CHAPTER 1

INTRODUCTION

1.1 Introduction

Messaging applications or chat applications are instant messaging clients that allow messages to be sent and received between mobile devices or a computer. Text, photos, videos, and voice messages are used by the user to communicate with each other frequently.

Auto blocking and history deletion messaging app aims to deliver a new experience for the user. The app will be focusing on blocking the notification of the incoming messages and removing of older messages automatically. The purpose of creating this app is to help users to manage their own messages in a convenient way. Therefore, it will help user to spend less time to clear the trash of the phone, and it will help user to save a lot of times for other activities. It will also help the user that didn’t have the habit of removing messages frequently to free up more storage of the phone.

1.2 Background of the problem

Auto blocking and history deletion messaging app is a mobile application to block the notification of incoming messages automatically and delete messages by following the schedule by the user. There are a few messaging apps in the market, but the auto blocking and message deletion features is implemented with limited option for the user.

Auto blocking features in WhatsApp, Messenger, and WeChat have limited option for the user to preset the auto blocking period. User has limited choice will lead to inconvenience when using the application. Majority of the users don’t care much about history deletion of their messaging app. It will consume up plenty of time to delete the message when the phone pop-out insufficient of storage alert to inform user to free up the space of the phone in order to reserve storage for incoming messages.

1.3 Problem Statement

- Messages that keep in local storage will use up the spaces of the internal memory of a smartphone after a long-term usage of messaging apps. It will cause insufficient of phone storage and degrade the performance of the phone.
- Clearing or deleting “tonnes” of messages manually is consider as a waste of time. It’s a tedious process for a user to keep track and manage their history of messages manually.
- Users have difficulties in searching for important messages by scrolling up and down the screen and check it one by one.
- The auto blocking function is not highly customizable by the users currently. There are also limited ~~of~~ options for users to block incoming messages automatically based on their preferences.

1.4 Project Objectives

The objective of the project is to develop a messaging app that aims to

- prevent excessive storage usage by deleting messages automatically in order to free up the internal storage of the phone.
- save user’s time spent in deleting messages manually by providing customizable options to perform the deletion of messages automatically.
- ease the user to search for important messages by deleting unwanted messages automatically.
- stop disruption from certain people by blocking notification of incoming messages based on the user’s customizations.

1.5 Proposed Approach

The methodology that is going to implement in this project is Lean Methodology. The core idea of this methodology is to maximize in user value while minimizing waste. In other words, we can say that this methodology is creating more value and benefit for user with lesser resources. In this project, the Build-Measure-Learn principle (one of the central principle of Lean methodology) will be used. The development cycle of the methodology is a Build-Measure-Learn loop in order to improve the quality and requirement of the product.

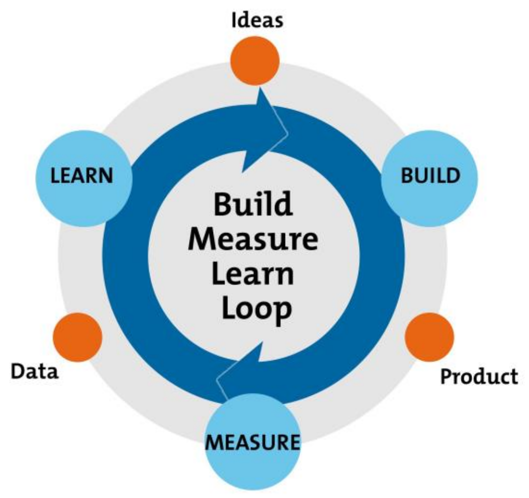


Figure 1.5.1 The development cycle of Lean Methodology
(Diagram adapted from Ries, E. (2011) ',' New York: Crown Business.)

Before the Build-Measure-Learn cycle, the project is better to start with a planning stage. First, develop a hypothesis that will happen during the project. The hypothesis can focus on the user ideas and the features of the product. After that, in the "Build" phase, create a Minimum Viable Product(MVP), it is a product that can work to fulfil the basic requirements to test out the hypothesis that create during the planning stage. During the "Measure" phase, the result in the "Build" phase is being analyse whether the idea is sufficient enough to continue the development of the product. It is good to question that whether the result is meeting the requirement of the product.

Lastly, the decision making will happen in the "Learn" phase. There are 2 ways to forward, it is "Persevere" and "Pivot". We choose "Persevere" when the hypothesis is correct and repeat the development cycle loop to improve and refine the idea. If "Pivot" is chosen, that means the hypothesis is proven to wrong, but we can gain valuable knowledge from the mistakes. Therefore, the loop is going to reset with using the knowledge that we learnt in the previous hypothesis.

Summary of the methodology:

Step 1: Planning of the project followed by a formal hypothesis

Step 2: Build a Minimum Viable Product (MVP) and test it.

Step 3: Analyse the result against the hypothesis to decide whether the product can be continued to develop to fulfil the business needs.

Step 4: Learn from the result on step 3, and decide whether to persevere or pivot.

1.6 Scope of the Project

The auto blocking and history deletion messaging app will include the basic features of a messaging app, i.e. the function of send and receive messages includes images and short videos. The application will provide a login interface for users to have their own identity. Moreover, the function of blocking the notification of incoming messages and deleting the previous messages will be implemented in the application. The application is able to mark important messages and customize messages deletion automatically. The messages being marked will be exempted from deleting. Furthermore, the app will have a filter to display important messages only. Backup and restore chat history function will also be implemented.

CHAPTER 2

LITERATURE REVIEW

2.1 Comparison of Texting, Messaging, and Online Chatting

The term of texting, messaging, and chatting has the meaning of two-way communication between 2 individuals or more. But, the three communication options have their own features and characteristics to best fit the requirement. The aim of the project is to develop a messaging app with certain features. Therefore, a clear picture to term of "Messaging" is important. The table below provides the comparison of the three types of communication.

	Texting (SMS)	**Messaging**	**Online Chatting**
Methods of obtaining the service	Pre-installed on the mobile phone before the release of smartphone. Available in mobile app or web-based after the smartphone was created	Comes with mobile app or web-based	Available in a form of website or a plugin
Example of usage in real world	Sending text messages via cellular network with the mobile service provider without Internet Service	Messaging apps like WhatsApp, Facebook Messenger, and WeChat that can get instant reply with mobile data or WiFi service.	Customer service of a website
Limitation of message size	160 characters	Unlimited	Unlimited
Content support	Text	Text, Image, Video	Text
Platform support	Pre-installed in phone by phone manufacturer before the release smartphone . Cross-platform in the form of mobile application in the app market of smartphone	Cross-platform (mobile phone, web browser)	Web browser

Table 2.1.1 Comparison of Texting, Messaging and Online Chatting

2.2 Push Notification

Nowadays, push notification is an important feature in an app to engage the app's users. It keeps users to get the latest information and notifications by the apps with running in the background of the phone. Push notification ensures that users able to read messages by pop-out notification while the app is running in background or is suspended. Majority of Android devices are based on Google Cloud Messaging (GCM).

GCM is a service by Google company that transmits information from a server to Android devices which connected to Google Play Service. GCM can carry small size of messages with data payloads not exceeding 4KB (Abrosimova, n.d.). Figure 2.2.1 indicates how mobile devices get push notification.

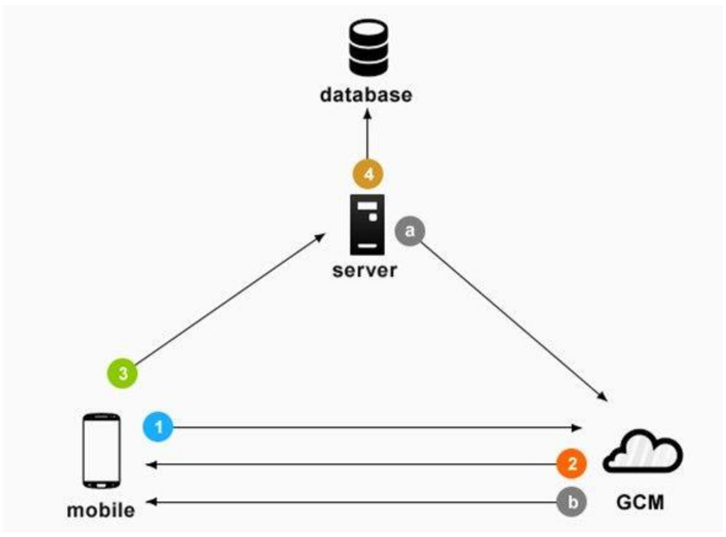


Figure 2.2.1 The process of Android devices receive push notification from GCM Cloud Connection Server

(Abrosimova, n.d.)

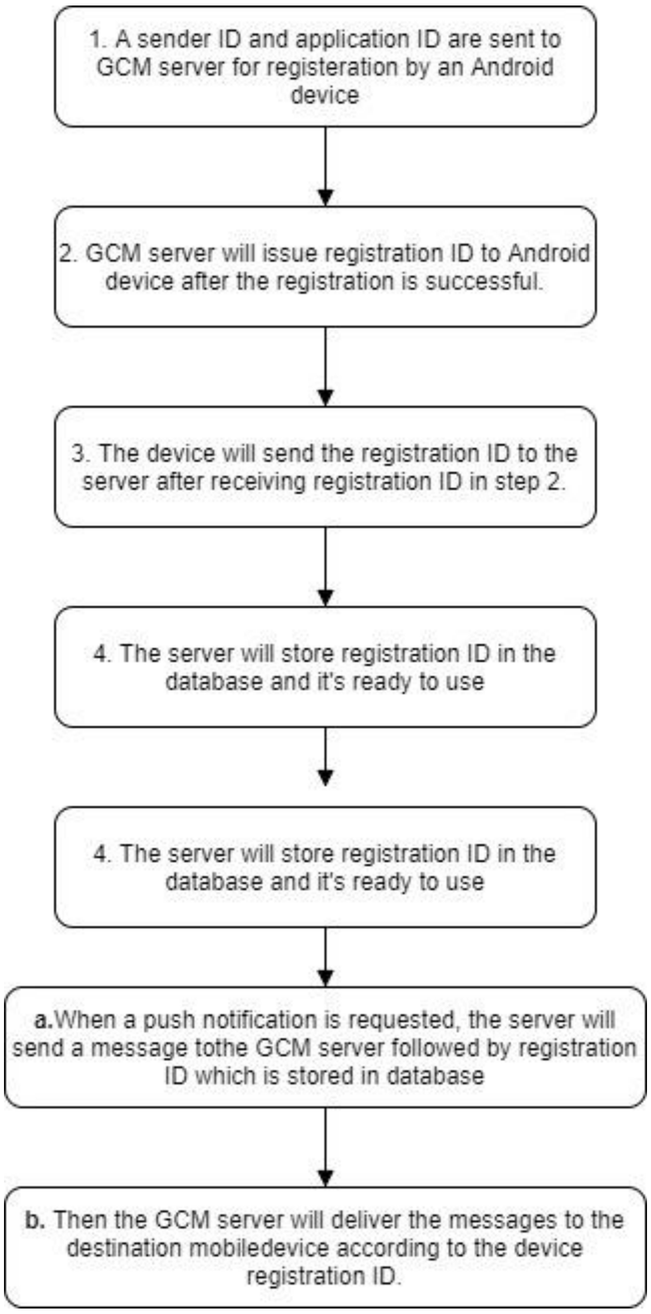

1. A sender ID and application ID are sent to GCM server for registeration by an Android device
2. GCM server will issue registration ID to Android device after the registration is successful.
3. The device will send the registration ID to the server after receiving registration ID in step 2.
4. The server will store registration ID in the database and it's ready to use
4. The server will store registration ID in the database and it's ready to use
a.When a push notification is requested, the server will send a message tothe GCM server followed by registration ID which is stored in database
b. Then the GCM server will deliver the messages to the destination mobiledevice according to the device registration ID.

Figure 2.2.2 Process of push notification of Android Devices

2.3 Java and Kotlin Programming Language

Having a basic of Java programming language knowledge is the prerequisite for a software developer to work with Android Studio. With using Java, Android Studio is able to create interactive UIs and design simple views for each activity of the mobile app.

2.3.1 Comparison between Java and Kotlin

Java and Kotlin are the current programming language that use to develop Android application. Each of the programming language has its own priority for any business. Table 2.3.1.1 is a comparison between Java and Kotlin.

	Java	**Kotlin**
Author	James Gosling	Andrey Breslav
Type	Object-Oriented	Object-Oriented
Learning curve	Low	Low
Code Size	Have to write more code than Kotlin	30-40% less code as compared to Java
Rendering	Client Side	Client Side
App Size	Have less size as compare to Kotlin. Gradle build time for Java is faster as compared to Kotlin	Used up more space as because contains of Kotlin as well as Java libraries.
Checked Exception	Yes	No
Use of semicolon	It is a must to write semicolon to terminate a line of code.	It is optional to write semicolon in the end of statements.

Table 2.3.1.1 Comparison between Java and Kotlin

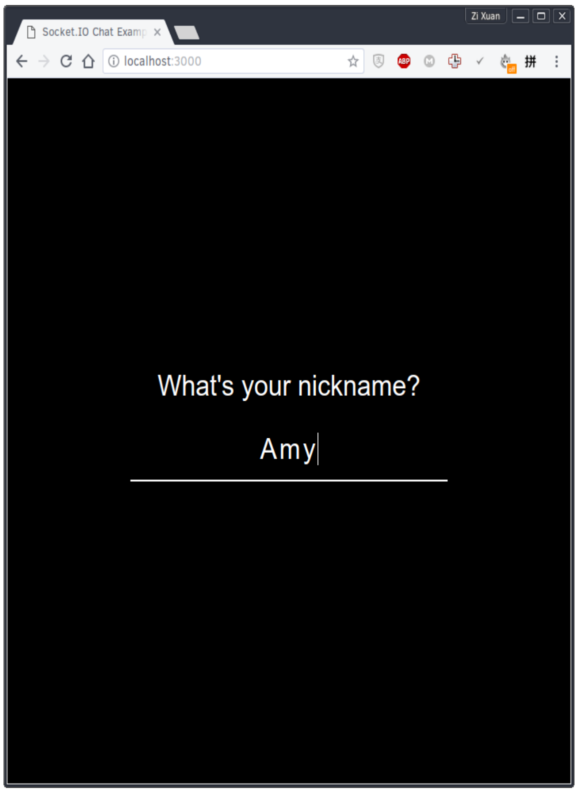
2.3.2 Advantages and Disadvantages of Java and Kotlin

Programming Language	Advantages	Disadvantages
Java	• Faster Gradle build time • More open source libraries and frameworks • More reliable and still have updates since 1995 until now	• It is not null safe by default • Consume more memory space • Vulnerable to security threats
Kotlin	• It is null safe by default • Easy to learn • Easy to adopt from Java	• Slower Gradle build time • No checked exception • The need for Java interoperability has forced some limitations

Table 2.3.2.1 Advantages and Disadvantages of Java and Kotlin

2.4 Messaging app with React Native and Socket.io

Socket.io is a framework that creates network socket for the server and client. It’s a real-time engine that supports instant messaging and chat with just a few lines of code. It works fine with the React Native. Socket.io supports communication between single client and multi clients once the socket is established and the users can speak freely without delay and it is faster that a HTTP request. Socket.io can create a chat group for a certain number of users in the same port. Figure 2.4.1 is an example of chat room that requires user nickname to enter a chat room. Figure 2.4.2 is an example of conversation between 2 users.



