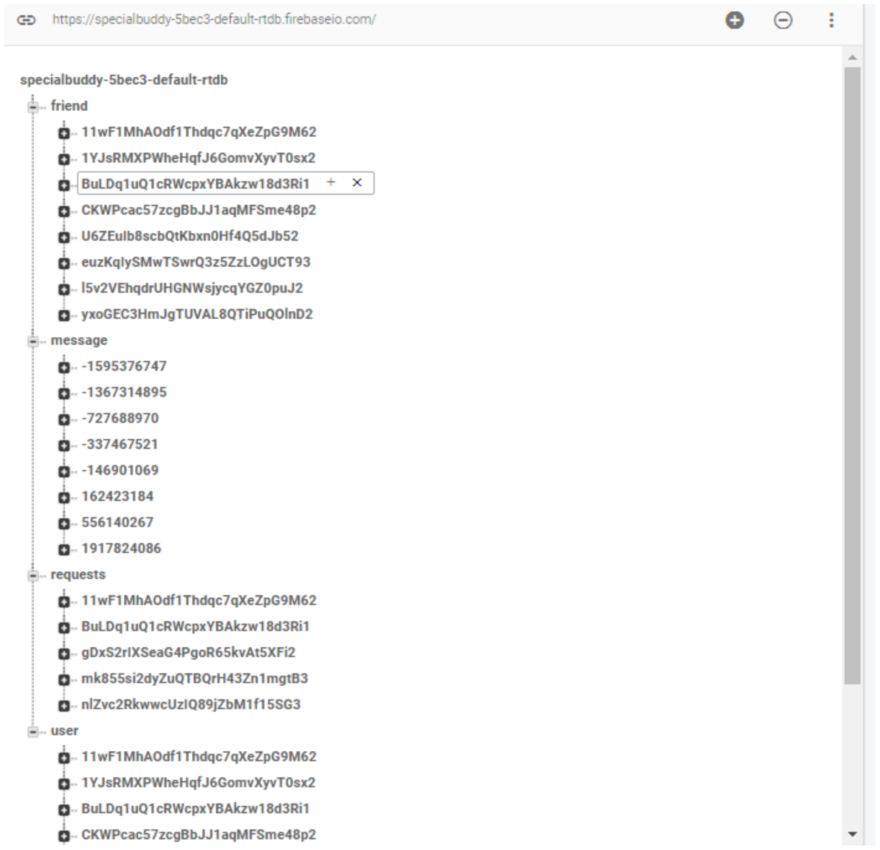
Figure 42: User Login

The second part of the Authentication process for the mobile application was creating the sign-in method to handle the login process for the mobile application. As you can see from Figure 42 above the method checks if the email is valid and already exists on the system, and then checks if the correct password is associated with that particular email. If the email and password match to a previously created account the application loads the user to their account, if not a message dialog would appear notifying the user that the email or password is incorrect.

The last part of the Authentication configuration and set up was the setting up and linking of the Authentication methods via the Firebase Authentication UI. The sign-in methods that was set up was the Email/Password Method.

5.3.2 Database Set-Up

For the database set-up the Author had used the Firebase Real-time Database as shown below in Figure 43. The Author did this by first creating a project on the Firebase UI Console, then selecting Realtime Databases configuring all the details in the UI such as the modes, region etc. After that the Author headed to the Project settings and configured the SDK and linked the Firebase Realtime Databases to the special buddy android mobile application.

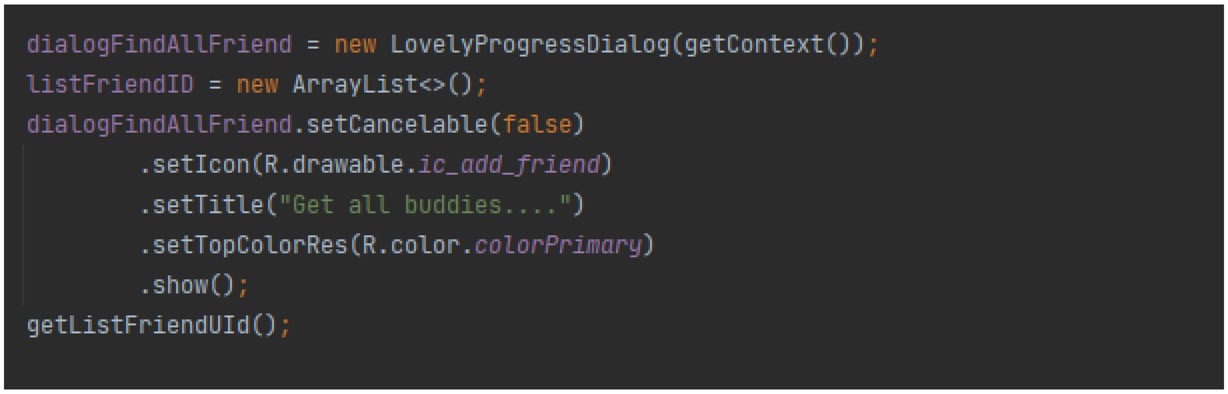


```
https://specialbuddy-5bec3-default-rtdb.firebaseio.com/

specialbuddy-5bec3-default-rtdb
  friend
    11wF1MhAOdf1Thdqc7qXeZpG9M62
    1YJsRMXPWheHqfJ6GomvXyvT0sx2
    BuLDq1uQ1cRWcpxYBAkzw18d3Ri1
    CKWPcac57zcgBbJJ1aqMFSme48p2
    U6ZEulb8scbQtKbxn0Hf4Q5dJb52
    euzKqlySMwTSwrQ3z5ZzLOgUCT93
    l5v2VEhqdrUHGNWsjycqYGZ0puJ2
    yxoGEC3HmJgTUVAL8QTiPuQOlnD2
  message
    -1595376747
    -1367314895
    -727688970
    -337467521
    -146901069
    162423184
    556140267
    1917824086
  requests
    11wF1MhAOdf1Thdqc7qXeZpG9M62
    BuLDq1uQ1cRWcpxYBAkzw18d3Ri1
    gDxS2rIXSeaG4PgoR65kvAt5XFi2
    mk855si2dyZuQTBQrH43Zn1mgtB3
    nlZvc2RkwwcUzIQ89jZbM1f15SG3
  user
    11wF1MhAOdf1Thdqc7qXeZpG9M62
    1YJsRMXPWheHqfJ6GomvXyvT0sx2
    BuLDq1uQ1cRWcpxYBAkzw18d3Ri1
    CKWPcac57zcgBbJJ1aqMFSme48p2
```

Figure 43: Firebase Realtime Database Data

5.3.3 Creating to Connect to a Buddy and Viewing List of Buddies

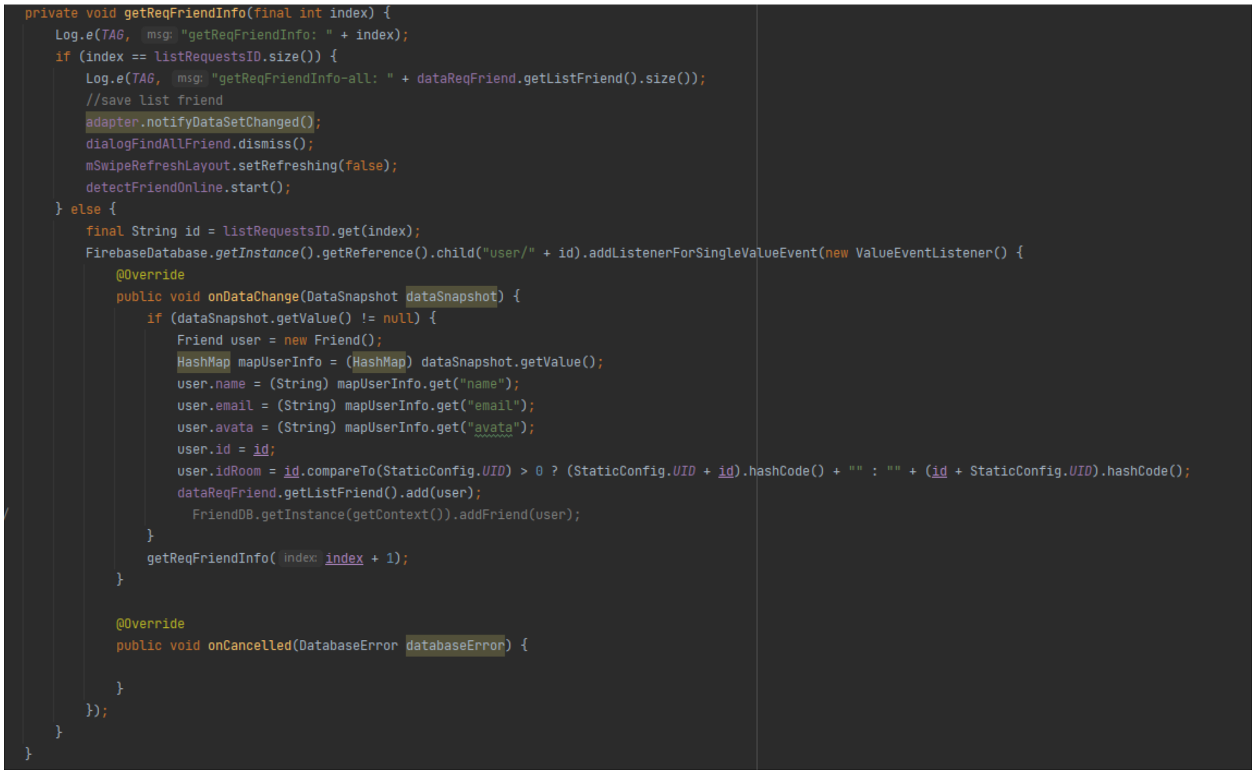
There were three parts involved in connecting to a buddy and then viewing a list of connected buddies. The Buddies Requests and Suggestion Screen had two parts, one for viewing a list of users using the application that a buddy can connect to and then a section where a user can view a pending request and accept and decline a request. The first thing the Author did was called the getListFreindUId() method as shown in Figure 44 which displays all the users that are using the mobile application and allows the buddy to connect with another buddy.



```
dialogFindAllFriend = new LovelyProgressDialog(getContext());
listFriendID = new ArrayList<>();
dialogFindAllFriend.setCancelable(false)
        .setIcon(R.drawable.ic_add_friend)
        .setTitle("Get all buddies....")
        .setTopColorRes(R.color.colorPrimary)
        .show();
getListFriendUId();
```

Figure 44: Calling method to Get a List of all Buddies

After that the Author created a method to get a friend request from a particular buddy as shown in Figure 45 and they can either accept or decline a pending request. If the buddy accepts a request the buddy gets added as a friend in the buddy's contacts list. But on the other hand, if the buddy declines a request the user does not get added as a buddy and does not display in the buddy's contacts list.