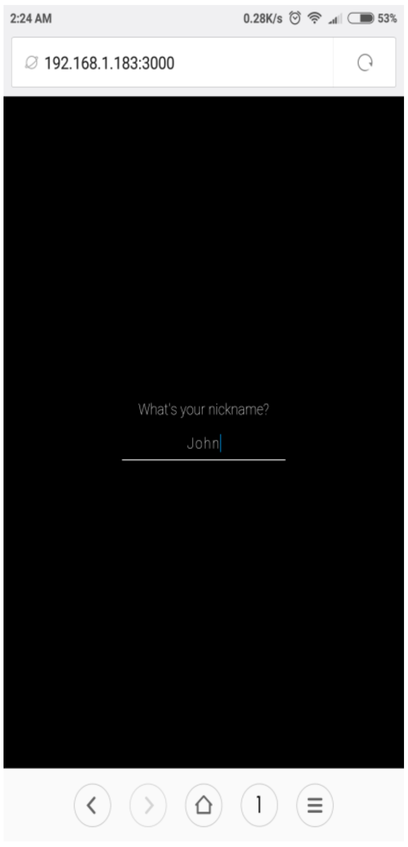
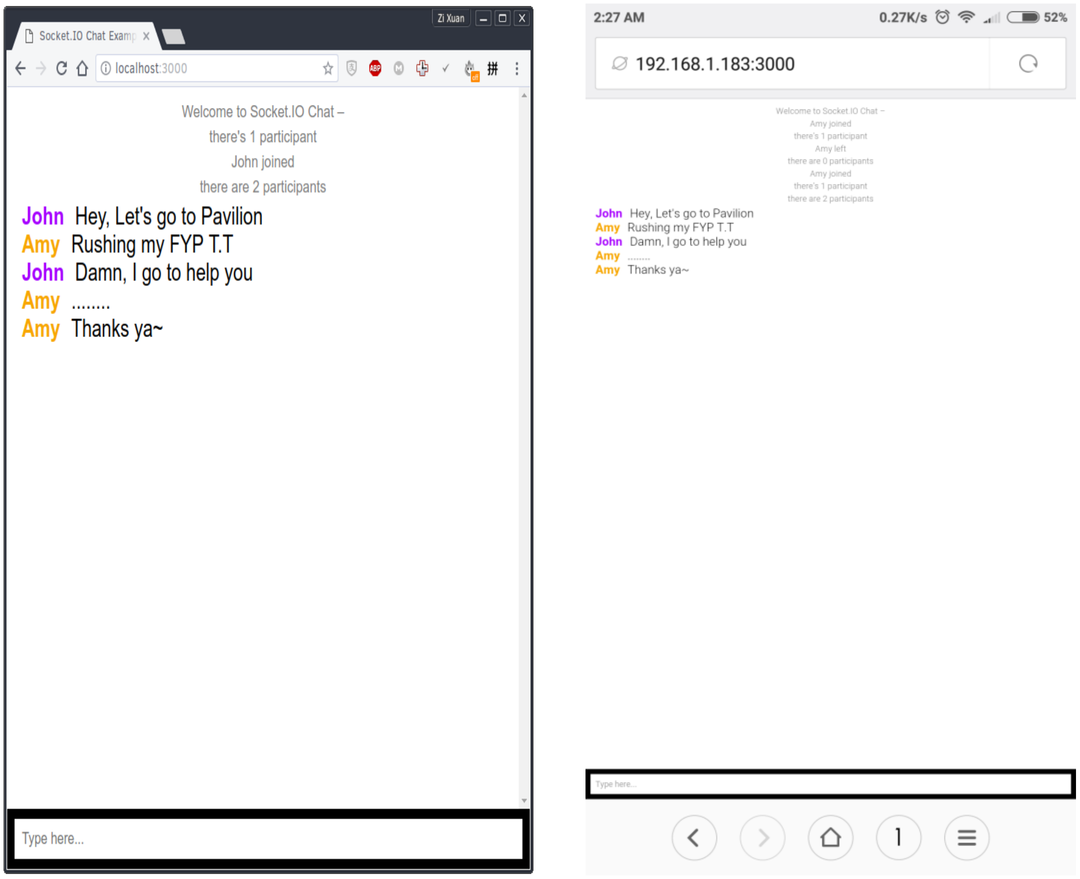


Figure 2.4.1 Example of Chat Room using Socket IO (Enter Nickname)



Welcome to Socket.IO Chat –
there's 1 participant
John joined
there are 2 participants

John Hey, Let's go to Pavilion
Amy Rushing my FYP T.T
John Damn, I go to help you
Amy ....
Amy Thanks ya~

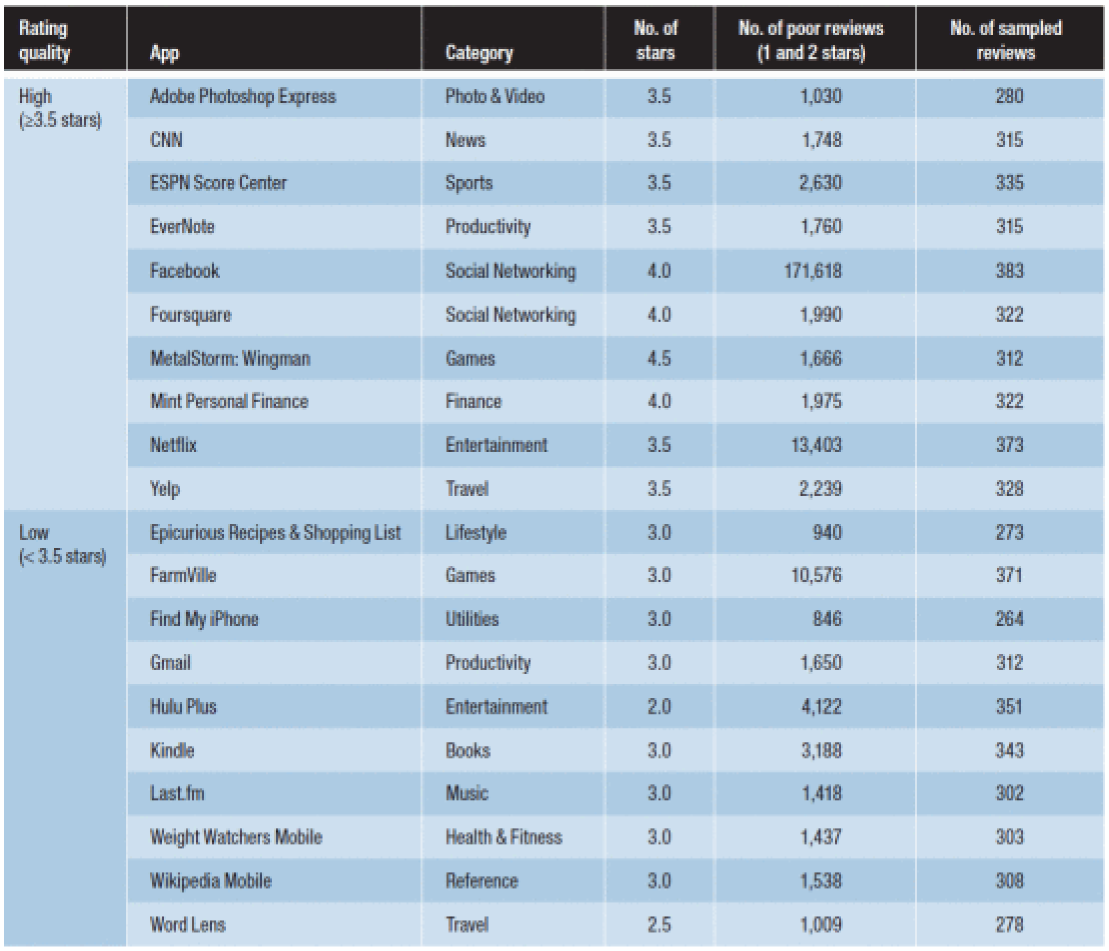
Welcome to Socket.IO Chat –
Amy joined
there's 1 participant
Amy left
there are 0 participants
Amy joined
there's 1 participant
there are 2 participants

John Hey, Let's go to Pavilion
Amy Rushing my FYP T.T
John Damn, I go to help you
Amy ....
Amy Thanks ya~

Figure 2.4.2 Example of Chat Room using Socket IO (Chatting)

2.5 What do mobile app users complain?

The quality of app has been the main concern for the app company currently. A research is carried out to study the feedback from users of 20 iOS apps and Figure 2.5.1 shows the statistics of the studies. The 20 iOS apps are from different category.

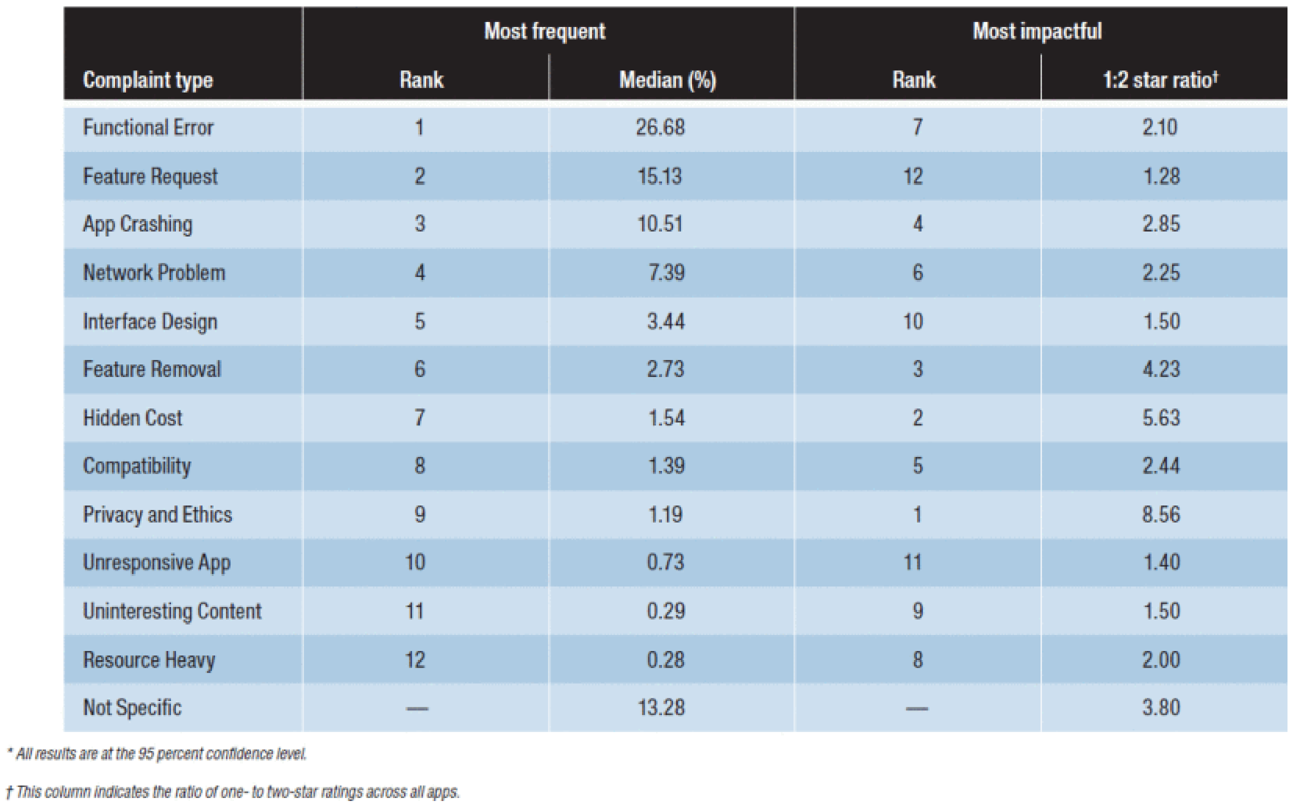


Rating quality	App	Category	No. of stars	No. of poor reviews (1 and 2 stars)	No. of sampled reviews
High (≥3.5 stars)	Adobe Photoshop Express	Photo & Video	3.5	1,030	280
	CNN	News	3.5	1,748	315
	ESPN Score Center	Sports	3.5	2,630	335
	EverNote	Productivity	3.5	1,760	315
	Facebook	Social Networking	4.0	171,618	383
	Foursquare	Social Networking	4.0	1,990	322
	MetalStorm: Wingman	Games	4.5	1,666	312
	Mint Personal Finance	Finance	4.0	1,975	322
	Netflix	Entertainment	3.5	13,403	373
	Yelp	Travel	3.5	2,239	328
Low (< 3.5 stars)	Epicurious Recipes & Shopping List	Lifestyle	3.0	940	273
	FarmVille	Games	3.0	10,576	371
	Find My iPhone	Utilities	3.0	846	264
	Gmail	Productivity	3.0	1,650	312
	Hulu Plus	Entertainment	2.0	4,122	351
	Kindle	Books	3.0	3,188	343
	Last.fm	Music	3.0	1,418	302
	Weight Watchers Mobile	Health & Fitness	3.0	1,437	303
	Wikipedia Mobile	Reference	3.0	1,538	308
	Word Lens	Travel	2.5	1,009	278

Figure 2.5.1 20 iOS app with different category

(Khalid, Sahihab, Nagappan and Hassan, 2014)

According to the research, there are 12 types of most frequent complaint by the mobile app users. The first three columns of Figure 2.5.2 show the type of most complaint issues by the users. Functional Error is ranked on the top column which indicates the most serious problem that encountered by the users. The examples of functional error are location identification issues and authentication problems. The feature request complaint is ranked in number 2. The most requests feature by the users were app specific. But, there are 6.12 percent of the request were for better notification support. (Khalid, Sahihab, Nagappan and Hassan, 2014). App crashing is also one of the critical complaints by the mobile app user which can affect the user experience and it indicates the stability of a mobile apps.