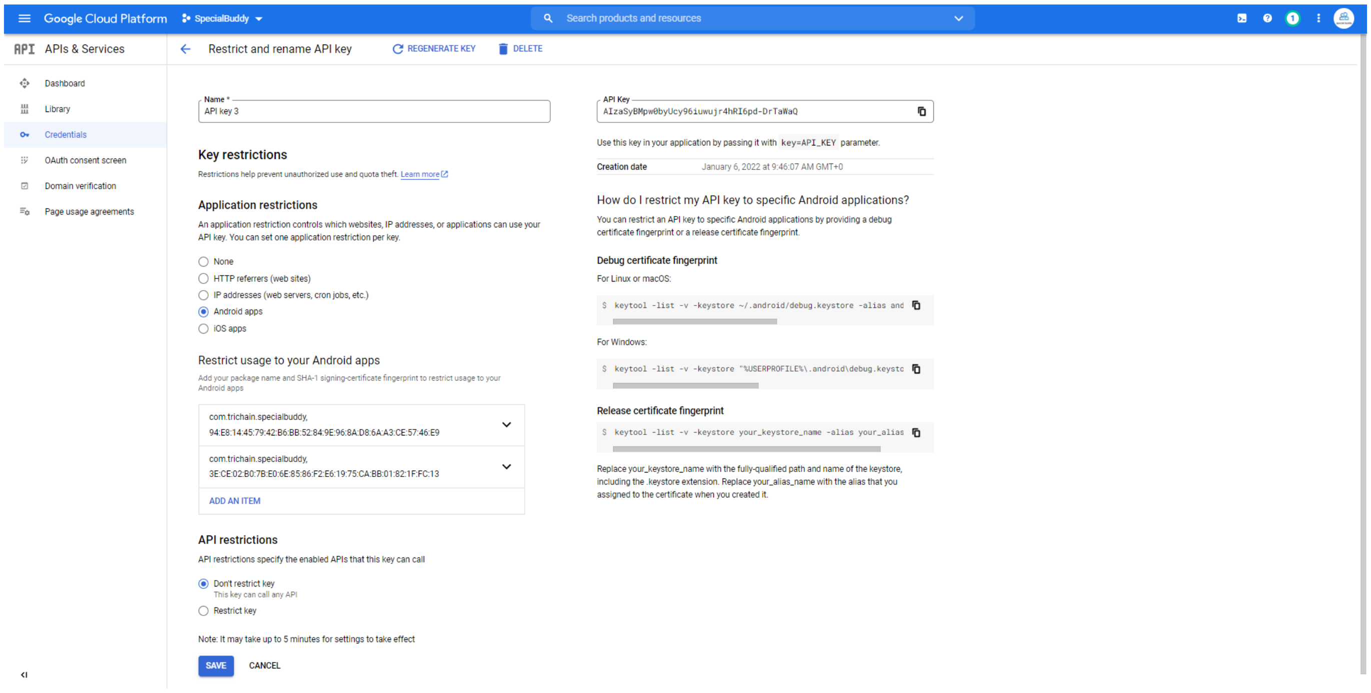
```
private void getReqFriendInfo(final int index) {
    Log.e(TAG,  msg: "getReqFriendInfo: " + index);
    if (index == listRequestsID.size()) {
        Log.e(TAG,  msg: "getReqFriendInfo-all: " + dataReqFriend.getListFriend().size());
        //save list friend
        adapter.notifyDataSetChanged();
        dialogFindAllFriend.dismiss();
        mSwipeRefreshLayout.setRefreshing(false);
        detectFriendOnline.start();
    } else {
        final String id = listRequestsID.get(index);
        FirebaseDatabase.getInstance().getReference().child("user/" + id).addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                if (dataSnapshot.getValue() != null) {
                    Friend user = new Friend();
                    HashMap mapUserInfo = (HashMap) dataSnapshot.getValue();
                    user.name = (String) mapUserInfo.get("name");
                    user.email = (String) mapUserInfo.get("email");
                    user.avata = (String) mapUserInfo.get("avata");
                    user.id = id;
                    user.idRoom = id.compareTo(StaticConfig.UID) > 0 ? (StaticConfig.UID + id).hashCode() + "" : "" + (id + StaticConfig.UID).hashCode();
                    dataReqFriend.getListFriend().add(user);
                      FriendDB.getInstance(getContext()).addFriend(user);
                }
                getReqFriendInfo( index: index + 1);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }
}
```

Figure 45:Method to Get Buddies Requests

The last part that was done by the Author was created a method to display all the list of friends that a particular buddy has. This would enable the user when clicked on the contacts list to display a list of friends that they are connected to, and in addition to searching for a particular friend if they have a large contact list.

5.3.4 Creating & Configuring the Geo-Location Services

For the Google Maps set up and configuration the Author first headed to the Google Cloud Platform Console and set the fingerprints and then copied the API key from the console as shown in Figure 46. Then the Author had created a new values file in Android Studio and pasted the API key as a string resource as shown in Figure 47 from console and set it in the values file which is then called into the Android Manifest file in the meta data as a string value.



Restrict and rename API key

Name *
API key 3

Key restrictions

Restrictions help prevent unauthorized use and quota theft. Learn more

Application restrictions

An application restriction controls which websites, IP addresses, or applications can use your API key. You can set one application restriction per key.

- None
- HTTP referrers (web sites)
- IP addresses (web servers, cron jobs, etc.)
- [✓ Android apps]
- iOS apps

Restrict usage to your Android apps

Add your package name and SHA-1 signing-certificate fingerprint to restrict usage to your Android apps

com.trichain.specialbuddy,
94:E8:14:45:79:42:B6:BB:52:84:9E:96:8A:D8:6A:A3:CE:57:46:E9

com.trichain.specialbuddy,
3E:CE:02:B0:7B:E0:6E:85:86:F2:E6:19:75:CA:BB:01:82:1F:FC:13

API restrictions

API restrictions specify the enabled APIs that this key can call

- [✓ Don't restrict key]
  This key can call any API
- Restrict key

Note: It may take up to 5 minutes for settings to take effect

API Key
AIzaSyBMpw0byUcy96iuwujr4hRI6pd-DrTaWaQ

Use this key in your application by passing it with `key=API_KEY` parameter.

Creation date January 6, 2022 at 9:46:07 AM GMT+0

How do I restrict my API key to specific Android applications?

You can restrict an API key to specific Android applications by providing a debug certificate fingerprint or a release certificate fingerprint.

Debug certificate fingerprint

For Linux or macOS:

```
$ keytool -list -v -keystore ~/.android/debug.keystore -alias and
```

For Windows:

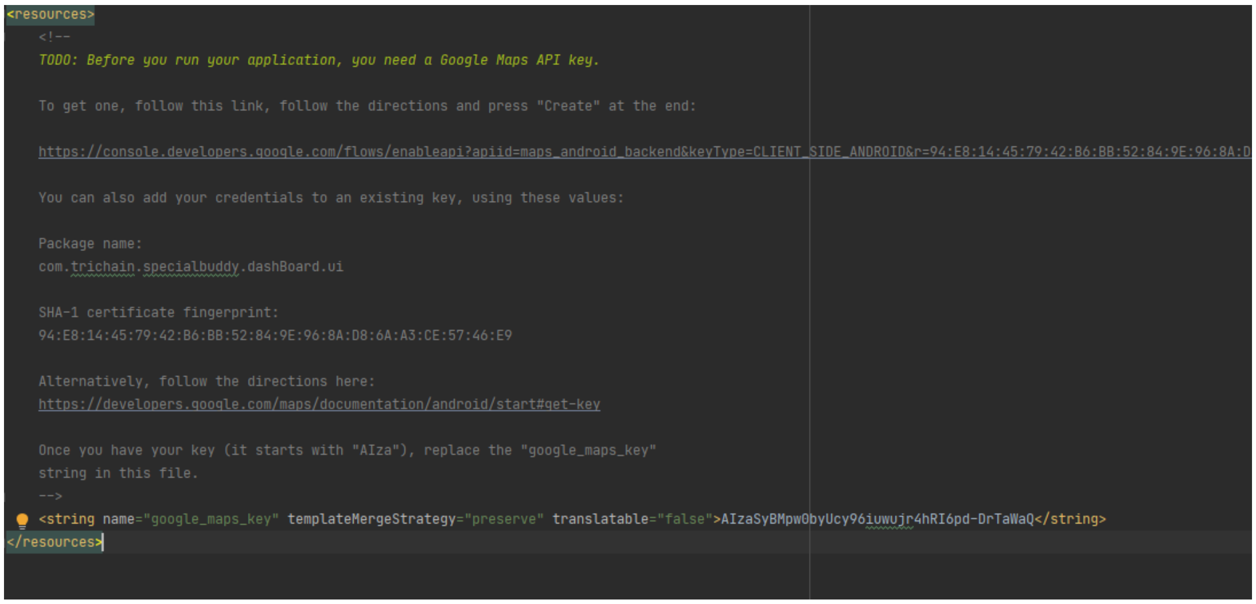
```
$ keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystc
```

Release certificate fingerprint

```
$ keytool -list -v -keystore your_keystore_name -alias your_alias
```

Replace your_keystore_name with the fully-qualified path and name of the keystore, including the .keystore extension. Replace your_alias_name with the alias that you assigned to the certificate when you created it.

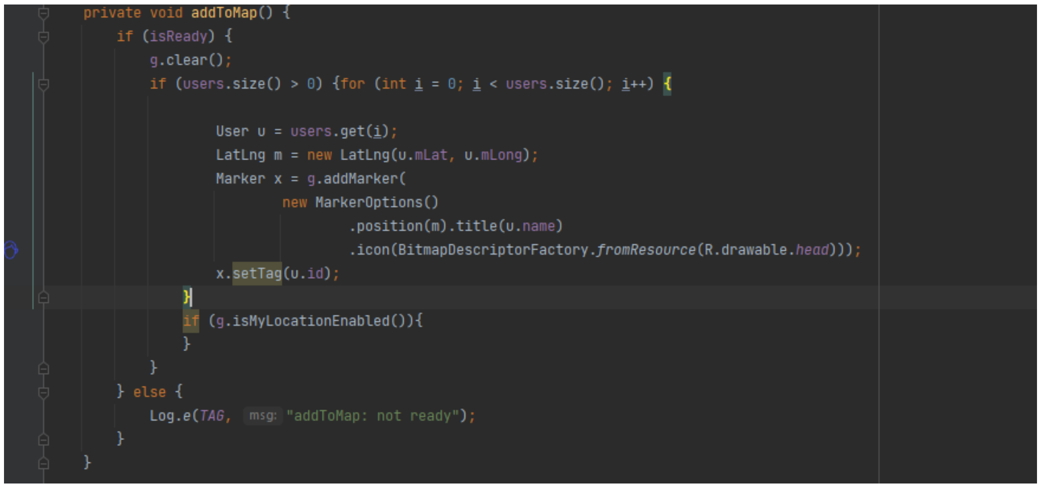
Figure 46: Google Cloud Platform for Google Maps API



```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:

    https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=94:E8:14:45:79:42:B6:BB:52:84:9E:96:8A:D

    You can also add your credentials to an existing key, using these values:

    Package name:
    com.trichain.specialbuddy.dashBoard.ui

    SHA-1 certificate fingerprint:
    94:E8:14:45:79:42:B6:BB:52:84:9E:96:8A:D8:6A:A3:CE:57:46:E9

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIzaSyBMpw0byUcy96iuwujr4hRI6pd-DrTaWaQ</string>
</resources>
```

Figure 47: Google Maps API Key Configuration in Android Studio

After that the Author had to create a method to display all users using the application on a google maps screen for the find a buddy screen on the mobile application. This was done by the Author by first gathering each user’s location using the for loop, after that the lat(latitude) and long(longitude) for each user were highlighted for each user using a profile icon on the google map with the use of markers which helped identify the users on the map as shown in Figure 48. Once the user clicks on a specific marker(buddy) it also displays a message showing the name of the user and if the user clicks on the name of the buddy its redirects to that specific buddy profile.

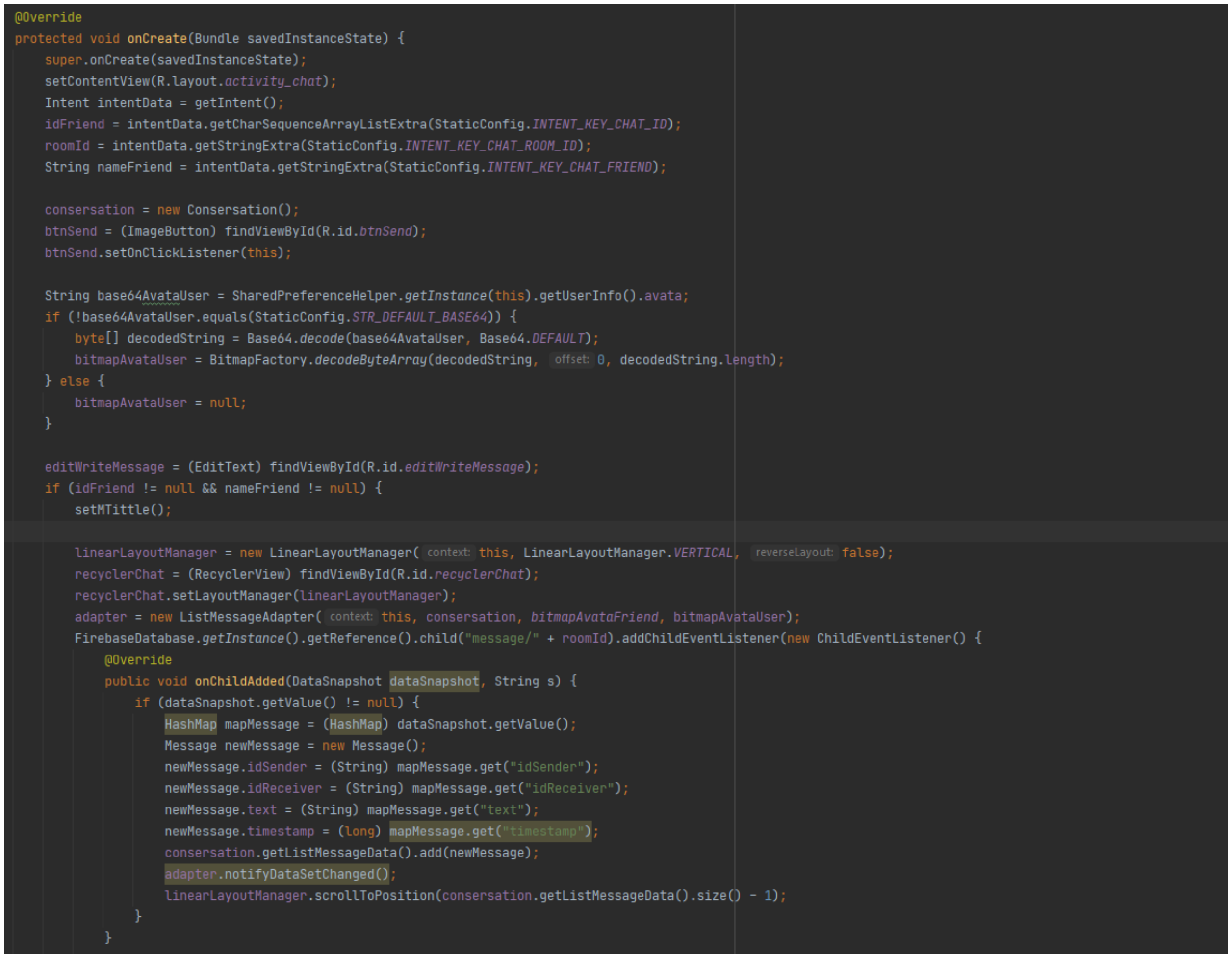


```
private void addToMap() {
    if (isReady) {
        g.clear();
        if (users.size() > 0) {for (int i = 0; i < users.size(); i++) {

                User u = users.get(i);
                LatLng m = new LatLng(u.mLat, u.mLong);
                Marker x = g.addMarker(
                        new MarkerOptions()
                                .position(m).title(u.name)
                                .icon(BitmapDescriptorFactory.fromResource(R.drawable.head)));
                x.setTag(u.id);
            }
            if (g.isMyLocationEnabled()){
            }
        }
    } else {
        Log.e(TAG,  msg: "addToMap: not ready");
    }
}
```

Figure 48: Adding User to Map

5.3.5 Creating & Configuring the Chat Service

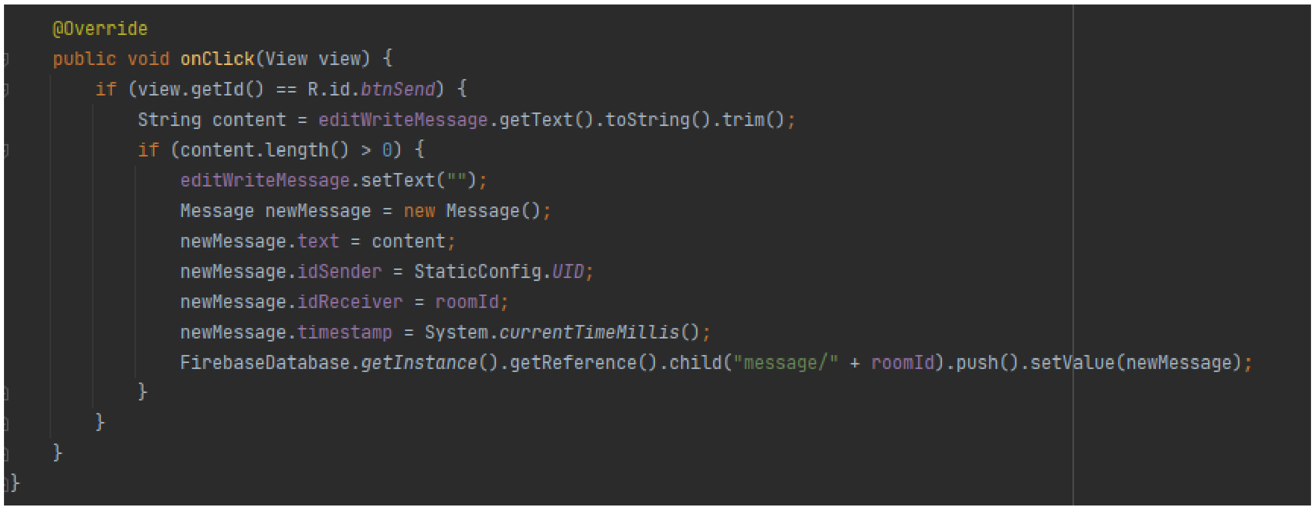
For creating the chat activity, the Author first designed the User interface of the page. After that the Author then went on to define the onCreate() method to handle what happens when the chat activity is started as shown in Figure 49. The Author first gathers the user's friend's id and other details of the friend and sets it in the header of the page. The Author then codes to display the chat history in a Recycler view in order to display large sets of data responsively on mobile. The Author also created a folder to store all the different data types for the messages, and conversations. Then the Author uses the Firebase Database to retrieve all the information of the chat history with the user's conversation with that particular buddy and pull all the details of each particular message with their relative sender's and receiver's ids and timestamps.



```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_chat);
    Intent intentData = getIntent();
    idFriend = intentData.getCharSequenceArrayListExtra(StaticConfig.INTENT_KEY_CHAT_ID);
    roomId = intentData.getStringExtra(StaticConfig.INTENT_KEY_CHAT_ROOM_ID);
    String nameFriend = intentData.getStringExtra(StaticConfig.INTENT_KEY_CHAT_FRIEND);

    consersation = new Consersation();
    btnSend = (ImageButton) findViewById(R.id.btnSend);
    btnSend.setOnClickListener(this);

    String base64AvataUser = SharedPreferenceHelper.getInstance(this).getUserInfo().avata;
    if (!base64AvataUser.equals(StaticConfig.STR_DEFAULT_BASE64)) {
        byte[] decodedString = Base64.decode(base64AvataUser, Base64.DEFAULT);
        bitmapAvataUser = BitmapFactory.decodeByteArray(decodedString, offset: 0, decodedString.length);
    } else {
        bitmapAvataUser = null;
    }

    editWriteMessage = (EditText) findViewById(R.id.editWriteMessage);
    if (idFriend != null && nameFriend != null) {
        setMTittle();

        linearLayoutManager = new LinearLayoutManager( context: this, LinearLayoutManager.VERTICAL, reverseLayout: false);
        recyclerChat = (RecyclerView) findViewById(R.id.recyclerChat);
        recyclerChat.setLayoutManager(linearLayoutManager);
        adapter = new ListMessageAdapter( context: this, consersation, bitmapAvataFriend, bitmapAvataUser);
        FirebaseDatabase.getInstance().getReference().child("message/" + roomId).addChildEventListener(new ChildEventListener() {
            @Override
            public void onChildAdded(DataSnapshot dataSnapshot, String s) {
                if (dataSnapshot.getValue() != null) {
                    HashMap mapMessage = (HashMap) dataSnapshot.getValue();
                    Message newMessage = new Message();
                    newMessage.idSender = (String) mapMessage.get("idSender");
                    newMessage.idReceiver = (String) mapMessage.get("idReceiver");
                    newMessage.text = (String) mapMessage.get("text");
                    newMessage.timestamp = (long) mapMessage.get("timestamp");
                    consersation.getListMessageData().add(newMessage);
                    adapter.notifyDataSetChanged();
                    linearLayoutManager.scrollToPosition(consersation.getListMessageData().size() - 1);
                }
            }
```

Figure 49: Viewing the Chat Activity

The Author had then proceeded to define what happens when the send button is clicked as shown in Figure 50 by the user in the chat activity. If there are no words inputted and the user clicks on the send button the message would not be inputted and saved into the database. However, if there is a message it saves the sender's and receiver's ids and timestamps and stores it into the Firebase Database.