	Most frequent		Most impactful	
Complaint type	Rank	Median (%)	Rank	1:2 star ratio†
Functional Error	1	26.68	7	2.10
Feature Request	2	15.13	12	1.28
App Crashing	3	10.51	4	2.85
Network Problem	4	7.39	6	2.25
Interface Design	5	3.44	10	1.50
Feature Removal	6	2.73	3	4.23
Hidden Cost	7	1.54	2	5.63
Compatibility	8	1.39	5	2.44
Privacy and Ethics	9	1.19	1	8.56
Unresponsive App	10	0.73	11	1.40
Uninteresting Content	11	0.29	9	1.50
Resource Heavy	12	0.28	8	2.00
Not Specific	—	13.28	—	3.80

* *All results are at the 95 percent confidence level.*

† *This column indicates the ratio of one- to two-star ratings across all apps.*

Figure 2.5.2 Category of user complaint

(Khalid, Sahihab, Nagappan and Hassan, 2014)

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Description of the project

The goal of the project is to develop a messaging app which has the auto blocking and history deletion features. The messaging app will also work like a normal messaging app which can send and receive text, image and video messages within 2 or more individuals.

One of the main features of auto blocking in the messaging app is to block incoming notifications and retain messages from the sender. Comparing to other messaging in the app market, there are more blocking options in the app. It will let user to customize the auto blocking period and it can be saved into a profile for future use. Besides that, users are able to customize the history deletion automatically, the messages that deleted automatically will uploaded to the cloud server as a backup. The older messages can be obtained when needed and it will to free up the local storage of the phone.

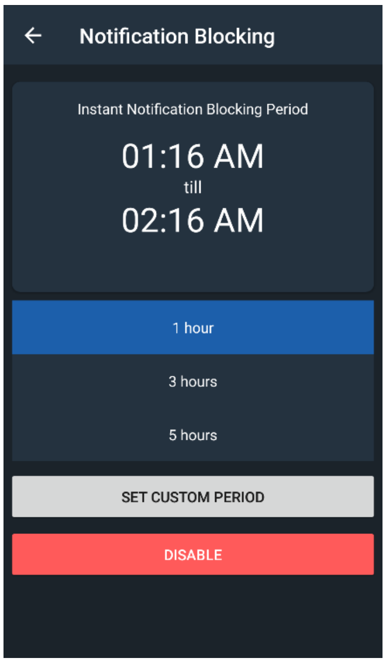


Figure 3.1.1 A prototype of auto blocking profile

Figure 3.1.1 is a prototype of auto blocking profile that can be customized by users. When the switch is turn on, the auto blocking period will start. The notification of the messaging app will auto mute but the messages will still receive by the phone. The auto blocking function will not apply to the whole app, users can tap in the and customize blocking profile and block specific contact.

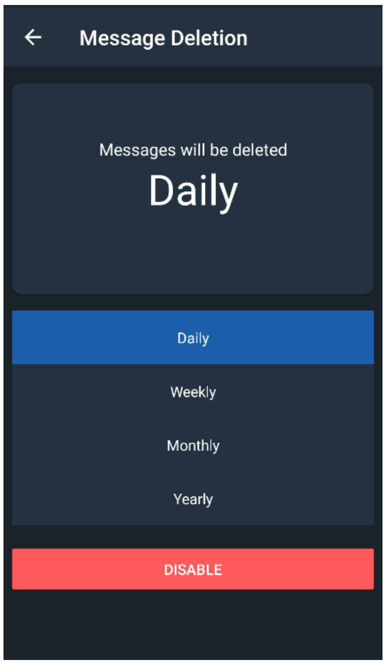


Figure 3.1.2 A prototype of message deletion setting

Figure 3.1.2 is a prototype of message deletion setting that allows users to customize their message deletion options. Recycle bin is a feature to archive deleted messages, users are able to restore deleted messages from recycle bin. Chat History Backup allows users to back up their previous messages to cloud server after the messages was removes permanently. Users can set the date to delete the messages every month or user can pick more specific date and time to delete the message automatically.

3.2 Methodology Adopted

The methodology that adopted in this project is Lean Methodology. The main concern of Lean Methodology is suitable for mobile app development which maximize users' value and minimize the waste of resources. The development cycle of the Lean Methodology is Build, Measure, and Learn. It is an effective way to build a mobile application that fulfil the requirements of users and speed up the development period. This method let the developer learns from the mistakes made during the development and help the developer to improve the product to match the hypothesis that made during the planning phase.

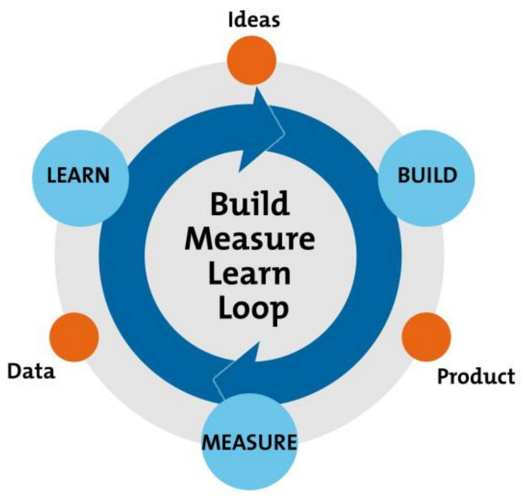


Figure 3.2.1 The development cycle of Lean Methodology

(Diagram adapted from Ries, E. (2011) ',' New York: Crown Business.)

Summary of the Build-Measure-Learn Loop

Planning

A planning of the project is needed before the process of development is going into the Build-Measure-Learn development cycle. The planning of the project must be followed by a formal hypothesis that made by the developer.

A messaging app will be developed in this project, the messaging app must planned with a few of hypothesis such as the messaging is able to send and receive messages with notifications. An auto blocking feature~~s~~ with customization period is needed. Message deletion must be able to be customized by users and could delete messages automatically based on setting.

Build

Build a Minimum Viable Product(MVP) based on the hypothesis that made during the planning phase. It must fulfil the basic requirements of the product before releasing for any testing.

the messaging app in this project must have at least a working function of send and receive messages without any bugs. It must also support text, image, and video. The auto blocking feature must at least can block all incoming notification after the setting is enabled by users. It can at least delete messages automatically after the customization of setting is done. Make sure the basic functions of the app can work fine before going to the next phase of development.

Measure

In this phase, the product is being analyse whether the result of the product is similar or diverge with the hypothesis. If yes, it is suitable to continue to fulfil the users' needs.

The auto blocking feature in the app doesn't work like the hypothesis that made by the developer and it needs to block message manually, which means it failed to achieve the.

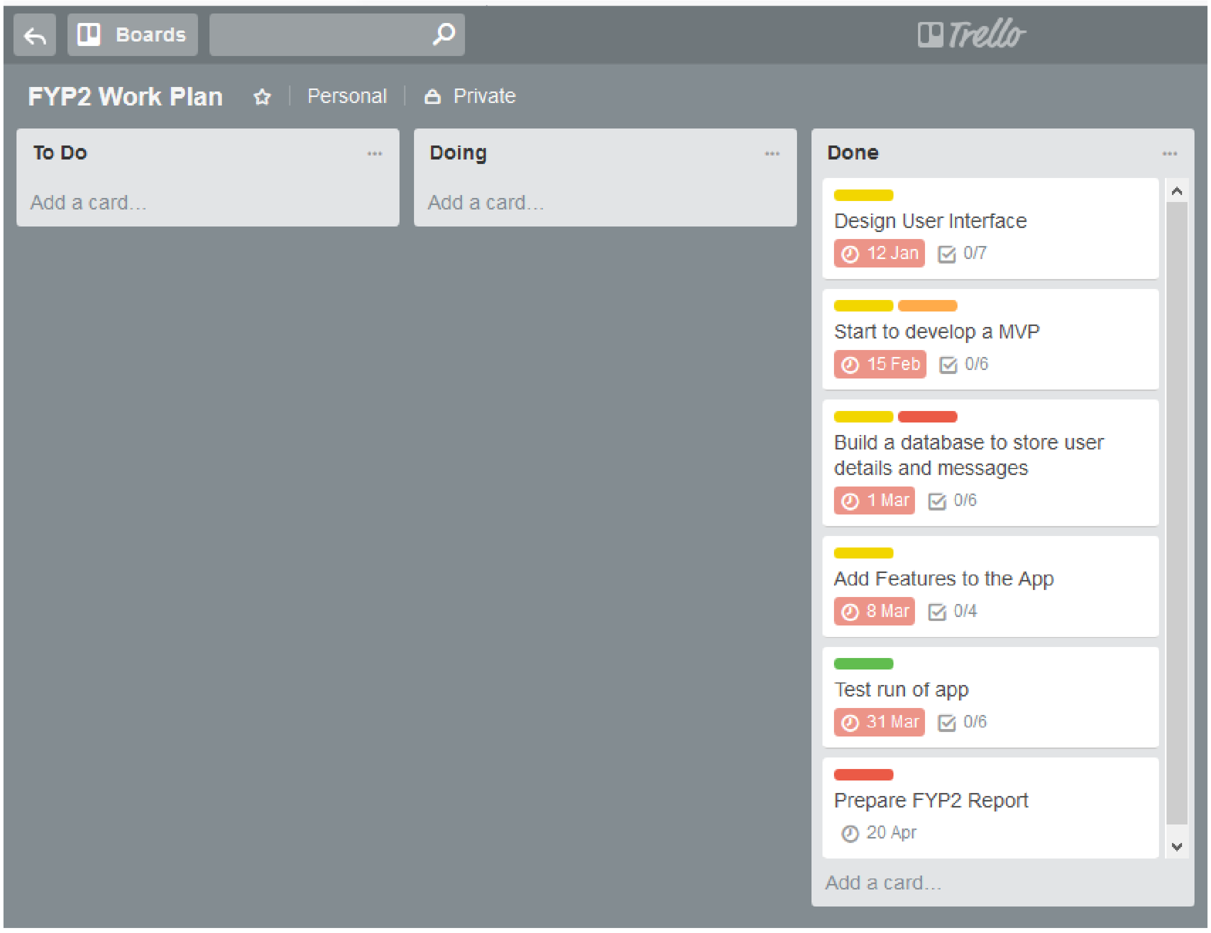
Learn

There are two ways to choose to bring the product to the next level. “Persevere” is chosen when the hypothesis is correct and fits the idea of product. Then, the product can undergo the cycle of Build-Measure-Learn again to improve or add-on some new features to the product. If the product doesn’t fit the hypothesis, “pivot” is chosen, which means the hypothesis is proven to be wrong, but it doesn’t matter, as long as the developer will gain valuable knowledge from the mistake that they made.

When the app basic functions fulfil the hypothesis, we choose “persevere” to go a step further to improve the basic functions to make it more reliable. If the function of the product is far away from expectation, choose “pivot” to find out how to deal with the mistakes or what is the core problem that makes the product failed to achieve the requirements.

3.3 Work Plan

The project work plan is keep track with Trello, which is suitable for a small size or individual project. Figure 3.3.1 is Trello board that lists out the work plan to fulfil in Project 2. The work plan is divided into 3 lists which are To Do, Doing, and Done. The deadline of each task is set to prevent any of the task left out from the project.



Design User Interface
12 Jan 0/7

Start to develop a MVP
15 Feb 0/6

Build a database to store user details and messages
1 Mar 0/6

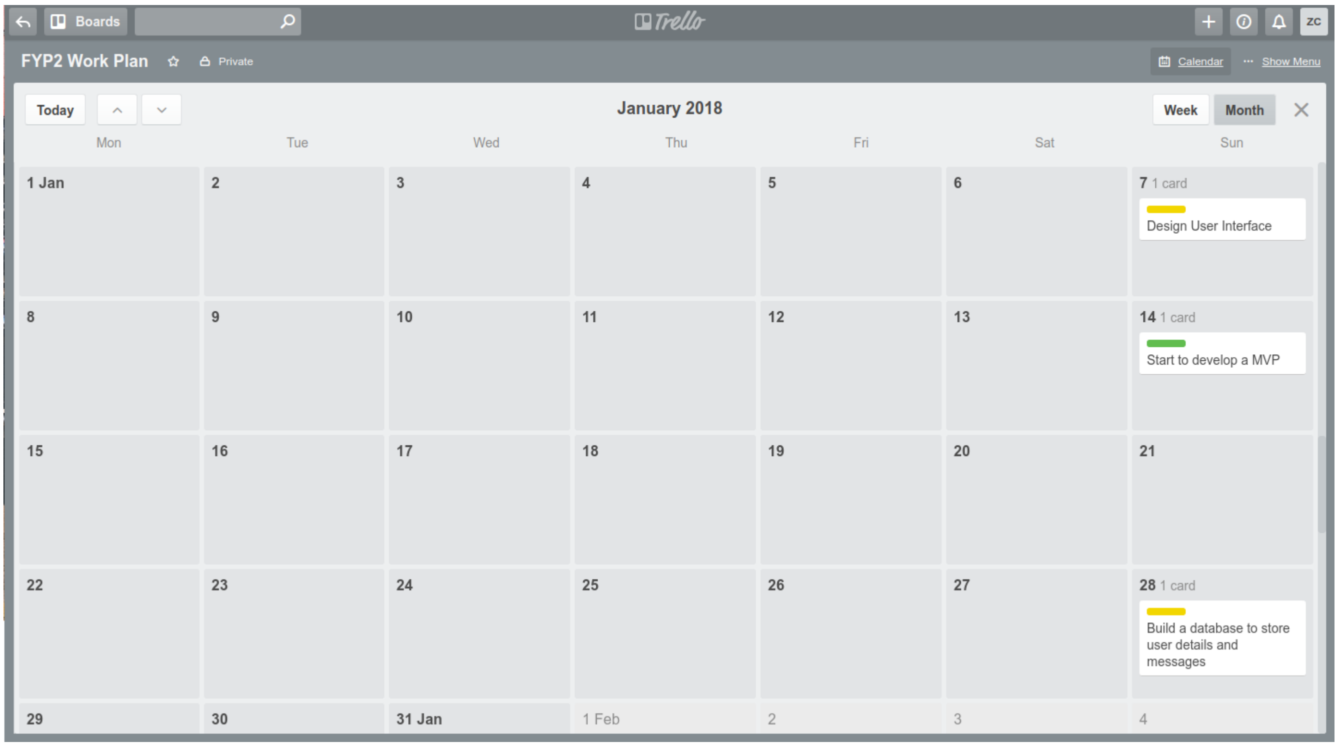
Add Features to the App
8 Mar 0/4

Test run of app
31 Mar 0/6

Prepare FYP2 Report
20 Apr

Figure 3.3.1 Trello Board

Trello Calendar is able to display the deadline of each task clearly Different colour of label indicates different level of importance.