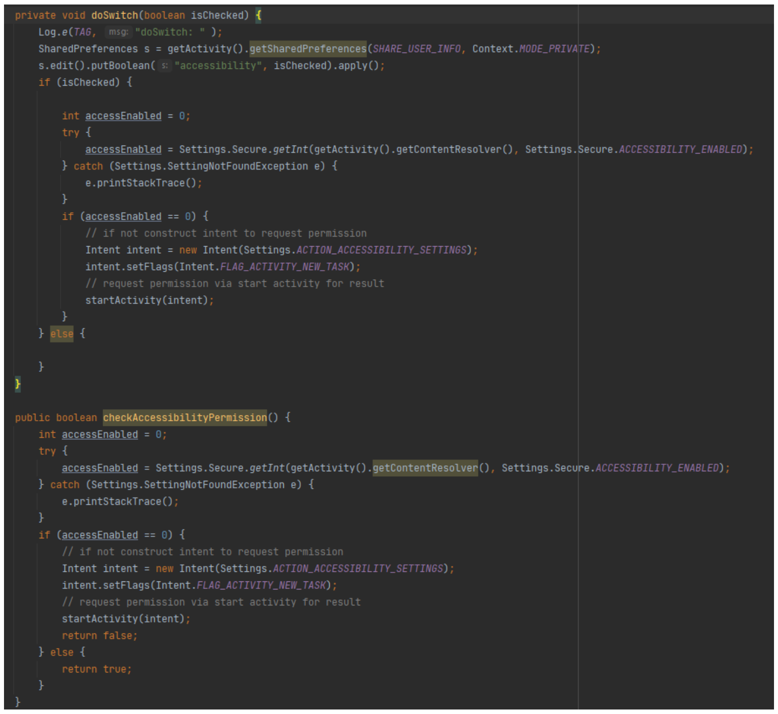


```
    @Override
    public void onClick(View view) {
        if (view.getId() == R.id.btnSend) {
            String content = editWriteMessage.getText().toString().trim();
            if (content.length() > 0) {
                editWriteMessage.setText("");
                Message newMessage = new Message();
                newMessage.text = content;
                newMessage.idSender = StaticConfig.UID;
                newMessage.idReceiver = roomId;
                newMessage.timestamp = System.currentTimeMillis();
                FirebaseDatabase.getInstance().getReference().child("message/" + roomId).push().setValue(newMessage);
            }
        }
    }
}
```

Figure 50: Sending Messages to other Users

5.3.6 Creating & Configuring the Accessibility Service

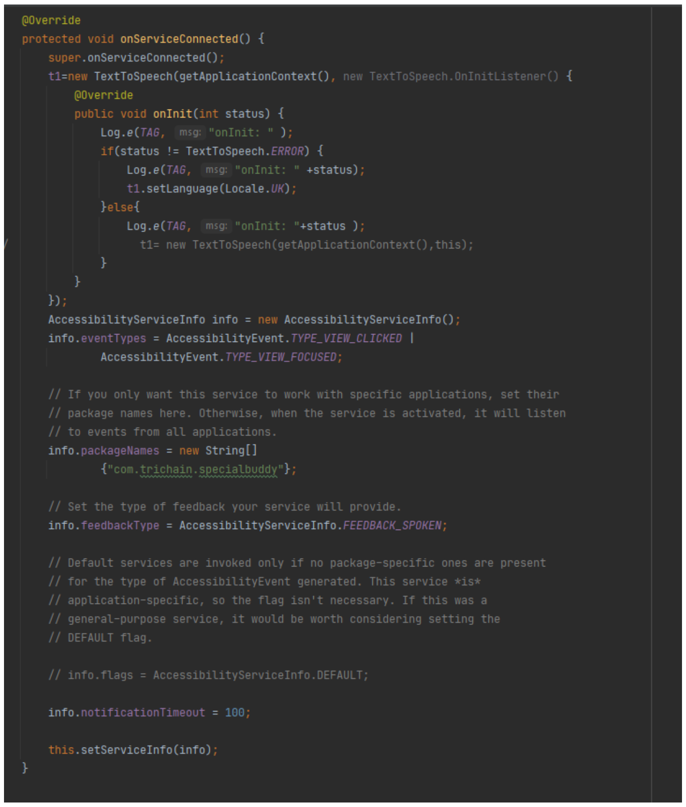
For the Accessibility settings and configurations, the first thing the Author had done was create a method to handle the toggle button to turn on and off the accessibility options. After that the Author created a method to check the accessibility permissions for the mobile application as shown in Figure 51.



```java
private void doSwitch(boolean isChecked) {
    Log.e(TAG,  msg: "doSwitch: " );
    SharedPreferences s = getActivity().getSharedPreferences(SHARE_USER_INFO, Context.MODE_PRIVATE);
    s.edit().putBoolean( s: "accessibility", isChecked).apply();
    if (isChecked) {

        int accessEnabled = 0;
        try {
            accessEnabled = Settings.Secure.getInt(getActivity().getContentResolver(), Settings.Secure.ACCESSIBILITY_ENABLED);
        } catch (Settings.SettingNotFoundException e) {
            e.printStackTrace();
        }
        if (accessEnabled == 0) {
            // if not construct intent to request permission
            Intent intent = new Intent(Settings.ACTION_ACCESSIBILITY_SETTINGS);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            // request permission via start activity for result
            startActivity(intent);
        }
    } else {

    }
}

public boolean checkAccessibilityPermission() {
    int accessEnabled = 0;
    try {
        accessEnabled = Settings.Secure.getInt(getActivity().getContentResolver(), Settings.Secure.ACCESSIBILITY_ENABLED);
    } catch (Settings.SettingNotFoundException e) {
        e.printStackTrace();
    }
    if (accessEnabled == 0) {
        // if not construct intent to request permission
        Intent intent = new Intent(Settings.ACTION_ACCESSIBILITY_SETTINGS);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        // request permission via start activity for result
        startActivity(intent);
        return false;
    } else {
        return true;
    }
}
```

Figure 51: Switch Toggle Button Configuration and Accessibility Permissions

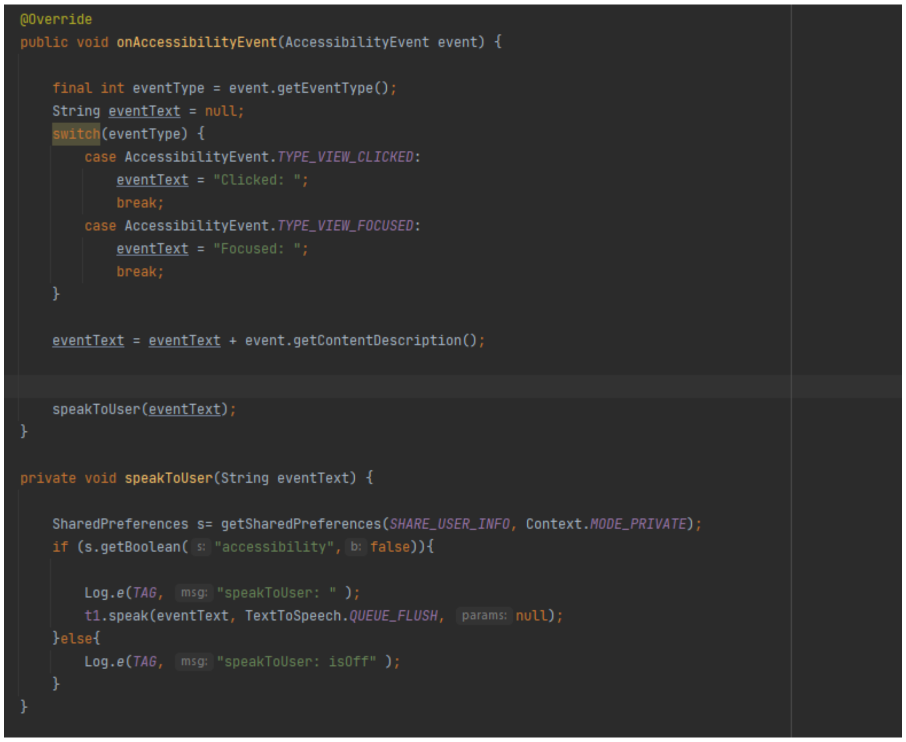
In Figure 52, The Author then defines when the Accessibility service is connected and also monitors the current state of the application service. In this instance the Author also sets the type of feedback service this application provides which is the Talk Back Feature.



```java
@Override
protected void onServiceConnected() {
    super.onServiceConnected();
    t1=new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {
        @Override
        public void onInit(int status) {
            Log.e(TAG,  msg: "onInit: " );
            if(status != TextToSpeech.ERROR) {
                Log.e(TAG,  msg: "onInit: " +status);
                t1.setLanguage(Locale.UK);
            }else{
                Log.e(TAG,  msg: "onInit: "+status );
                  t1= new TextToSpeech(getApplicationContext(),this);
            }
        }
    });
    AccessibilityServiceInfo info = new AccessibilityServiceInfo();
    info.eventTypes = AccessibilityEvent.TYPE_VIEW_CLICKED |
            AccessibilityEvent.TYPE_VIEW_FOCUSED;

    // If you only want this service to work with specific applications, set their
    // package names here. Otherwise, when the service is activated, it will listen
    // to events from all applications.
    info.packageNames = new String[]
            {"com.trichain.specialbuddy"};

    // Set the type of feedback your service will provide.
    info.feedbackType = AccessibilityServiceInfo.FEEDBACK_SPOKEN;

    // Default services are invoked only if no package-specific ones are present
    // for the type of AccessibilityEvent generated. This service *is*
    // application-specific, so the flag isn't necessary. If this was a
    // general-purpose service, it would be worth considering setting the
    // DEFAULT flag.

    // info.flags = AccessibilityServiceInfo.DEFAULT;

    info.notificationTimeout = 100;

    this.setServiceInfo(info);
}
```

Figure 52: Accessibility Service being Established

As shown in Figure 53, the Author created the onAccessibilityEvent() method to handle the Accessibility when the event is called. Then the Author in the speakToUser() method manages on when the Talk Back Feature is enabled.



```
@Override
public void onAccessibilityEvent(AccessibilityEvent event) {

    final int eventType = event.getEventType();
    String eventText = null;
    switch(eventType) {
        case AccessibilityEvent.TYPE_VIEW_CLICKED:
            eventText = "Clicked: ";
            break;
        case AccessibilityEvent.TYPE_VIEW_FOCUSED:
            eventText = "Focused: ";
            break;
    }

    eventText = eventText + event.getContentDescription();


    speakToUser(eventText);
}

private void speakToUser(String eventText) {

    SharedPreferences s= getSharedPreferences(SHARE_USER_INFO, Context.MODE_PRIVATE);
    if (s.getBoolean( s: "accessibility", b: false)){

        Log.e(TAG,  msg: "speakToUser: " );
        t1.speak(eventText, TextToSpeech.QUEUE_FLUSH,  params: null);
    }else{
        Log.e(TAG,  msg: "speakToUser: isOff" );
    }
}
```

Figure 53: When Accessibility Event and Speak to User Method is Called

5.4 Completed System

The system is set up with a completely working android application with an operational Firebase Realtime Database which supports authorization to allow sign-in and register. The Chat service is completely set up using firebase cloud messaging and accessibility features are also all fully implemented. The android also has the Geo-Location services implemented for the find a buddy screen with other features included such as the view and edit profile, contacts list, buddies' requests and suggestions, and file complaint on the mobile application.

5.5 Conclusion

The implementation of this project was a challenge but a very knowledgeable and rewarding experience as the Author had gained invaluable comprehension on android application development. The project is completed and follows exactly as from the design and analysis chapter was laid out, except for few minor changes as mentioned above.