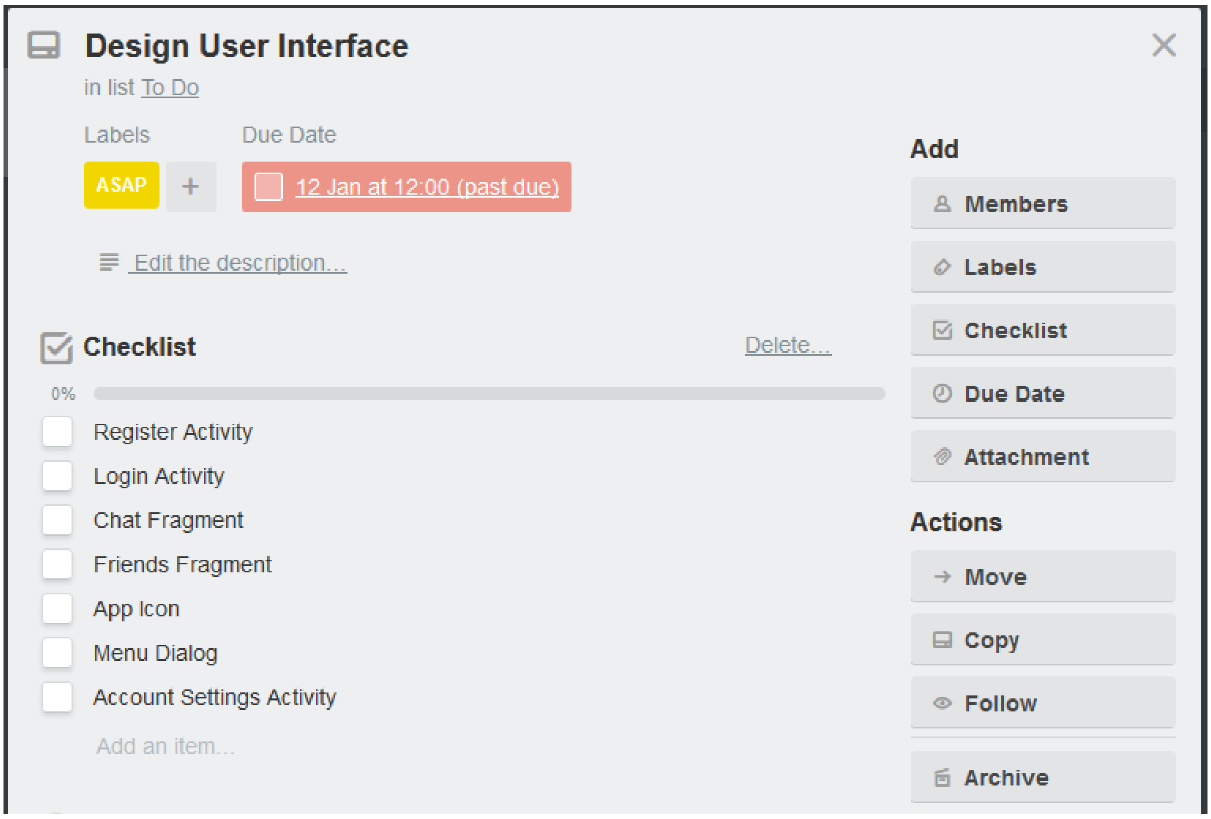


7 1 card — Design User Interface

14 1 card — Start to develop a MVP

28 1 card — Build a database to store user details and messages

Figure 3.3.2 Trello Calendar



Design User Interface

in list To Do

Labels

ASAP

Due Date

12 Jan at 12:00 (past due)

Edit the description…

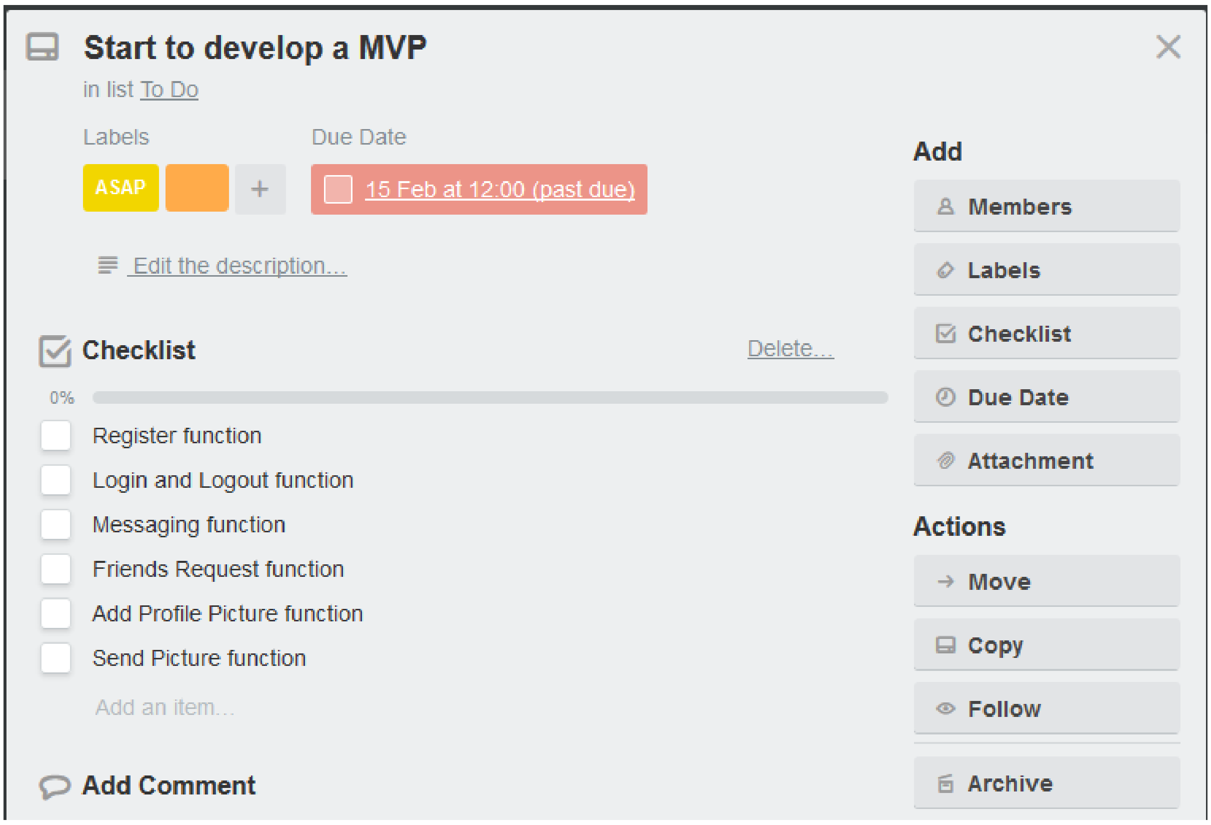
Checklist

0%

- Register Activity
- Login Activity
- Chat Fragment
- Friends Fragment
- App Icon
- Menu Dialog
- Account Settings Activity

Add an item…

Figure 3.3.3 Checklist of User Interface design


Start to develop a MVP

in list To Do

Labels

ASAP

Due Date

15 Feb at 12:00 (past due)

Edit the description…

Checklist

Delete…

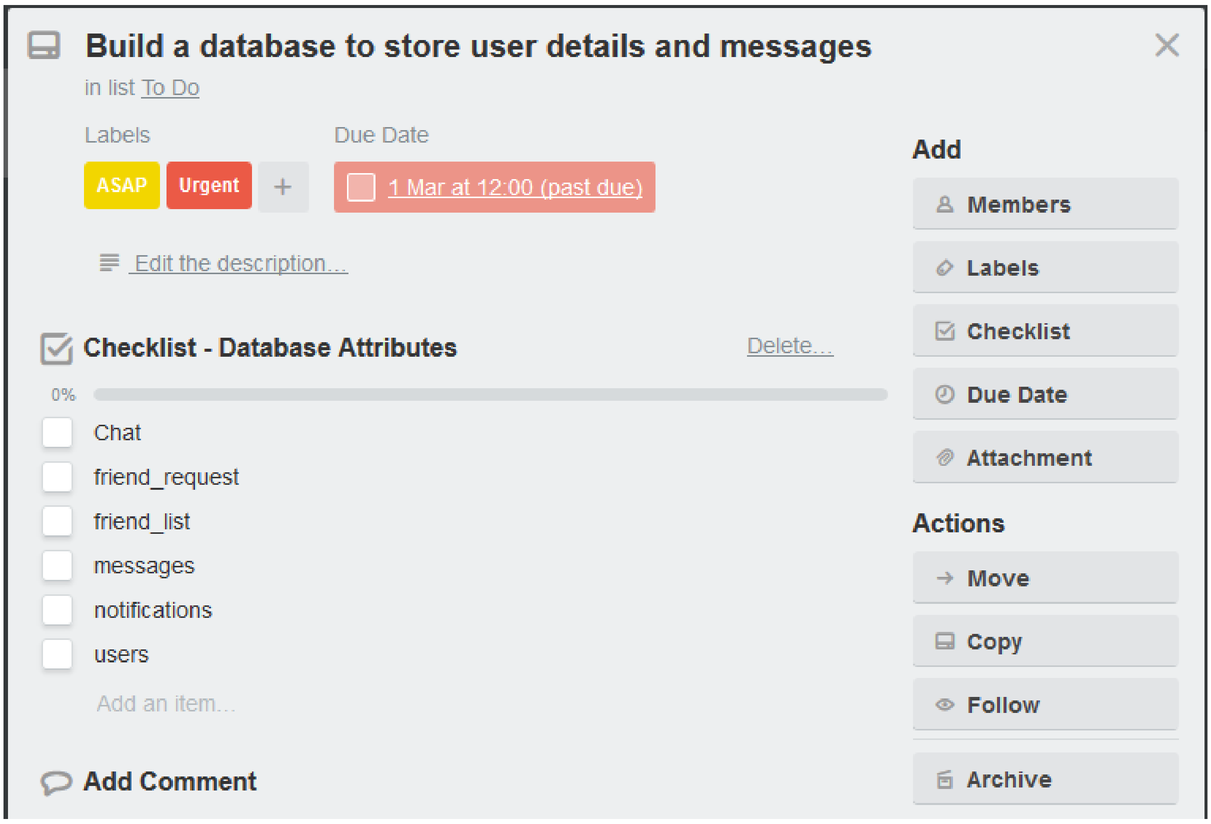
0%

- [ ] Register function
- [ ] Login and Logout function
- [ ] Messaging function
- [ ] Friends Request function
- [ ] Add Profile Picture function
- [ ] Send Picture function

Add an item…

Add Comment

Figure 3.3.4 Checklist of Develop a MVP



Build a database to store user details and messages

in list To Do

Labels: ASAP Urgent

Due Date: 1 Mar at 12:00 (past due)

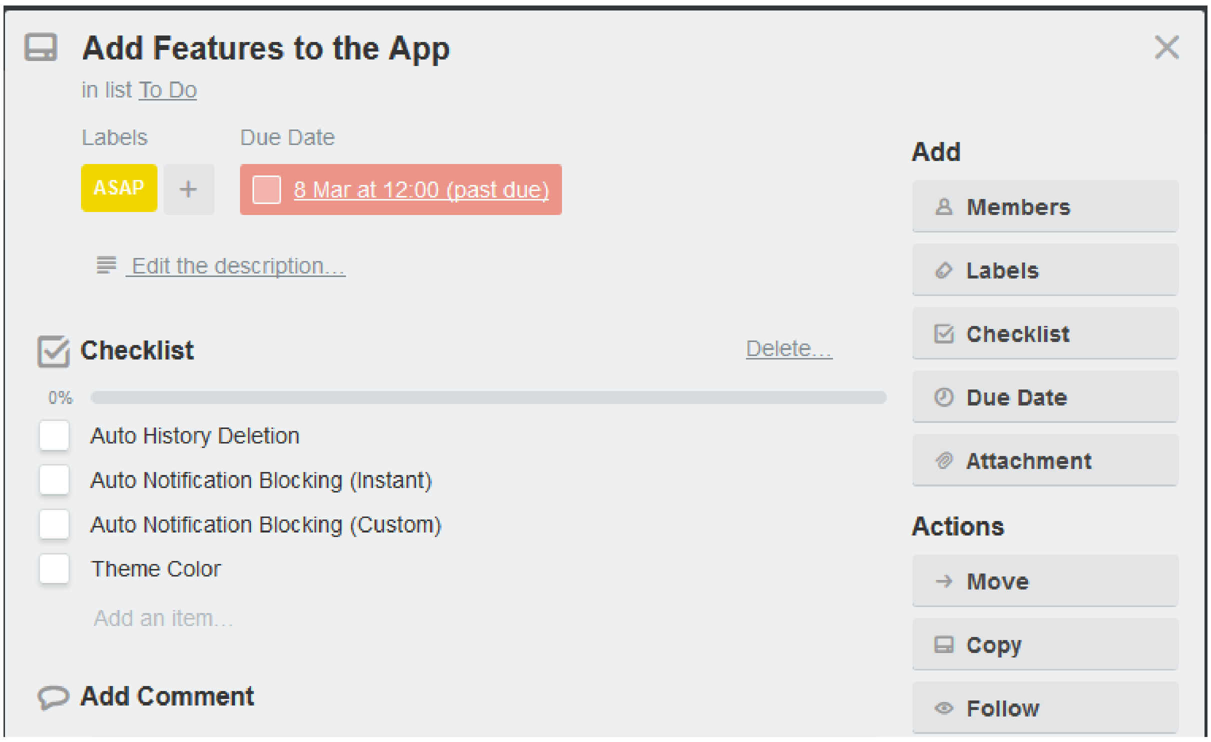
Edit the description…

Checklist - Database Attributes

0%

- [ ] Chat
- [ ] friend_request
- [ ] friend_list
- [ ] messages
- [ ] notifications
- [ ] users