6 Testing and Results

6.1 Testing and Results Overview

The final result of this process is a software system that was created to cater for an android application. This system consists of various technologies and incorporates the use of chat services, geo location, and accessibility features to successfully implement an application to communicate with people with/without disability. All the software components were tested to ensure that the functionalities operate as defined which in turn helps developing better cases for the system. The three types of software testing's that were conducted for this application were:

- Functional Testing
- Usability Testing
- Security Testing

6.2 Software Testing

6.2.1 Functional Testing

Test Case Name:	Create Account
Test Case ID	#001
Test Priority	High
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	This test allows the user to create an Account so that they would be able to use the application.
Pre-Condition	User needs to have the app downloaded and installed.
Test Steps	1. User clicks on the sign-up button and fills out details. 2. The system responds by checking if the email is unique and if so, it adds it to the database, and the account has been successfully created. 3. User is then redirected to their profile screen
Expected Results	If email is unique and data is entered correctly. User's account is created and redirected. If email is not unique, validation error on input box will appear.

	If fields are not entered correctly, validation error on input box will appear
Post-Condition	App should have transitioned to their profile screen.
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	Unique email, password, phone, date of birth, name, gender, and disability status, and allow geo location access.
Automation? (Yes/No)	No

Table 18: Create Account Test Case

Attachments:

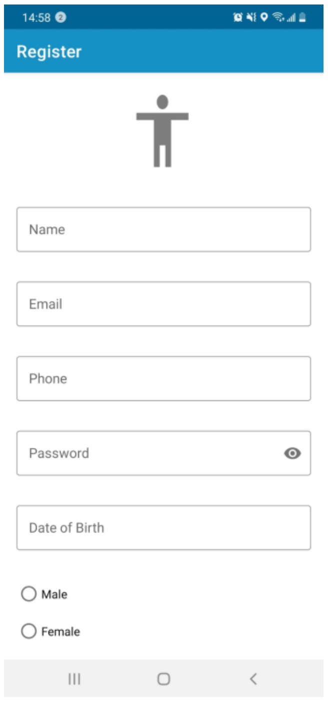


Figure 54: On Signup Page

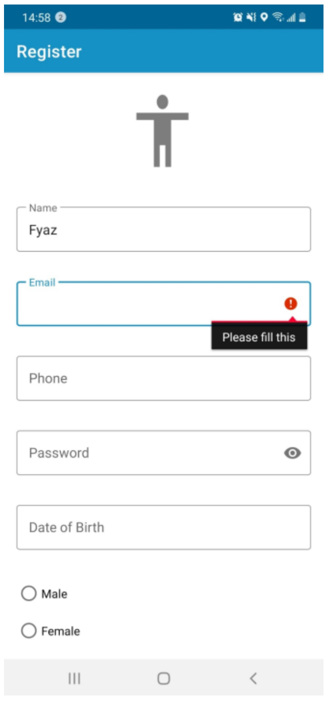


Figure 55: If field is empty

This is one of the test cases. Please check Appendix A for the complete set of functional test cases conducted for this project.

6.2.2 Usability Testing

To test the usability of the special buddy mobile application the Author had used to Think aloud technique. This was conducted by choosing 5 people with/without disability with different age groups to use the application and given specific tasks to complete such as sign up, login, chat, find and connect with a buddy etc. The test was guided by looking at the user experience and observing the level of efficiency of the mobile application. After the test the application did run smoothly, and the users were able to complete most of the tasks given by the Author. During the interview after the usability test using the Think Aloud Technique, the participants had indicated that the special buddy mobile application should allow them to sign up via Gmail or other Email platforms, rather than filling up the form in the sign-up page. The participants also recommended having in-built accessibility features rather than through the accessibility services from the phone's device settings. Finally, the users also suggested on having a toggle button that allows users to switch between light and dark mode as well.

6.2.3 Security Testing

Security testing was conducted on users account and their logging credentials. The data collected from users, friends, messaging, and requests are all hosted using Firebase Realtime Databases. Login authentication was also tested for security to ensure if an email and password was existing on the application to a previously created account.

7 Discussion

7.1 Overall Product Quality

The flow of this report illustrates how the final product was approached, and the final result reflects how this project was designed and implemented. Geo Location, Chat Services, Accessibility Features, Technology Application, and other elements combined together to form this mobile android application, Nonetheless, to have a good and durable special buddy mobile application project it requires high quality in order to compare with the existing solutions. The different test that was carried in this project indicate that the functionality of the project working, however, to improve the quality more testing need to be carried, and one test includes giving the mobile application to the targeted audience, to document their experience with the final product and study the different cases they might face while using it in. This would give a better indication of the level of success the device has achieved.

During the implementation of this project some decisions were changed/removed from the original idea. These changes were:

1. The Accessibility Features that were mentioned, where not fully implemented and also a few accessibility features were finally incorporated into the mobile application such as the Talk back Feature and Visual Features such as using big text, good coloring, and understandable descriptions for each field. This was due to time constraint.
2. Another Feature that was not implemented due to time restriction into the project was the admin panel where the admin could delete users based on complaints given by other users.
3. Another feature that was changed from the original objective was the different roles for the two different types of users (person with/without disability). The Author had implemented it as one type of user but in the database, it was collected as what type of the user they were. The purpose of this to remove redundancy of two separate account types as the two types of users (people with/without disability) had all the same functionality.
4. The adding of booking appointments was also not incorporated as there was a tight schedule to get the functionality to work.
5. The emergency bay that directly links to the local emergency's phone number was also not implemented due to restriction in time.

During the designing and development of this project there was some challenges related to the software aspects of the implementation. Some of these challenges include learning how to use new programs such as Android Studio for coding the mobile application and other tools such as Firebase, Google Maps API to incorporate their services into the mobile application.

7.2 Project Improvement

There were many improvements that the Author had recognized after completing the project in various aspects such as in the design, functionality, and durability. Improvements can be made to the project as mentioned below:

- Improving the application design and add more features into the application
- Change the logo design and colour to better understand the meaning of the idea and to make it more presentable
- Add more Accessibility Features to handle other types of people with disability
- Add the Admin panel to the mobile application to allow them to remove users from various complaints

7.3 Drawbacks of the system

The special buddy mobile android application works, and functions as expected. However, due to the fact of the deliverable deadline the performance was not the best. Also having a time constraint, the mobile application's design, accessibility features and other features mentioned above in the Overall Product Quality section could not be implemented and refined on. Nonetheless more improvements and features could be incorporated into the project in the future.

8 Conclusion

8.1 Summary and Improvements

To conclude, it is necessary to determine if the end results of this project has met the aims and objective at the start. The main goal of this project was to design an android based mobile application that would help people with disabilities to overcome their solitude and to interact with peers/buddies without much difficulty using a social online platform. The potential of learning new tools and technology was also one of the aims that was achieved during the development of this project. However, there was some changes in the objectives due to the time constraint. In addition, the project was done to study how an android based mobile applications can help people with disabilities to recover from solitude and loneliness.

The outcome of this project has successfully met the objectives. It offers good quality and performance with a straightforward mobile application the offers different functions for the user. However, it can be still improved and worked on for a better quality.

8.2 Benefits to people with/without disability

The special buddy mobile application project demonstrates a proof of concept only. It is a prototype idea that was implemented for the final year project. Further advice and rigorous testing are needed to determine if the special buddy mobile application can be used by anyone.

8.3 Future Work

This project can be taken to a new level with more work that could be implemented in it. For future work the special buddy application could implement more features for accessibility to cater for various more types of people with disability. The Special buddy application could also incorporate a speak message to allow users to speak the message to send rather than typing. Another feature that could be implemented into the application can be to include the admin side to the application so they could delete users based on filed complaints and also to add the Frequently Asked Questions page to help users with any questions they might have regarding the application.

Bibliography

Best Buddies International. 2021. e-Buddies - Best Buddies International. [online] Available at: <https://www.bestbuddies.org/what-we-do/ebuddies/> [Accessed 27 August 2021]. Statistics and Background Research

Liu, S., Xie, W., Han, S., Mou, Z., Zhang, X. and Zhang, L., 2021. Social Interaction Patterns of the Disabled People in Asymmetric Social Dilemmas. Background Research on Area of Study and Social Interaction behaviors within the disabled Community

Tough, H., Siegrist, J. and Fekete, C., 2017. Social relationships, mental health and wellbeing in physical disability: a systematic review. BMC Public Health, 17(1). Importance of social relationships and mental health for people with disabilities

Hardman, M. and Clark, C., 2006. Promoting Friendship Through Best Buddies: A National Survey of College Program Participants. Mental Retardation, 44(1), pp.56-63. Research Context on survey completed on area of research

SAGE Journals. 2021. Increasing Social Interactions for People with More Severe Learning Disabilities Who Have Difficulty Developing Personal Relationships - R. Whitehouse, P. Chamberlain,A.O'Brien,2001.[online]Availableat:<https://journals.sagepub.com/doi/10.1177/146900470100500301> [Accessed 27 August 2021]. Levels of social interaction and behavior for people who have learning difficulties

Anderson, R., 2021. Disabled and out? Social Interaction Barriers and Mental Health among Older Adults with Physical Disabilities. [online] DigitalCommons@University of Nebraska - Lincoln. Available at: <https://digitalcommons.unl.edu/sociologydiss/51/> [Accessed 27 August 2021].Social Isolation for people with disabilities on an older age scale

Hodapp, R. and Fidler, D., 2020. International Review Research in Developmental Disabilities. San Diego: Elsevier Science & Technology. In depth background research in Developmental Disabilities

Raising Children Network. 2021. Social communication disorder (SCD). [online] Availableat:<https://raisingchildren.net.au/guides/a-z-healthreference/scd#:~:text=Social%20communication%20disorder%20is%20a,communication%20appropriately%20in%20social%20situations.> [Accessed 27 August 2021]. In depth review of Social Communication Disorder (SCD)

Google Developers. 2021. Maps SDK for Android QuickStart | Google Developers. [online]Available at: https://developers.google.com/maps/documentation/androidsdk/start> [Accessed 17 September 2021]. Overview of Google Maps API and Reference Guides

Android Developers. 2021. Documentation | Android Developers. [online] Available at: <https://developer.android.com/docs> [Accessed 17 September 2021]. Android Studio Documentation

Firebase.2021.Documentation|Firebase.[online]Availableat:<https://firebase.google.com/docs> [Accessed 17 September 2021]. Firebase Documentation

Statista. 2021. Ireland mobile internet users 2016-2026 | Statista. [online] Available at:<https://www.statista.com/statistics/567200/predicted-number-of-mobileinternet-users-in-ireland/> [Accessed 9 October 2021]. Statistics for mobile users in Ireland

Statista. 2021. Forecast number of mobile users worldwide 2020-2025 | Statista.[online] Available at: <https://www.statista.com/statistics/218984/number-ofglobal-mobile-users-since-2010/> [Accessed 9 October 2021]. Statistics for mobile users globally

DataReportal – Global Digital Insights. 2021. Digital 2021: Global Overview Report — DataReportal–GlobalDigitalInsights.[online]Availableat: <https://datareportal.com/reports/digital-2021-global-overview-report> [Accessed 9 October 2021]. Statistics for global population and internet users

Loneliness, social support, social isolation and wellbeing among working age adults with and without disability: Cross-sectional study. (n.d.). PubMed Central (PMC). [online] Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7403030/>[Accessed 12 October 2021].