Figure 3.3.5 Checklist of Building a Database



Add Features to the App

in list To Do

Labels: ASAP

Due Date: 8 Mar at 12:00 (past due)

Edit the description…

Checklist

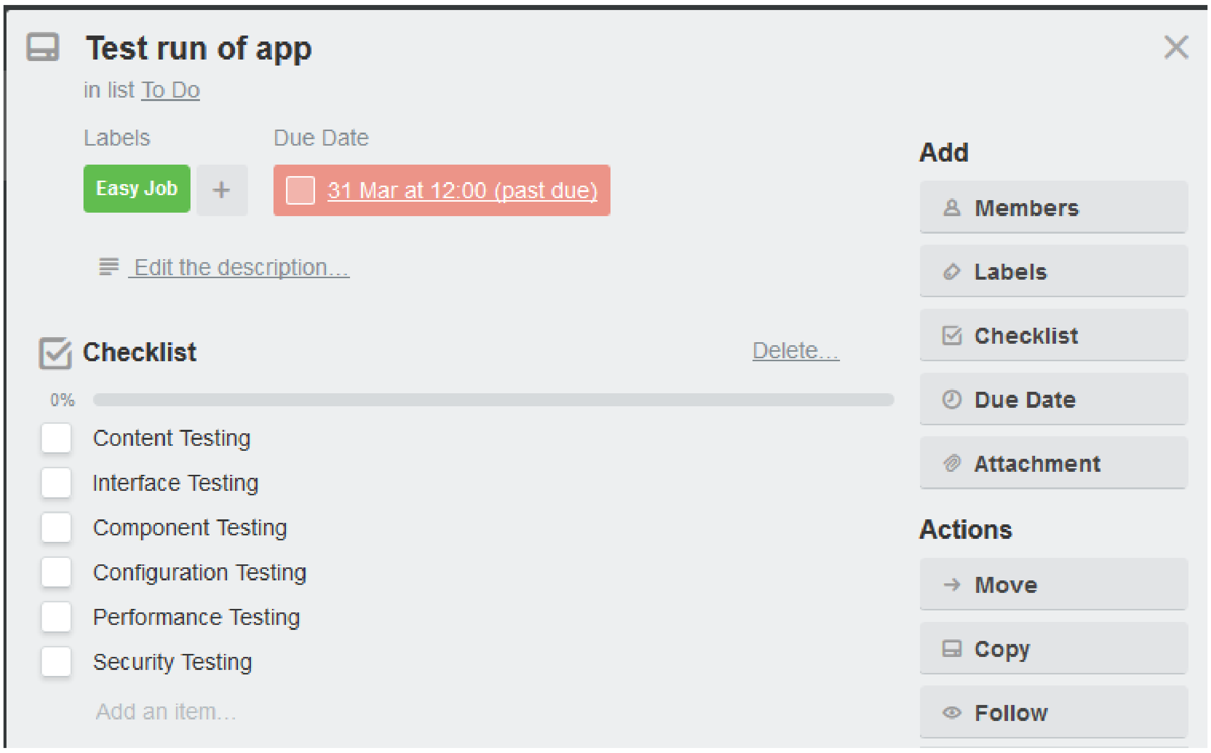
0%

- [ ] Auto History Deletion
- [ ] Auto Notification Blocking (Instant)
- [ ] Auto Notification Blocking (Custom)
- [ ] Theme Color

Add an item…

Add Comment

Figure 3.3.6 Checklist of Add Features



Test run of app

in list To Do

Labels: Easy Job

Due Date: 31 Mar at 12:00 (past due)

Edit the description…

Checklist

Delete…

0%

- [ ] Content Testing
- [ ] Interface Testing
- [ ] Component Testing
- [ ] Configuration Testing
- [ ] Performance Testing
- [ ] Security Testing

Add an item…

Figure 3.3.7 Checklist of Test Run of App

3.4 Development tools

Android Studio

It is used to develop the front-end of the mobile app, and it could also install plugin like Firebase and Crashlytics to for database implementation and bug reporting respectively.

Sublime

A text editor that can support the develop of React JS syntax and can code faster with the auto-complete function of the text editor. It has a nice folder and file management for a project. Using the plugin of Babel JS, the syntax of React JS can be shown clearly in its own colour tone.

MockingBot

It is a prototype tools that is easy to use to sketch the user interface of the mobile app. It includes the elements of iOS and Android which can produce a working prototype. The prototype can save and test run on a web browser on PC or a phone.

BitBucket

A version control repository that support Git. It can store project in private state for free which can access by the author only. Bitbucket can support up to 5 members in a team and manage the role of the 5 member to read, write or read and write permission of the repository.

Android Virtual Device

To test run the result of the app development before shipping out the workable mobile application. The Android Virtual Device comes with Android 6.0 Marshmallow which can support the development of React Native.

Terminal

To view the error when the app crash during the development with using React Native. It is mandatory to use it to run React Native. Terminal is available in macOS or Linux, it’s a powerful tool to manage a computer without using the Graphical User Interface.

Firebase
A NoSQL database that provides by Google and it is performed well with Android Studio. In this project, it is used as a real-time database for the messaging purpose. Besides that, it also able to store user’s data.

Crashlytics

It is one of the product from Fabric to monitor app crashing of the mobile app on a device. It can generate a bug report to the developer when there is an app crash from the client side. In addition, it can also monitor the growth of the app.

CHAPTER 4

PROJECT INITIAL SPECIFICATION

4.1 Use Case Diagram

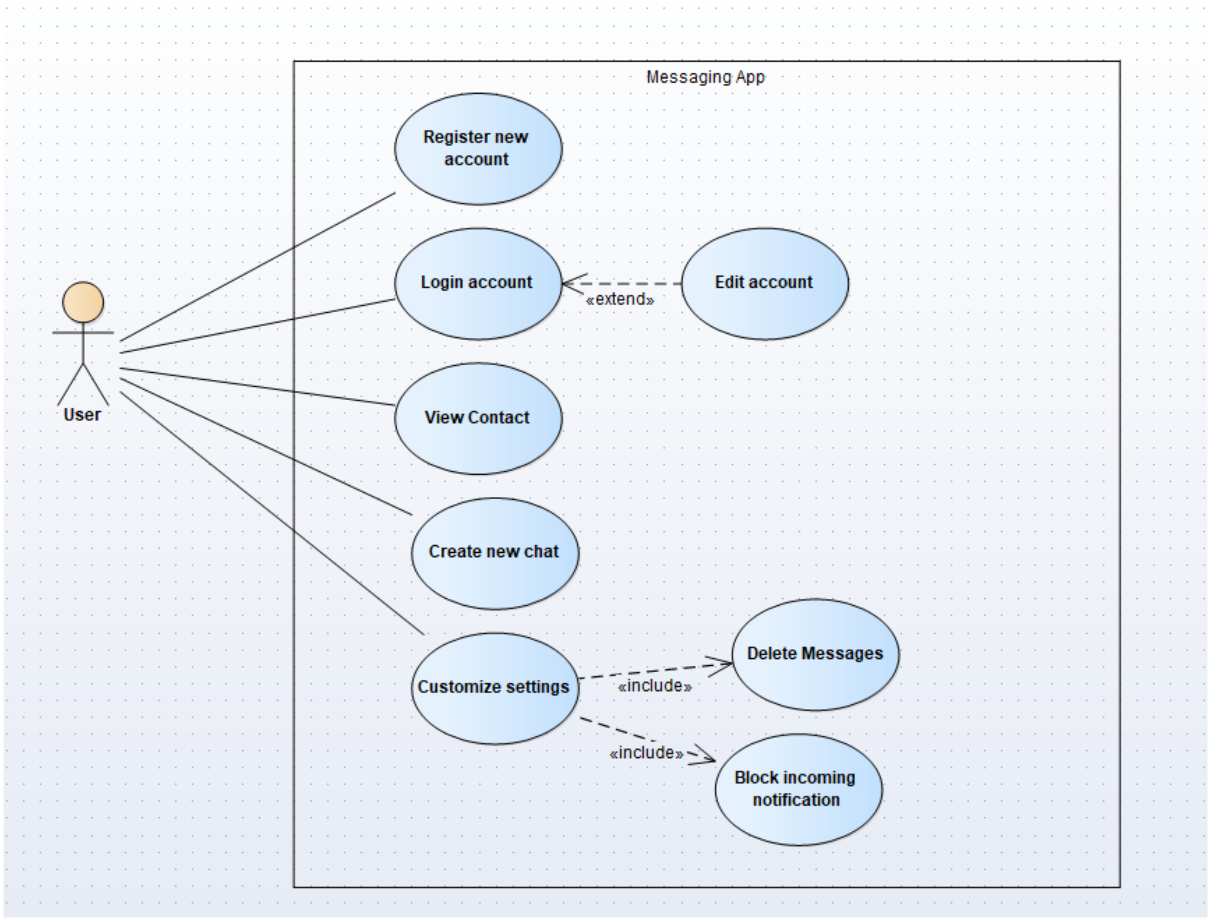
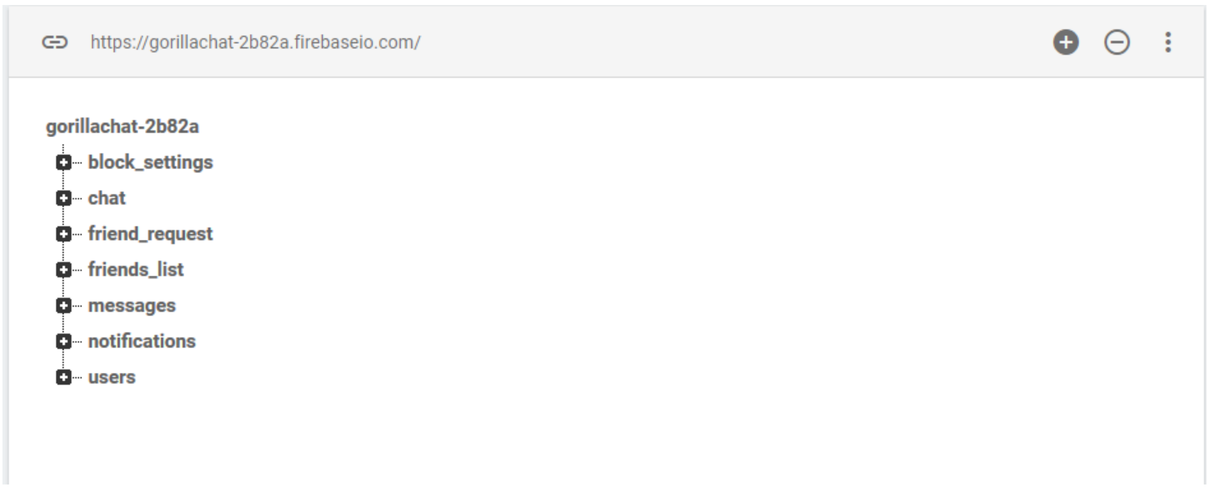


Figure 4.1.1 Use-Case Diagram for messaging app

4.1.1 Use-Case Diagram Description

User is the person that uses the messaging app and the use-case indicates the actions that can be done by the user. New user must register an account before login to the messaging app. After login to the messaging app, the user will be able to edit account information such as name, age, birth date, etc. User can view contacts that they added. User can create new chat include group chat. User can customize setting like auto blocking incoming notification and history deletion.

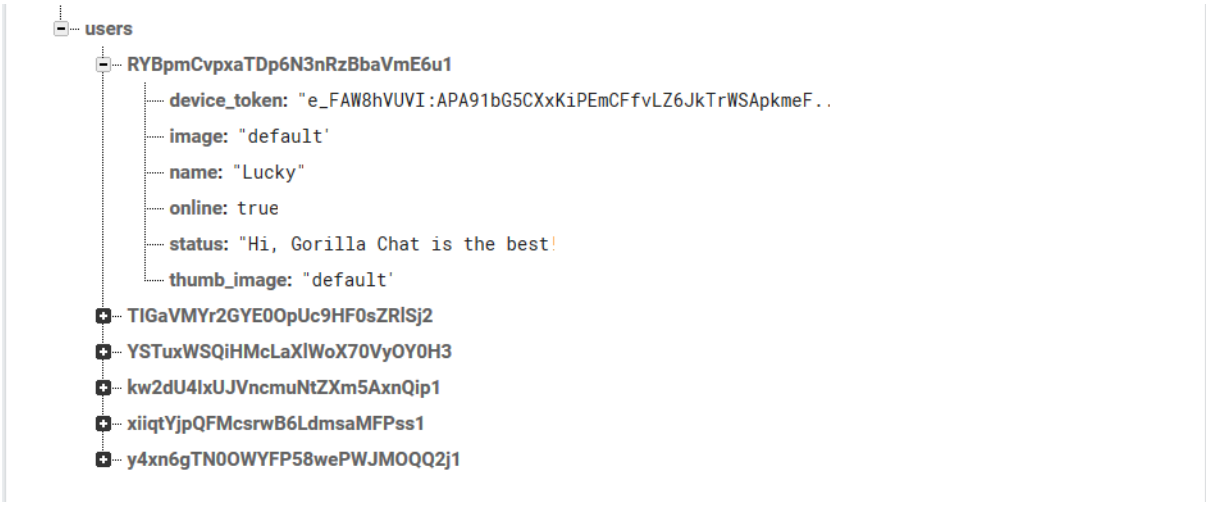
4.2 Database Structure



```
https://gorillachat-2b82a.firebaseio.com/

gorillachat-2b82a
  block_settings
  chat
  friend_request
  friends_list
  messages
  notifications
  users
```

Figure 4.2.2 Database structure from Firebase

Figure 4.2.2 is a database structure of the messaging app, it is store in NoSQL database model and it is perform well for real-time database service. “gorillachat-2b82a” is the parents of the block_settings, chat, friend_request, friends_list, messages, notifications, and users.