Appoly. 2021. The Importance of Accessibility in a Mobile App – Appoly Blog. [online] Available at: <https://www.appoly.co.uk/2020/10/26/the-importance-ofaccessibility-in-a-mobile-app/> [Accessed 12 October 2021].

Blog, W., 2021. Accessibility and your app design. [online] Windows Developer Blog.Availableat:<https://blogs.windows.com/windowsdeveloper/2016/09/13/accessibility-and-your-app-design/> [Accessed 12 October 2021].

Institute of Public Health. 2021. Press release: Loneliness emerging as a key public health challenge for the population during the pandemic - Institute of Public Health. [online] Availableat:<https://publichealth.ie/press-release-loneliness-emergingas-a-key-public-health-challenge-for-the-population-during-the-pandemic/> [Accessed 18 October 2021].

Tilda.tcd.ie.2021.[online]Availableat:<https://tilda.tcd.ie/publications/reports/pdf/Report_Loneliness.pdf> [Accessed 18 October 2021].

Medium. 2021. How to create an accessible app (and why you should). [online] Availableat: <https://medium.com/oberonamsterdam/how-to-create-an-accessibleapp-and-why-you-should-5493f41f8bdb> [Accessed 12 October 2021].

PubNub. 2021. What is a Geolocation API and how is it used?. [online] Available at:<https://www.pubnub.com/learn/glossary/what-is-a-geolocation-api/>[Accessed12 October 2021].

Helpshift.2021.WhatiChatService-Helpshift.[online]Availableat:
<https://www.helpshift.com/glossary/chatservice/#:~:text=A%20chat%20service%20is%20any,real%2Dtime%2C%20instantaneous%20conversation.> [Accessed 13 October 2021].

Statista. 2021. Loneliness and isolation by physical or mental health condition U.S. 2018|Statista.[online]Availableat:<https://www.statista.com/statistics/1082853/loneliness and-isolation-by-physical-or-mental-health-condition-us/> [Accessed 13 October 2021].

Tutorialspoint.com.2021.Android-Overview.[online]Availableat: <https://www.tutorialspoint.com/android/android_overview.htm> [Accessed 14 October 2021].

MUO.2021.TheProsandConsofAndroid.[online]Availableat:<https://www.makeuseof.com/android-pros-and-cons/> [Accessed 14 October 2021].

Medium. 2021. Advantages and Disadvantages of Android & iOS. [online] Availableat:<https://medium.com/@saranyaan2710/advantages-anddisadvantages-of android-ios-aa76e2b8f41> [Accessed 14 October 2021].

Investopedia.2021.AppleiOS.[online]Availableat:<https://www.investopedia.com/terms/a/apple-ios.asp> [Accessed 14 October 2021].

StatCounter Global Stats. 2021. Mobile & Tablet iOS Version Market Share Worldwide|StatcounterGlobalStats.[online]Availableat:<https://gs.statcounter.com/osversion-market-share/ios/mobile-tablet/worldwide> [Accessed 14 October 2021].

StatCounter Global Stats. 2021. Mobile & Tablet Android Version Market Share Worldwide|StatcounterGlobalStats.[online]Availableat:<https://gs.statcounter.com/os-version-market-share/android/mobile-tablet/worldwide> [Accessed 14 October 2021].

Business of Apps. 2021. Top Reasons to Choose the iOS Platform for Mobile App Development. [online] Available at: <https://www.businessofapps.com/news/topreasons-to-choose-ios-platform-for-mobile-app-development/> [Accessed 14 October 2021].

Digital Aptech. 2021. Advantages and Disadvantages of iOS | IOS App DevelopmentService.[online]Availableat:<https://www.digitalaptech.com/advantages-and-disadvantages-of-ios/> [Accessed 14 October 2021].

TechAhead. 2021. Android vs iOS - Which mobile platform is better in 2020?.[online] Available at: <https://www.techaheadcorp.com/blog/android-vs-ios/> [Accessed 14 October 2021].

Blog | Imaginary Cloud. 2021. Kotlin vs Java: the 12 differences you should know. [online] Availableat:<https://www.imaginarycloud.com/blog/kotlin-vsjava/#:~:text=Functional%20Programming-,Kotlin%20is%20a%20mix%20of%20object%2Doriented%20and%20functional%20programming,concept%20of%20object%2Doriented%20programming.&text=Primitive%20Types-,In%20Kotlin%2C%20as%20soon%20as%20you%20initiate%20a%20variable%20of,be%20automatically%20considered%20an%20object.> [Accessed 15 October 2021].

Lastovetska, A., 2021. Your Complete Guide to the Kotlin Programming Language. [online] Learn.g2.com. Available at: <https://learn.g2.com/what-is-kotlin> [Accessed 15 October 2021].

Oracle.com.2021.Whatisadatabase?.[online]Availableat:https://www.oracle.com/ie/database/what-is-database/> [Accessed 16 October 2021].

Khawas, C. and Shah, P., 2018. Application of Firebase in Android App Development-A Study. International Journal of Computer Applications, 179(46), pp.49-53.

OSDB. 2021. Firebase Pros and Cons: When You Should and Shouldn’t Use Firebase | OSDB. [online] Available at: <https://osdb.io/firebase-pros-and-conswhen-you-should-and-shouldnt-use-firebase-osdb/#ib-toc-anchor-20> [Accessed 16 October 2021].

Firebase. 2021. Firebase Realtime Database | Firebase Documentation. [online] Available at: <https://firebase.google.com/docs/database> [Accessed 16 October 2021].

www.javatpoint.com. 2021. SQLite Advantages and Disadvantages - javatpoint. [online]Availableat:<https://www.javatpoint.com/sqlite-advantages-anddisadvantages> [Accessed 16 October 2021].

Sqlite.org.2021.SQLiteHomePage.[online]Available at:<https://www.sqlite.org/index.html> [Accessed 16 October 2021].

Amazon Web Services, Inc. 2021. Fast NoSQL Key-Value Database – Amazon DynamoDB–AmazonWebServices.[online]Availableat: <https://aws.amazon.com/dynamodb/> [Accessed 16 October 2021].

DynamoDB, A., 2021. Advantages and Disadvantages of DynamoDB. [online] Infiflex.com.Availableat:<https://www.infiflex.com/advantages-anddisadvantages-of-dynamodb> [Accessed 16 October 2021].

Gravitate. 2021. What is Geolocation? How it Works & Why it Matters | Gravitate.[online]Availableat:<https://www.gravitatedesign.com/blog/what-is-geolocation/> [Accessed 17 October 2021].

Level Access. 2021. The Mobile Accessibility Landscape - Level Access. [online] Available at: <https://www.levelaccess.com/the-mobile-accessibility-landscape/> [Accessed 17 October 2021].

Support.google.com. 2021. Android accessibility overview - Android Accessibility Help.[online]Availableat:<https://support.google.com/accessibility/android/answer/6006564?hl=en> [Accessed 17 October 2021].

Android Developers. 2021. Create your own accessibility service | Android Developers.[online]Availableat:<https://developer.android.com/guide/topics/ui/accessibility/service> [Accessed 17 October 2021].

Android Developers. 2021. Make apps more accessible | Android Developers. [online]Availableat:<https://developer.android.com/guide/topics/ui/accessibility/apps> [Accessed 17 October 2021].

Sendbird. 2021. Android chat tutorial: How to build a messaging UI - Sendbird. [online] Available at: <https://sendbird.com/developer/tutorials/android-chattutorial-building-a-messaging-ui> [Accessed 17 October 2021].

Android Developers. 2021. Android Studio features | Android Developers. [online] Available at: <https://developer.android.com/studio/features> [Accessed 29 October 2021].

Support.google.com. 2021. Use the Accessibility Menu - Android Accessibility Help.[online]Availableat:<https://support.google.com/accessibility/android/answer/9078941?hl=en> [Accessed 31 October 2021].

Getstream.io.2021.[online]Availableat:<https://getstream.io/tutorials/androidchat/?language=java> [Accessed 31 October 2021].

Appendix A – Technical Specifications

Programming Language: Java with XML Syntax

Application Compatibility: Android Operating System

Development IDE & Version: Android Studio Jelly Bean 4.2 (17)

Other Tools/Software's: Firebase and Google Maps API

Appendix B – Complete set of Functional Test Cases

Login

Test Case Name:	Login
Test Case ID	#002
Test Priority	High
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	This test allows the user to log-in to a previously created account by that user.
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account.
Test Steps	1. User enters the email and password associated with their account. 2. The system responds by checking if the email and password match the email and password of a previously created account. 3. User successfully logs in and is redirected to their profile page.
Expected Results	If the email and password match an account from the database, the user is logged in. If email and password do not match a previously created account, the user fails to login. If fields are not entered correctly, a validation error message on the input box will appear.
Post-Condition	App should have transitioned to their profile screen
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	Email and password that matches an account previously created.
Automation? (Yes/No)	No

Table 19: Login Test Case

Attachments:

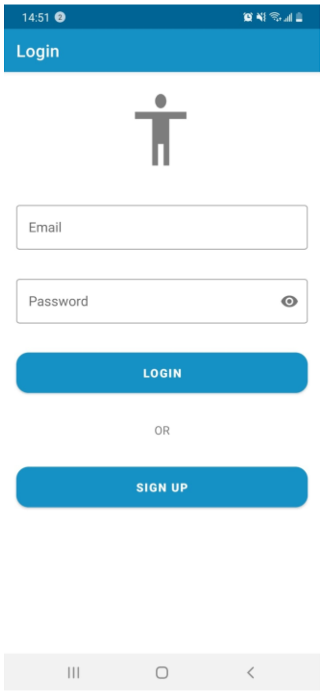


Figure 56: On Login Page

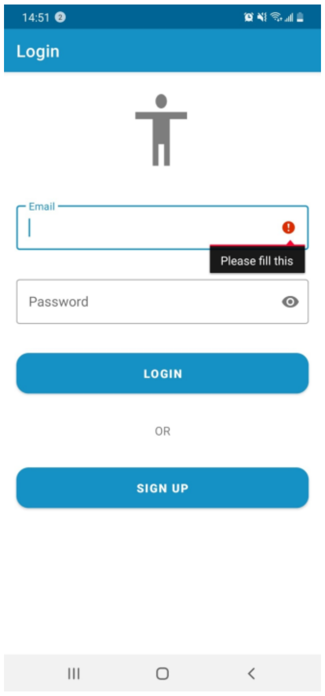
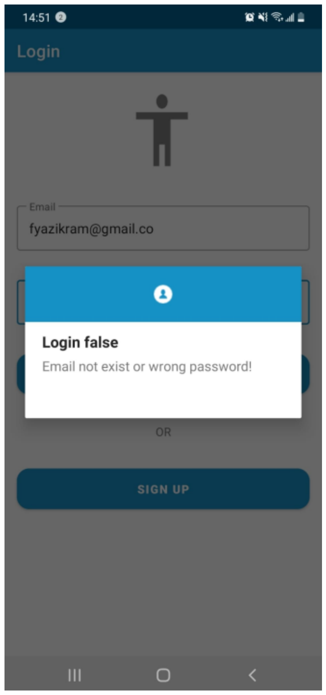


Figure 57: If field is empty



Login false

Email not exist or wrong password!

Figure 58: If email/password is does not match to created account

Logout

Test Case Name:	Logout
Test Case ID	#003
Test Priority	High
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to logout of a previously created account by that user.
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider. 3. The user then clicks the Logout from the navigation menu.
Expected Results	The user gets logged out of their account upon click.
Post-Condition	App should have transitioned back to the login screen
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 20: Logout Test Case

Attachments:

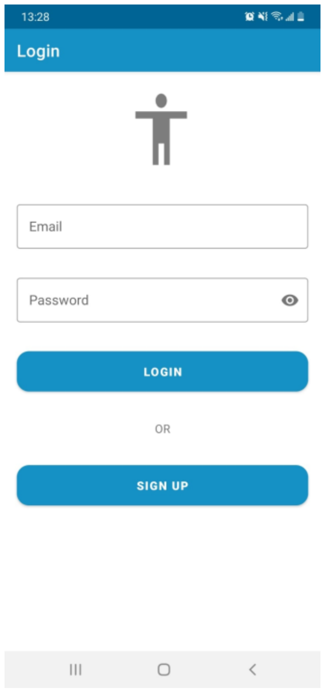


Figure 59: After Clicking Logout

Chat Between Users

Test Case Name:	Chat Between Users
Test Case ID	#004
Test Priority	High
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to chat between different users.
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider. 3. The user then clicks the Contacts List from the navigation menu. 4. The system responds by displaying a list of the users connected buddies. 5. The user chooses a buddy. 6. The system responds by displaying the chat screen allowing the user to send and view messages to the buddy.
Expected Results	The user can then type/view a message and send to the buddy.
Post-Condition	App should stay on the chat screen until the user clicks on the back button.
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 21: Chat Between Users Test Case

Attachments:

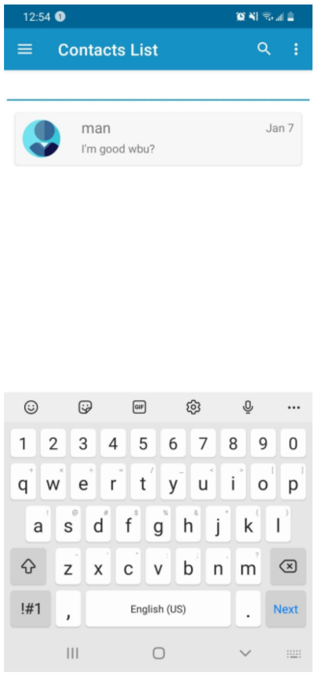
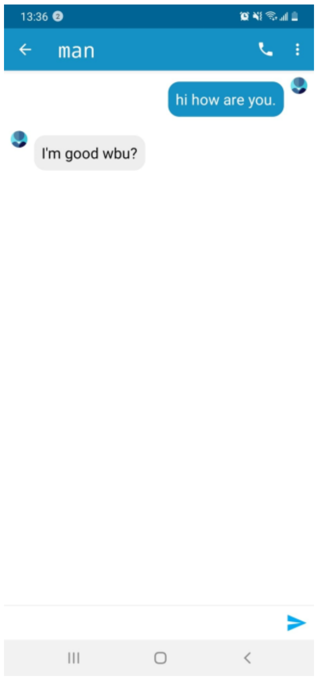


Figure 60: On clicking the contacts list



hi how are you.

I'm good wbu?

Figure 61: After clicking on user to chat with

View and use Accessibility Services

Test Case Name:	View and use Accessibility Services
Test Case ID	#005
Test Priority	High
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to enable accessibility services within the app.
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider. 3. The user then clicks the Enable Accessibility from the navigation menu.
Expected Results	The user can enable accessibility for the app upon click.
Post-Condition	App should stay on the Enable Accessibility screen until the user clicks on the back button.
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 22: View and use Accessibility Services

Attachments:

12:59
Buddies Requests and Sug...

Enable Accessibility service

Figure 62: On clicking the Enable Accessibility

View Connected Buddies from Search List

Test Case Name:	View Connected Buddies from Search List
Test Case ID	#006
Test Priority	High
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to view connected buddies from search list.
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider. 3. The user then clicks the Contacts List from the navigation menu.
Expected Results	The user can view or search from the list of connected buddies upon click.
Post-Condition	App should stay on the contacts list screen until the user clicks on the back button.
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 23: View Connected Buddies from Search List Test Case

Attachments:

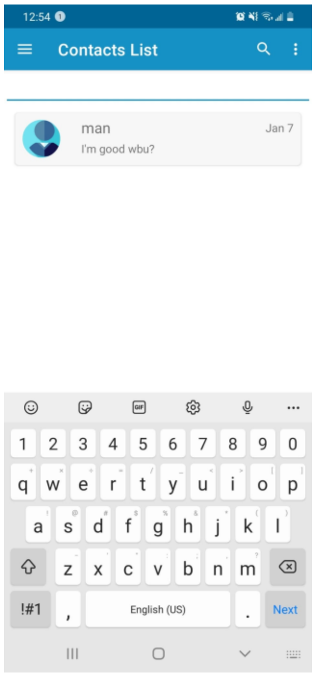


Figure 63: On clicking the contacts list

View Buddies using app from Google Maps and View their Profiles

Test Case Name:	View Buddies using app from Google Maps and View their Profiles
Test Case ID	#007
Test Priority	High
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to view other users using the app from Google maps and view their profiles
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider. 3. The user then clicks Find a Buddy from the navigation menu.
Expected Results	The user can then view users using the application using Google Maps upon click and when the user selects a user from the map it displays their name and once the user clicks the name it would redirect them to the other user's profile so they can view.
Post-Condition	App should have transitioned to the other buddy's profile screen if user clicks on another user's name from the map, if not app should stay in the find a buddy screen (Google Map).
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 24: View Buddies using app from Google Maps and View their Profiles Test Case

Attachments:

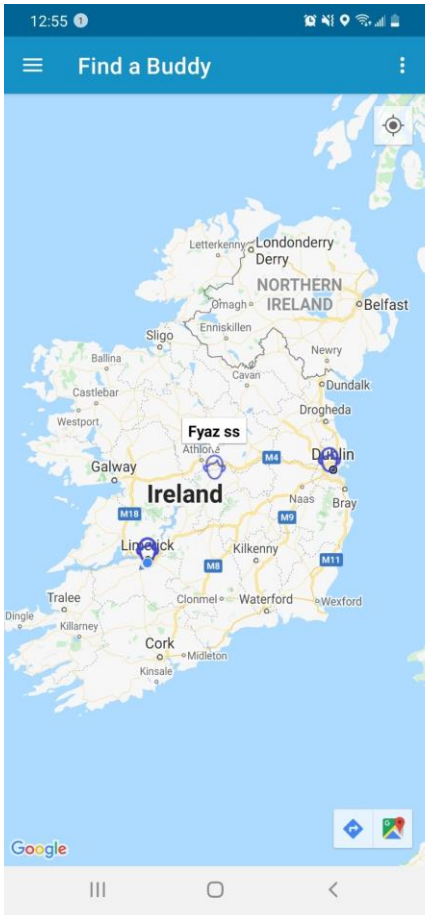
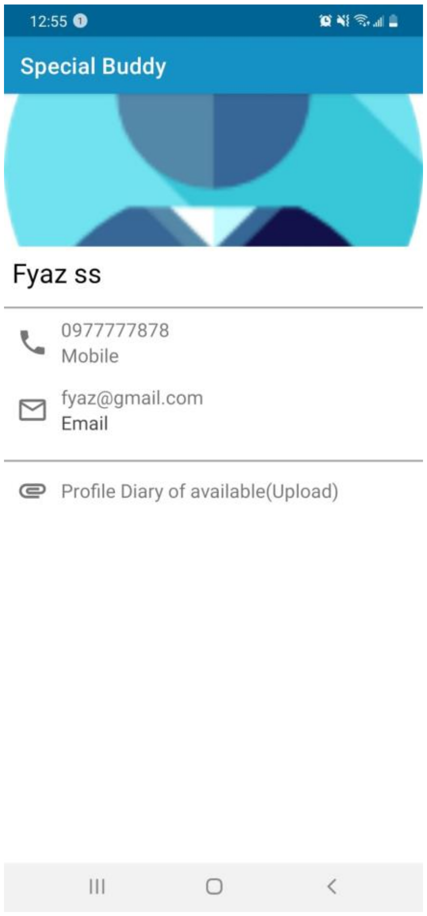


Figure 64: On clicking the find a buddy


Special Buddy

Fyaz ss

0977777878
Mobile

fyaz@gmail.com
Email

Profile Diary of available(Upload)

Figure 65: On clicking the buddies name from map

View User’s Personal Profile

Test Case Name:	View User’s Personal Profile
Test Case ID	#008
Test Priority	Low
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to view their personal profile.
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider. 3. The user then clicks the My Profile from the navigation menu.
Expected Results	The user can view their own profile upon click.
Post-Condition	App should stay on my profile screen until the user clicks on the back button.
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 25: View User’s Personal Profile Test Case

Attachments:

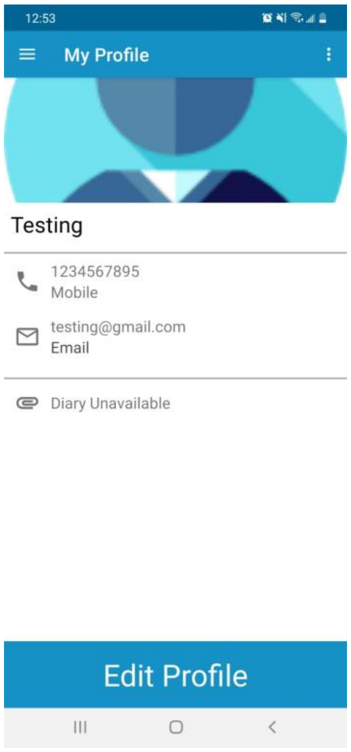


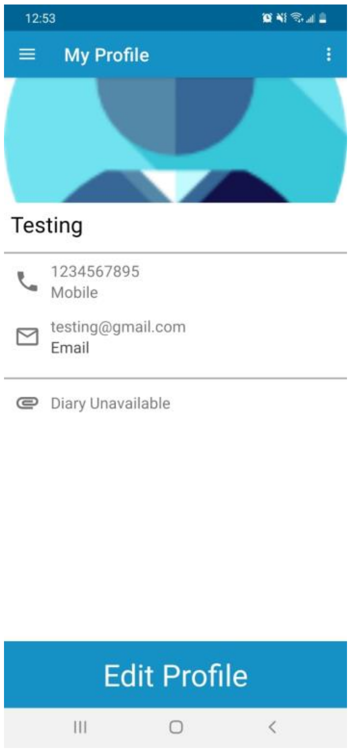
Figure 66: On clicking my profile

Edit User's Personal Profile

Test Case Name:	Edit User's Personal Profile
Test Case ID	#009
Test Priority	Low
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to edit their personal profile.
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider. 3. The user then clicks the My Profile from the navigation menu. 4. The system displays their profile and allows the user to click on the edit button. 5. The user clicks the edit button
Expected Results	The user can edit their own profile upon click.
Post-Condition	App should stay on the edit profile screen until the user clicks save and clicks on the back button.
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 26: Edit User's Personal Profile Test Case