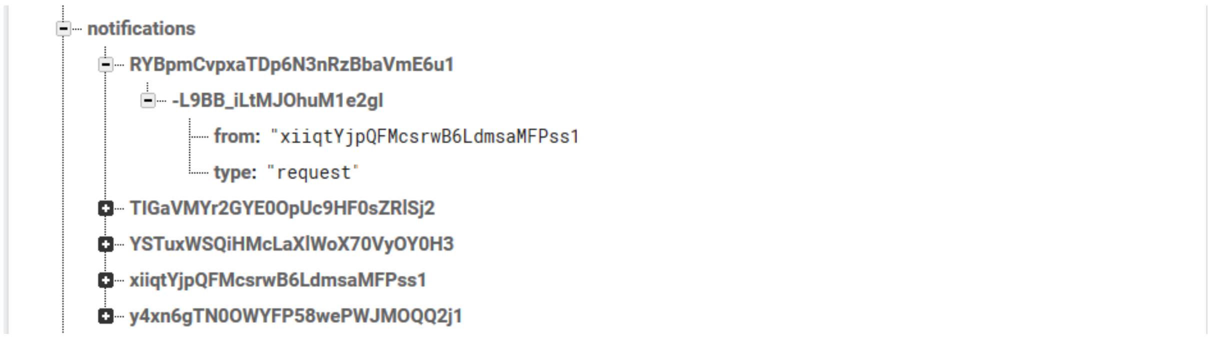


```
users
  RYBpmCvpxaTDp6N3nRzBbaVmE6u1
    device_token: "e_FAW8hVUVI:APA91bG5CXxKiPEmCFfvLZ6JkTrWSApkmeF..
    image: "default'
    name: "Lucky"
    online: true
    status: "Hi, Gorilla Chat is the best!
    thumb_image: "default'
  TIGaVMYr2GYE0OpUc9HF0sZRlSj2
  YSTuxWSQiHMcLaXlWoX70VyOY0H3
  kw2dU4IxUJVncmuNtZXm5AxnQip1
  xiiqtYjpQFMcsrwB6LdmsaMFPss1
  y4xn6gTN0OWYFP58wePWJMOQQ2j1
```

Figure 4.2.3 users’ database in Firebase

The structure of users’ database

```
|-users
        |-user_id
                |- device_token
                |- image
                |- name
                |- online
                |- status
                |- thumb_image
```

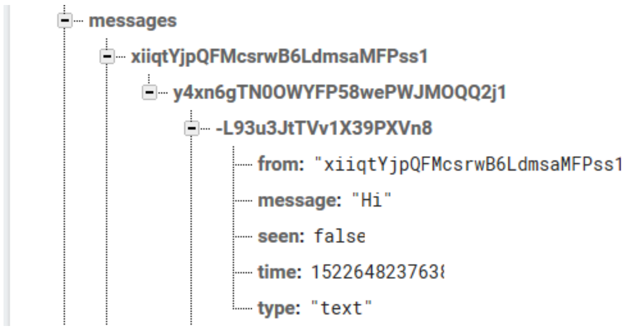


```
notifications
  RYBpmCvpxaTDp6N3nRzBbaVmE6u1
    -L9BB_iLtMJOhuM1e2gl
      from: "xiiqtYjpQFMcsrwB6LdmsaMFPss1
      type: "request"
  TIGaVMYr2GYE0OpUc9HF0sZRlSj2
  YSTuxWSQiHMcLaXlWoX70VyOY0H3
  xiiqtYjpQFMcsrwB6LdmsaMFPss1
  y4xn6gTN0OWYFP58wePWJMOQQ2j1
```

Figure 4.2.4 notifications' database in Firebase

The structure of notifications' database

```
|-notificataios
        |- sender_id
                |- receiver_id
                        |- from
                        |- type
```

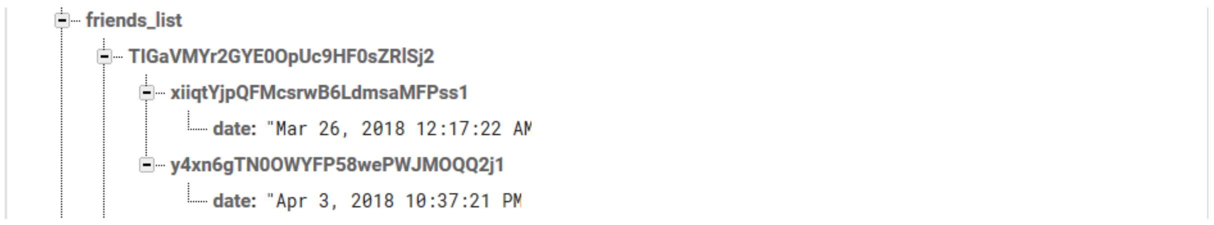


```
messages
  xiiqtYjpQFMcsrwB6LdmsaMFPss1
    y4xn6gTN0OWYFP58wePWJMOQQ2j1
      -L93u3JtTVv1X39PXVn8
        from: "xiiqtYjpQFMcsrwB6LdmsaMFPss1
        message: "Hi"
        seen: false
        time: 152264823763
        type: "text"
```

Figure 4.2.5 messages' database in Firebase

The structure of notifications' database

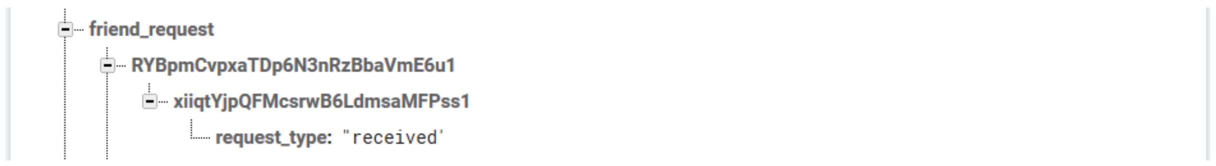
```
|-notificataios
        |- sender_id
                |- receiver_id
                        |- push_id
                                |- from
                                |- message
                                |- seen
                                |- time
                                |- type
```



```
friends_list
  TIGaVMYr2GYE0OpUc9HF0sZRlSj2
    xiiqtYjpQFMcsrwB6LdmsaMFPss1
      date: "Mar 26, 2018 12:17:22 AM
    y4xn6gTN0OWYFP58wePWJMOQQ2j1
      date: "Apr 3, 2018 10:37:21 PM
```

Figure 4.2.6 friends_list's database in Firebase

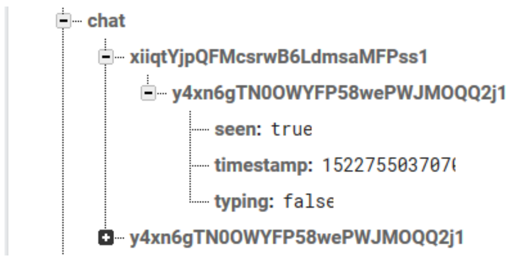
The structure of friends_list's database
|-notificataios
 |- sender_id
  |- receiver_id
   |- date



```
friend_request
  RYBpmCvpxaTDp6N3nRzBbaVmE6u1
    xiiqtYjpQFMcsrwB6LdmsaMFPss1
      request_type: "received"
```

Figure 4.2.7 friends_request's database in Firebase

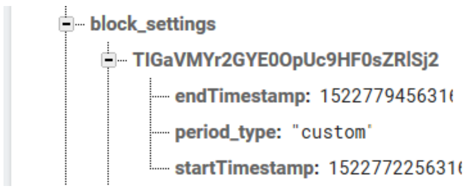
The structure of friends_list's database
|- friend_request
 |- sender_id
  |- receiver_id
   |- request_type



```
chat
  xiiqtYjpQFMcsrwB6LdmsaMFPss1
    y4xn6gTN0OWYFP58wePWJMOQQ2j1
      seen: true
      timestamp: 152275503707
      typing: false
  y4xn6gTN0OWYFP58wePWJMOQQ2j1
```

Figure 4.2.8 chat's database in Firebase

The structure of chat's database
|- chat
 |- sender_id
  |- receiver_id
   |- seen
   |- timestamp
   |- typing



```
block_settings
    TIGaVMYr2GYE0OpUc9HF0sZRlSj2
        endTimestamp: 152277945631
        period_type: "custom"
        startTimestamp: 152277225631
```

Figure 4.2.9 block_settings' database in Firebase

The structure of block_settings' database

```
|- block_settings'
        |- user_id
                |- endTimestamp
                |- period_type
                |- startTimestamp
```

4.3 Class Diagram

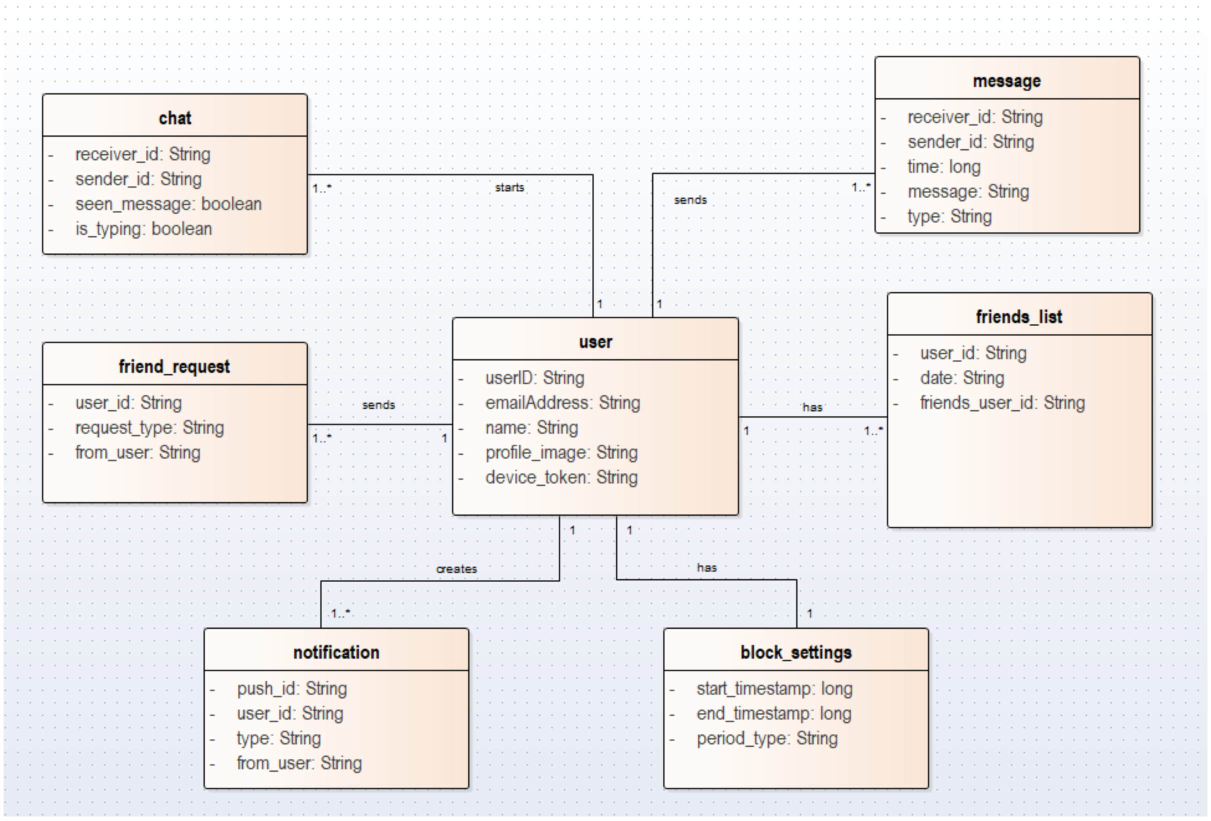


Figure 4.3.1 Class Diagram of the messaging app

4.4 Requirement

The requirements of the project will be divided into functional and non-functional requirements.

4.4.1 Functional Requirements

The messaging app should able to perform send and receive messages function. It includes text, image, voice and video messages. Auto blocking of incoming notification must be implemented to the messaging app to block incoming messages automatically after the setting is done by the user. The user must be able to customize the setting for auto blocking.

Previous messages should be deleted automatically according to the setting. User can set messages to be deleted by daily, weekly, monthly or by custom date. The messages that marked as important will not be removed automatically unless it is deleted by user manually.

4.4.2 Non-Functional Requirements

The app will be developed by using React Native to gain stability and cross-platform development at the same time. The performance of the messaging app must be fast in order to meet the instant send and receive of messages. The security level of the app must be high in order to secure the privacy of the users.

4.5 Project Prototype

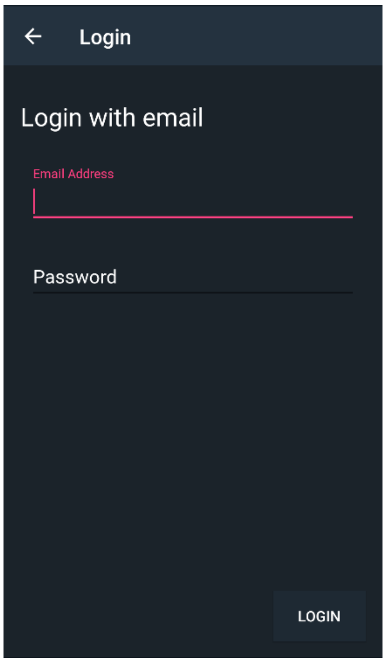
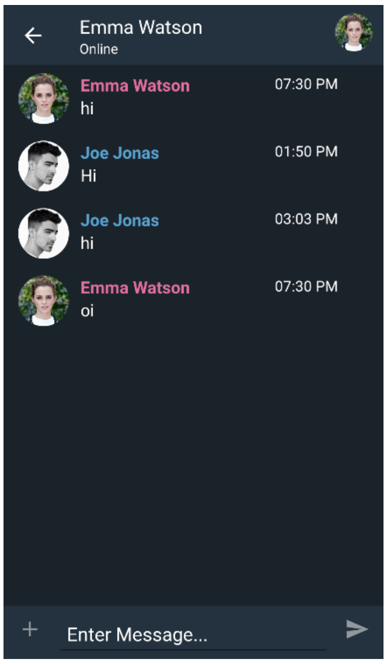


Figure 4.5.1
Login Activity


Emma Watson
Online

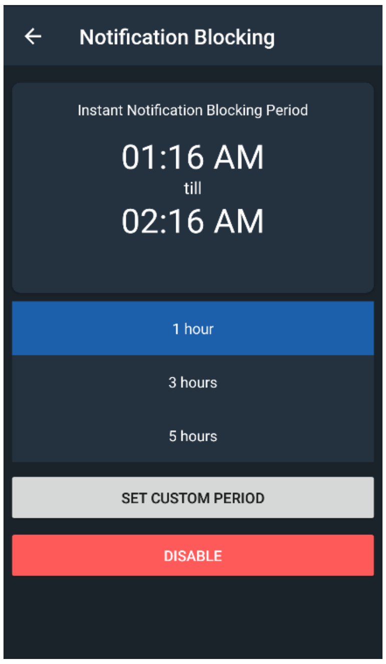
Emma Watson 07:30 PM
hi

Joe Jonas 01:50 PM
Hi

Joe Jonas 03:03 PM
hi

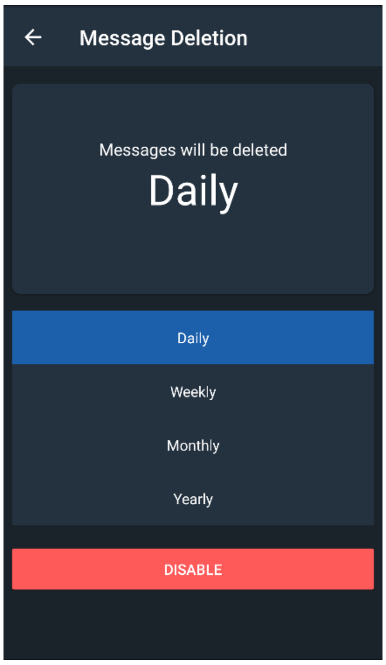
Emma Watson 07:30 PM
oi

Figure 4.5.2
Chat Activity



Instant Notification Blocking Period
01:16 AM
till
02:16 AM

Figure 4.5.3
Notification
Blocking Activity



Messages will be deleted
Daily

Figure 4.5.4
Message
Deletion Activity

Figure 4.5.1 is a simple login user interface for user to login to their account with email, phone number, or username. It allows user to retrieve forgotten password with email address.