Attachments:


12:53

My Profile

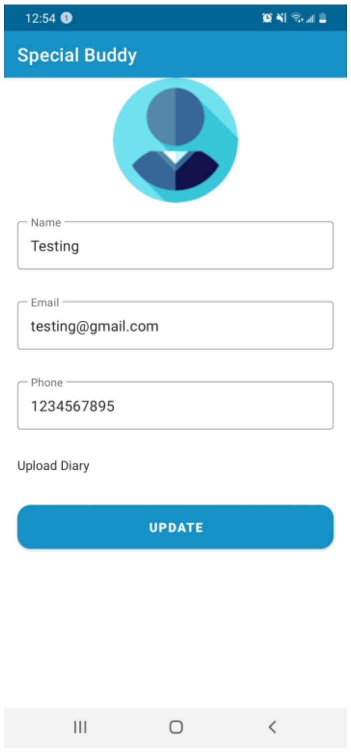
Testing

1234567895
Mobile

testing@gmail.com
Email

Diary Unavailable

Figure 67: On clicking my profile


12:54

Special Buddy

Name: Testing

Email: testing@gmail.com

Phone: 1234567895

Figure 68: On clicking Edit Profile button

Add Diary to Personal Profile

Test Case Name:	Add Diary to Personal Profile
Test Case ID	#010
Test Priority	Low
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to add diary to their personal profile.
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider. 3. The user then clicks the My Profile from the navigation menu. 4. The system displays their profile and allows the user to click on the edit button. 5. The user clicks the edit button. 6. The system responds by allowing the user to upload a diary to their profile.
Expected Results	The user can then add diary to their own profile upon click.
Post-Condition	App should stay on the edit profile screen until the user clicks save and clicks on the back button.
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 27: Add Diary to Personal Profile Test Case

Attachments:

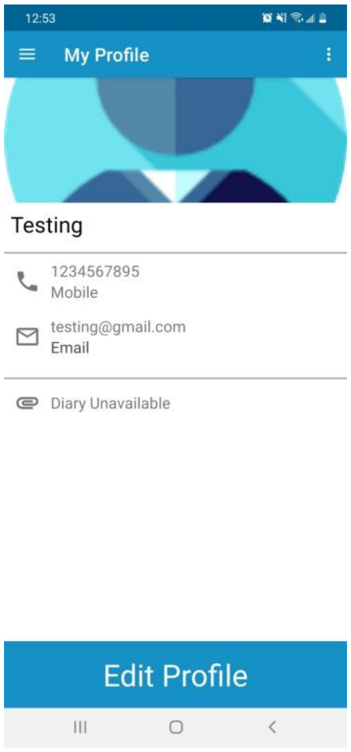


Figure 69: On clicking my profile

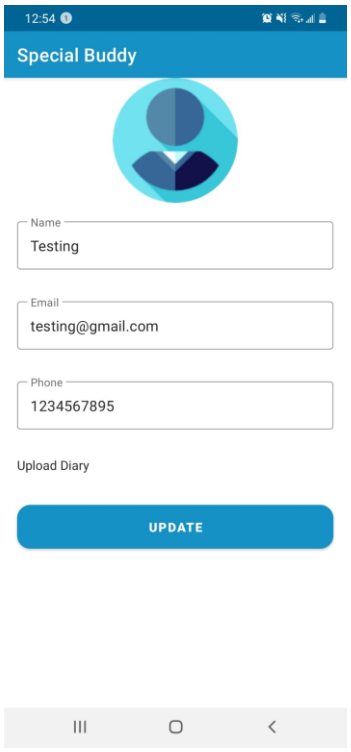


Figure 70: On clicking Edit Profile button

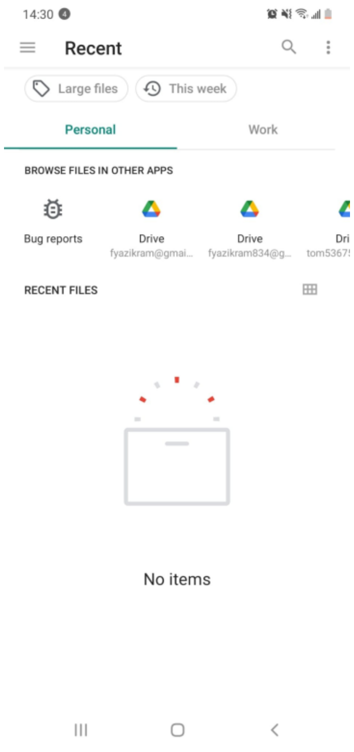


Figure 71: On clicking Upload Diary

Connect with a Buddy

Test Case Name:	Connect with a Buddy
Test Case ID	#011
Test Priority	High
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to connect with a buddy.
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider. 3. The user then clicks the Buddies Requests and Suggestions from the navigation menu.
Expected Results	The user can then view other buddies and click connect to connect with a buddy, which would then send a request to the other buddy.
Post-Condition	App should stay on the Buddies Requests and Suggestions screen until the user clicks on back button
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 28: Connect with a Buddy Test Case

Attachments:

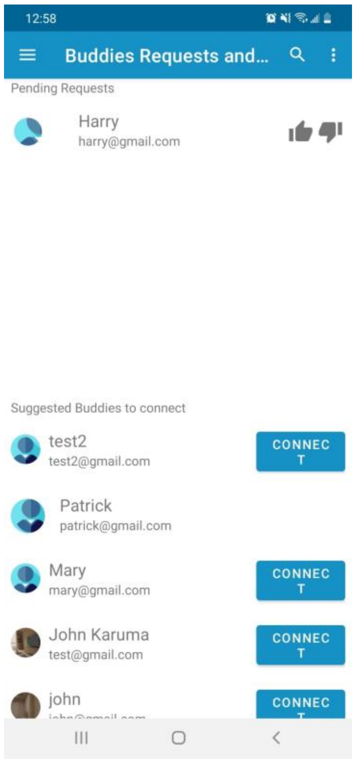


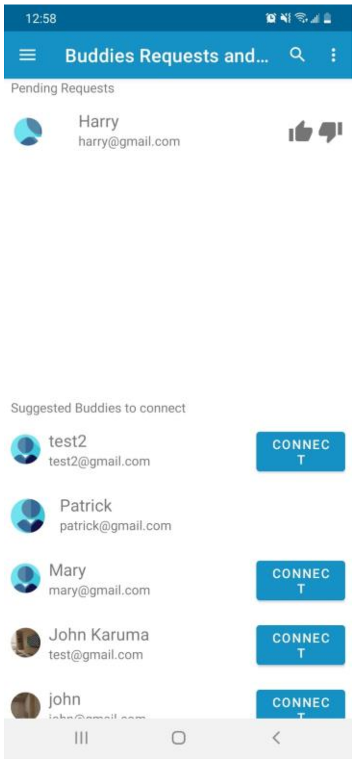
Figure 72: On clicking Buddies requests and Suggestions

Accept and Decline Buddy Requests

Test Case Name:	Accept and Decline Buddy Requests
Test Case ID	#012
Test Priority	High
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to accept or decline buddy requests.
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider. 3. The user then clicks the Buddies Requests and Suggestions from the navigation menu.
Expected Results	The user can then view their pending requests and either accept or decline requests from clicking on the up and down thumbs icons.
Post-Condition	App should stay on the Buddies Requests and Suggestions screen until the user clicks on back button
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 29: Accept and Decline Buddy Requests Test Case

Attachments:



12:58

Buddies Requests and...

Pending Requests

Harry
harry@gmail.com

Suggested Buddies to connect

test2
test2@gmail.com
CONNECT

Patrick
patrick@gmail.com

Mary
mary@gmail.com
CONNECT

John Karuma
test@gmail.com
CONNECT

john
CONNECT

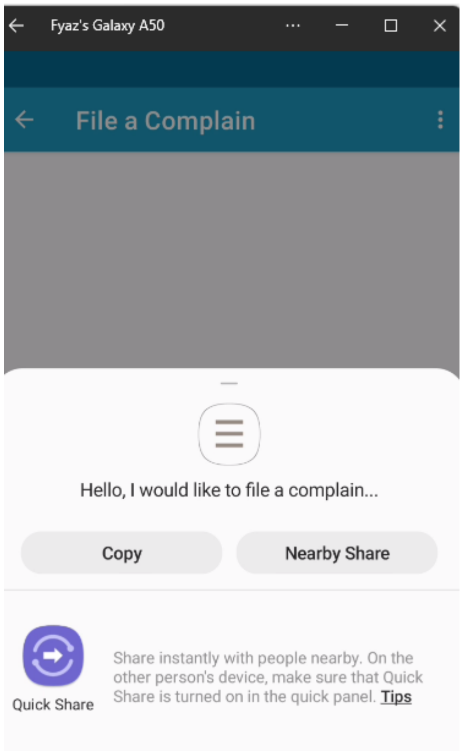
Figure 73: On clicking Buddies requests and Suggestions

File Complaint

Test Case Name:	File Complaint
Test Case ID	#013
Test Priority	Low
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to file a complaint
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider. 3. The user then clicks the File Complaint from the navigation menu.
Expected Results	The user can the file a complaint upon click.
Post-Condition	App should have transitioned to a sharing popup that allows the user to file a complaint by different methods.
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 30: File Complaint Test Case

Attachments:



Hello, I would like to file a complain...

Share instantly with people nearby. On the other person's device, make sure that Quick Share is turned on in the quick panel.

Figure 74: On clicking File a Complaint

View Navigation Menu

Test Case Name:	View Navigation Menu
Test Case ID	#014
Test Priority	High
Test Executed By	Fyaz Ikram
Date of Test Execution	21/01/2022
Description/Summary of Test	The test allows the user to view the navigation menu.
Pre-Condition	User needs to have the app downloaded and installed. User needs to have already created an account, and currently been logged into the account.
Test Steps	1. The user clicks the menu button from the top left-hand side of the app. 2. The system responds by opening the navigation menu with a slider.
Expected Results	The user can view the navigation menu.
Post-Condition	App should stay on the navigation menu until the user clicks out of the navigation menu.
Status (Fail/Pass)	Pass
Notes/Comments/Questions:	N/A
Requirements	N/A
Automation? (Yes/No)	No

Table 31: View Navigation Menu Test Case

Attachments:

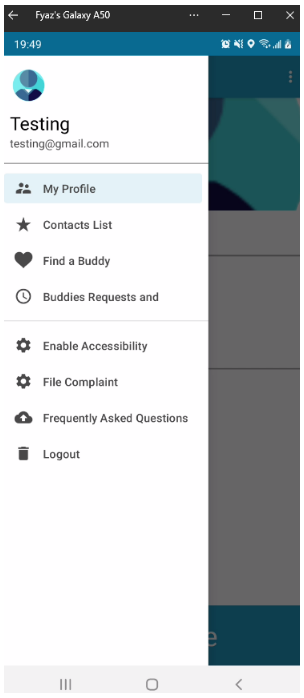


Figure 75: On clicking Nav Menu