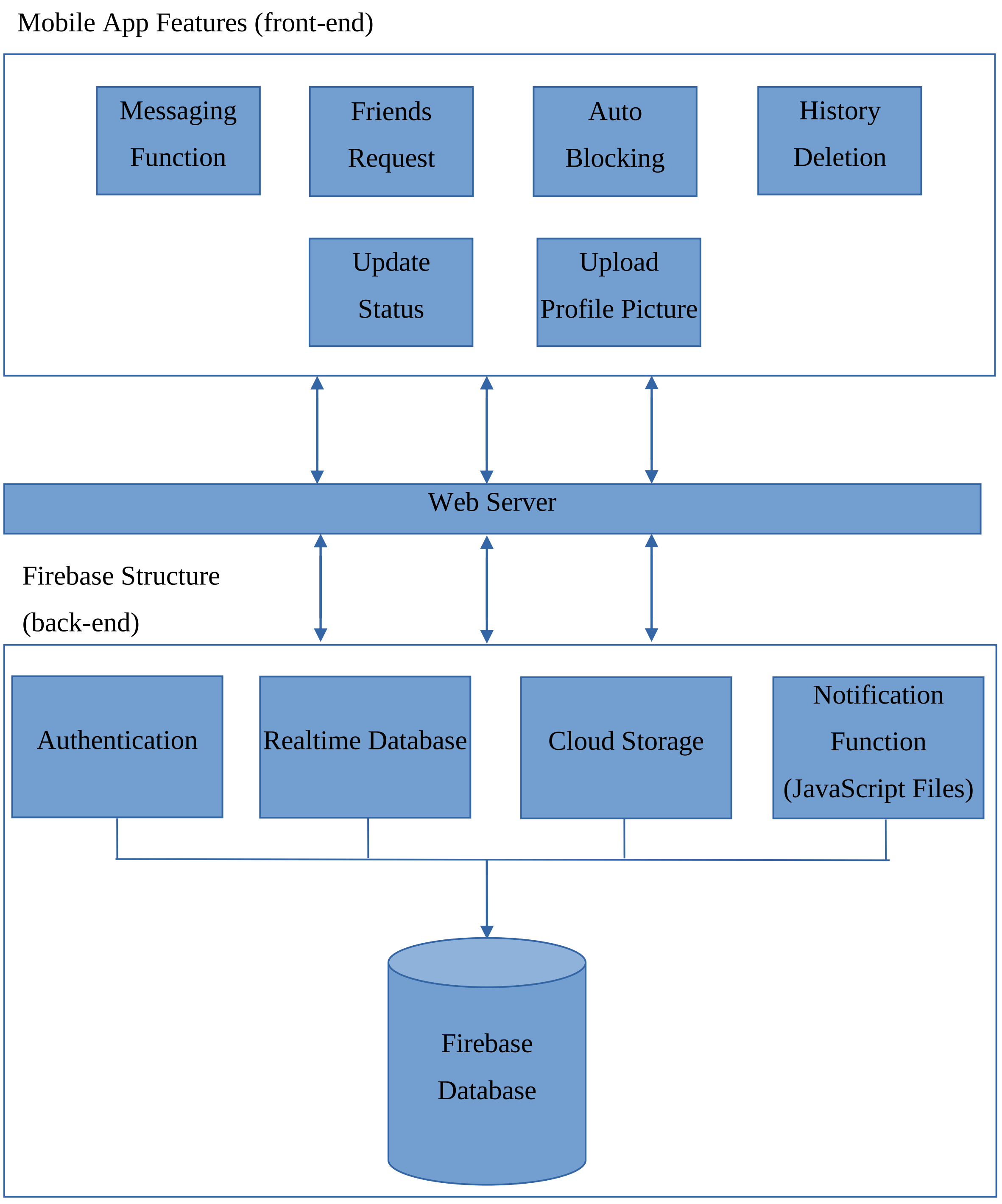
Figure 4.5.2 is a sample of chatting interface of the messaging app, the left-side conversation indicates the receiver messages and the right-side conversation indicates the sender messages.

Figure 4.5.3 is a user interface to let users set the auto blocking time or period and Figure 4.5.4 is the interface to customize message deletion to work automatically.

4.6 Development Framework

Name	**Description**	**Source**
CircleImageView	To transform a square imageview into circle imageview with just a line of code.	https://github.com/hdodenhof/CircleImageView
Android Image Cropper	To crop picture before uploading to the server. It's used to crop the area of profile picture that is needed by the user.	https://github.com/ArthurHub/Android-Image-Cropper
Compressor	To resize the resolution of the profile picture in order to store as a thumbnail.	https://github.com/zetbaitsu/Compressor
TextDrawable	To replace the profile picture with first letter of alphabet of the user's name if the user doesn't upload any profile picture.	https://github.com/amulyakhare/TextDrawable
Crashlytics	To detect crash at client side and receive error report.	https://fabric.io/kits/android/crashlytics
Firebase	To store user's information and messages	https://firebase.google.com/

4.7 System Architecture



CHAPTER 5
PROGRAM VERIFICATION

5.1 Testing

Mobile app testing is a collection of activities that similar with software testing in order to uncover as many errors as possible to improve customer satisfactions. The test will start from creating test cases of the mobile app, performing UAT and finally the device testing stage.

In this project, the testing project will be carried out by the app author. The test will be performed are listed below:

Testing stages	**Description**
Content Testing	To test and verify the content of the app to ensure the visibility of the content is suitable for the users
Interface Testing	To test the verify the interface to the app in order to ensure it is easy to manipulate by the user.
Component Testing	The test and verify each component of the app to prevent bugs like app crashing.
Configuration Testing	To test and verify the app on different brand and screen size of android device to ensure that the app is able to perform well in different device
Performance Testing	To test and verify the speed of opening the app to ensure good user experience
Security Testing	To test and verify the user information is secure from the database

Table 5.1: Testing stages

5.2 Content Testing

Test Case: Application information			
Description: To verify the adequacy of application information to the user in order to let user to have clear understanding on how to use the app.			
Step No.	**Step Description**	**Result**	**Step taken**
1	Find a user to test out the app without any help of user manual and guide.	Notification Blocking activity has confused the user on how to perform the function.	Pop-out a help manual to guide user to complete the function

Table 5.2: Content Testing for App

Test Case: Inconsistent of title			
Description: The title of activities is not consistent and it confuse the user when using it.			
Step No..	**Step Description**	**Result**	**Step taken**
1	Tap on the menu button, the “All User” option in drop-down bar is different with its activity	It will confuse user whether they have tap on the correct option	Change the name of activity to the name in the drop-down option

Table 5.3: Content Testing for App

5.3 Interface Testing

Test Case: Text Visibility			
Description: To enhance the text to ensure that user can be easily recognize and read.			
Step No.	**Step Description**	**Result**	**Step taken**
1	User's Name doesn't bold and the thickness of text is same with message description	It used up a little time for user to differentiate the user's display name and message text	Bold the user's display name

Table 5.4: Interface Testing for App

Test Case: Size of icon			
Description: The size of icon does not fit different size of phone.			
Step No.	**Step Description**	**Result**	**Step taken**
1	Check the of size of the icon for auto message deletion and auto blocking of notification is too big for a small size phone screen that below 5 inches	It affected the beauty of user interface and looked like a low-quality product	Use constraint layout instead of relative layout in android.

Table 5.5: Interface Testing for App

Test Case: Blank Chat Fragment			
Description: The Chat Fragment is blank when there is no conversation created.			
Step No.	**Step Description**	**Result**	**Step taken**
1	Find a user to test it out in order to find out whether the user knows the purpose of the Chat Fragment	The user doesn’t know what the purpose of Chat Fragment is.	Add a line of text in the center of Chat Fragment to let the user know how to add conversation.

Table 5.6: Interface Testing for App

5.4 Component Testing

Test Case: Testing on auto message deletion				
Description: To test and ensure message will be deleted by following the preset time.				
Step No.	**Step Description**	**Expected Result**	**Actual Result**	**Conclusion**
1	Test the auto message deletion button to ensure that the destination activity is correct.	The button navigates to Message deletion page	The button navigates to Message deletion page	The auto message deletion button is working fine as expected.
2	Test the radio button option “Daily”, “Weekly”, “Monthly”, and “Yearly”.	The button selected and saved, and all the deletion can be performed well.	The button selected and saved, and all the deletion can be performed well.	The radio button is working correctly.
3	Test “Disable” button	The button navigates back to the main activity, and the icon of auto message deletion will become red color to indicate the function is disabled	The button navigates back to the main activity, and the icon of auto message deletion is becoming red color to indicate the function is disabled	The disable button is working correctly as expected.

Table 5.7: Component Testing for App

Test Case: Testing on auto blocking of notification				
Description: To test and ensure message will be deleted by following the pre-set time.				
Step No.	**Step Description**	**Expected Result**	**Actual Result**	**Conclusion**
1	Test the auto notification blocking button to ensure that the destination activity is correct.	The button will navigate to the notification blocking activity	The button navigates to the notification blocking activity	The auto notification blocking button is working fine as expected.
2	Test the radio button option “1 hour”, “3 hours”, and “5 hours”	The button will be selected and saved into the phone preference and performed notification blocking	The button is selected and saved into the phone preference and performed notification blocking	The radio button is working correctly.
3	Test “Set Custom Period” Button	A time picker dialog will pop-out and prompt user to select the start time and end time. The blocking period will be shown.	A time picker dialog is pop-out and prompt user to select the start time and end time. The blocking period will be shown.	The “Set Custom Period” button is working correctly.
4	Test “Disable” Button	The button navigates back to the main activity, and the icon of auto notification blocking will become red color to indicate the function is disabled	The button navigates back to the main activity, and the icon of auto notification blocking is becoming red color to indicate the function is disabled	The “Disable” button is working correctly

Table 5.8: Component Testing for App

Test Case: Testing on chat components on Main Activity				
Description: To test and ensure every chat function is working correctly.				
Step No.	**Step Description**	**Expected Result**	**Actual Result**	**Conclusion**
1	Tap on the chat user to open profile and send message	A dialog box will pop-out with option of “Open Profile” and “Send message”. The “Open “Profile” option will navigate to user profile activity whereas the “Send message” option will navigate to a chat conversation activity.	A dialog box will pop-out with option of “Open Profile” and “Send message”. The “Open “Profile” option will navigate to user profile activity whereas the “Send message” option is navigate to a chat conversation activity.	The dialog box options are working correctly.
2	Menu button	A dialog box will pop-out with 3 options, “Account Settings”, “Global Users”, and “Logout”	A dialog box is pop-out with 3 options, “Account Settings”, “Global Users”, and “Logout”	The dialog box is working correctly.
3	Account Settings (From Menu button)	The option will navigate to user’s account settings activity	The option is navigated to user’s account settings activity	The “Account Settings” option is working correctly
4	Global Users (From Menu button)	The option will navigate to global users activity, and show all user of the app	The option is navigated to global users activity, and showed all user of the app	The “Global Users” option is working

				correctly
5	Logout (From Menu button)	The user will logout from firebase account and it will navigate to Login activity	The user is logout from firebase account and it is navigate to Login activity	The "Logout" option is working correctly

Table 5.9: Component Testing for App

5.5 Configuration Testing

Test Case: Test the app across different phone			
Description: To test the app content whether it is suitable for viewing by the user			
No.	**Phone Brand / Screen Size**	**Result**	**Step taken**
1	Xiaomi Redmi Note 2 / 5.5 inches	The size of text and icon is suitable for the phone	N/A
2	Xiaomi Redmi 1S / 4.7 inches	The size of text and icon is a little bit small but the content is still legible for the phone	N/A
3	Samsung Galaxy S3 Mini / 4 inches	The size of text and icon is small but the content is still legible for the phone	N/A

Table 5.10: Configuration Testing for App

5.6 Performance Testing

Test Case: Test the app startup speed and the transition speed between activities				
Description: To test and ensure every chat function is working correctly.				
Step No.	**Initial Activity**	**Destination Activity**	**Time taken**	**Conclusion**
1	Launcher Activity	Main Activity	2.7 seconds	The time taken is acceptable
2	Gorilla Fragment (Main Activity)	Auto Message Deletion Activity	0.8 second	The time taken is acceptable
3	Gorilla Fragment (Main Activity)	Auto Notification Blocking Activity	0.8 second	The time taken is acceptable
4	Chat Fragment (Main Activity)	Chat Activity	1.0 second	The time taken is acceptable
5	Main Activity	Account Settings Activity	0.6 second	The time taken is acceptable
6	Main Activity	User Activity	0.7 second	The time taken is acceptable
7	Main Activity	Login Activity	0.5 second	The time taken is acceptable
8	Login Activity	Main Activity	2.5 seconds	The time taken is acceptable

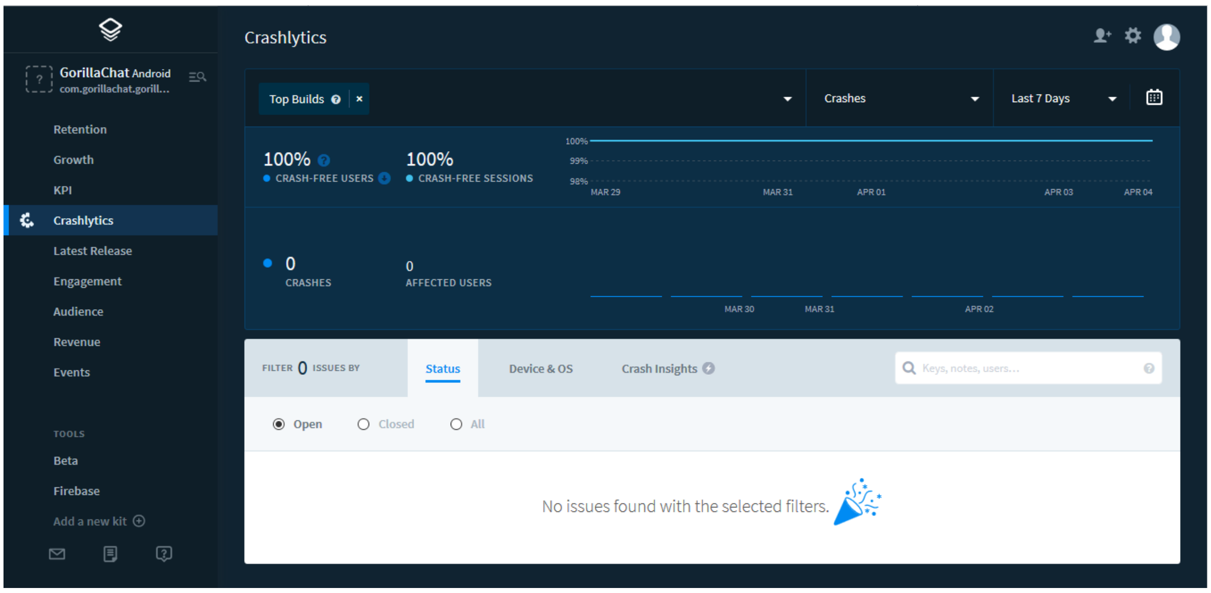
Table 5.11: Performance Testing for App

5.7 Security Testing

Test Case: Testing on user authentication				
Description: To test and ensure the user must enter the correct information to access their account				
Step No.	**Step Description**	**Expected Result**	**Actual Result**	**Conclusion**
1	Login with incorrect username but correct password	User will be not able to login	User is unable to login	The authentication is working correctly.
2	Login with correct username but incorrect password	User will be not able to login	User is unable to login	The authentication is working correctly.

Table 5.12: Security Testing for App

5.8 Bug Reporting



No issues found with the selected filters.

Figure 5.8.1 Crashlytic bug reporting

The messaging app will be implemented with a crash report plugin to ensure the app is free from crash and improve user experience.

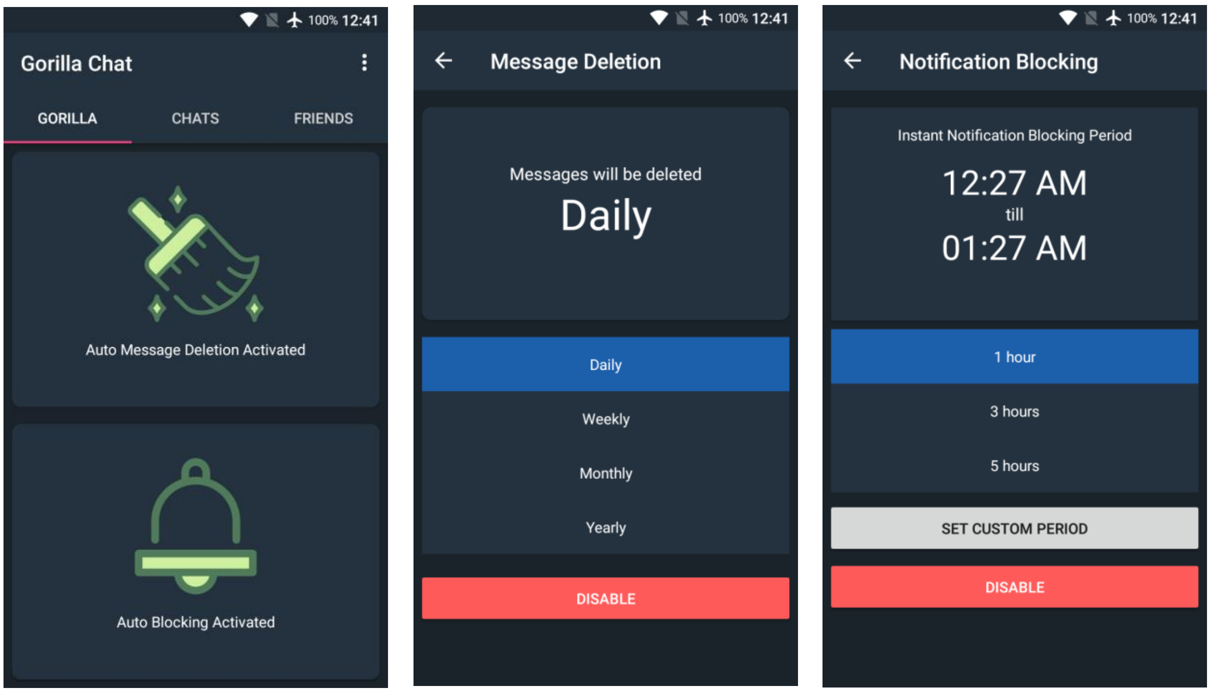
CHAPTER 6

SYSTEM RESULT AND FUNCTIONALITY

6.1 Introduction

In this section, the mobile app interface and the features will be shown. All the interface of the mobile app will be explained step by step on how to manipulate the function of the app. Beside the interface, the database of the mobile app will also be shown.

6.2 System Interface



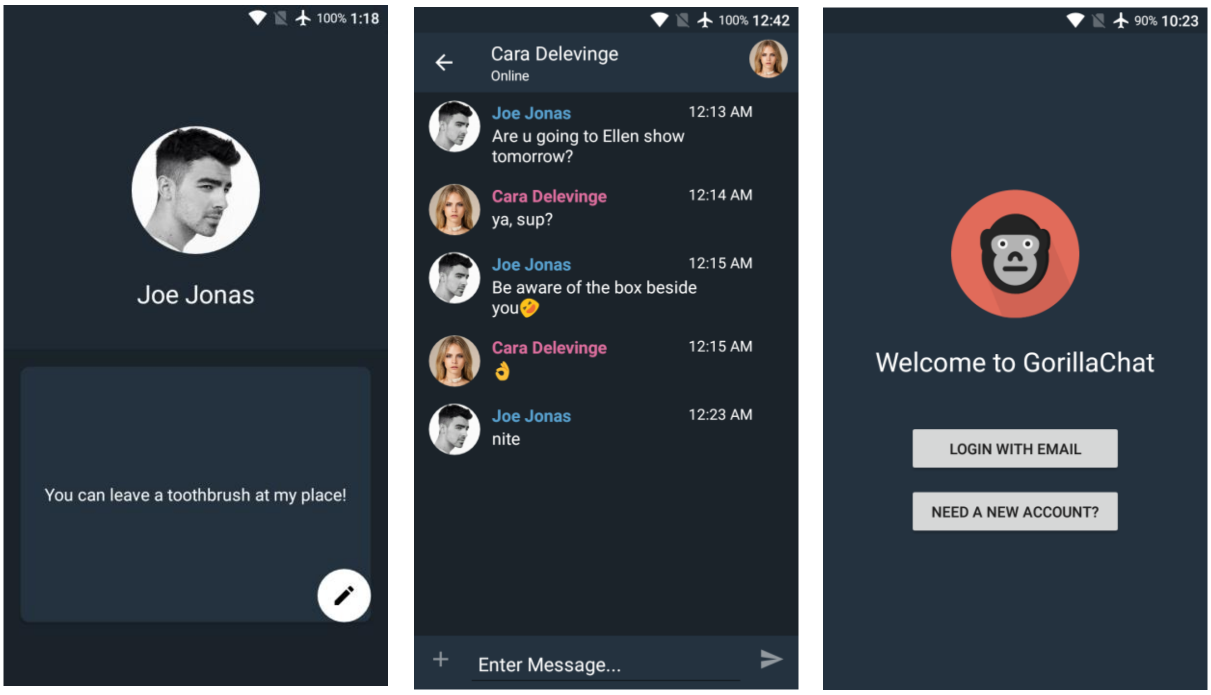
Auto Message Deletion Activated

Auto Blocking Activated

Messages will be deleted
Daily

Instant Notification Blocking Period
12:27 AM
till
01:27 AM

Figure 6.1 Interface of Auto Blocking and History Deletion



Joe Jonas

You can leave a toothbrush at my place!

Joe Jonas 12:13 AM
Are u going to Ellen show tomorrow?

Cara Delevinge 12:14 AM
ya, sup?

Joe Jonas 12:15 AM
Be aware of the box beside you😘

Cara Delevinge 12:15 AM
👌

Joe Jonas 12:23 AM
nite

Figure 6.2 Interface of Profile, Conversation, Register and Login

6.3 System demonstration

The demonstration will be having on 2 mobile devices for sending and receiving the messages.

6.3.1 Register and Login

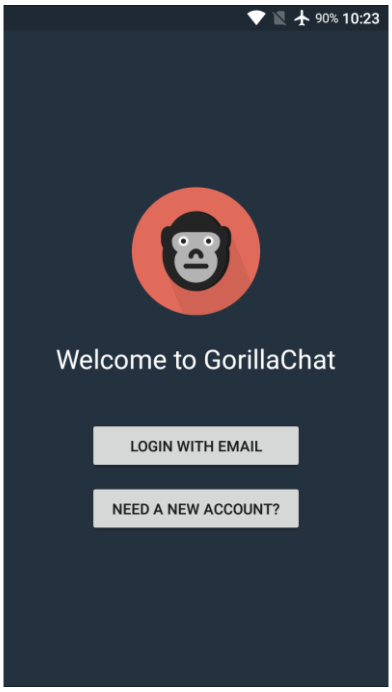


Figure 6.3.1.1
Welcome activity

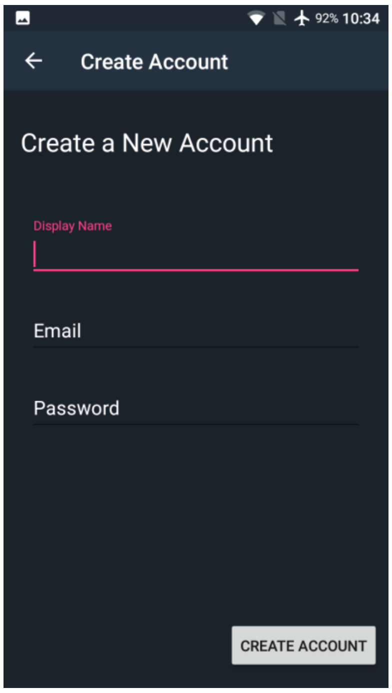


Figure 6.3.1.2
Register Activity

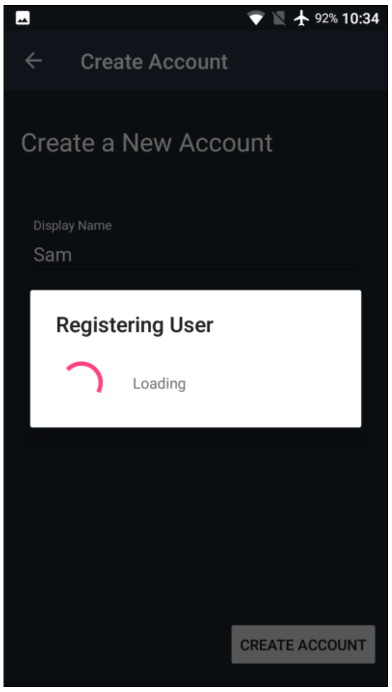


Figure 6.3.1.3
Processing Registration

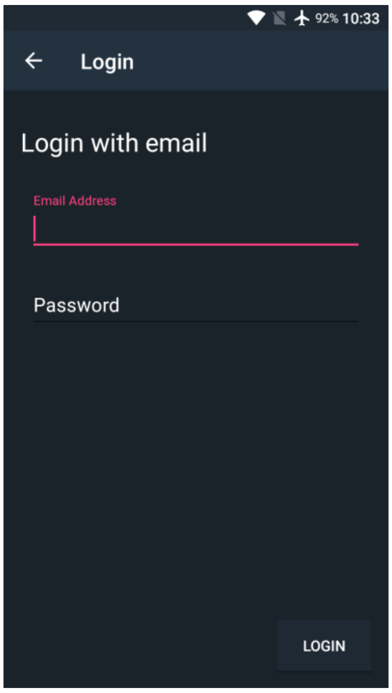


Figure 6.3.1.4
Login activity

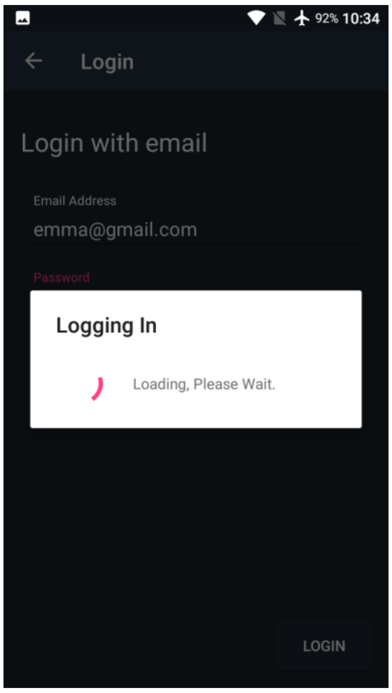
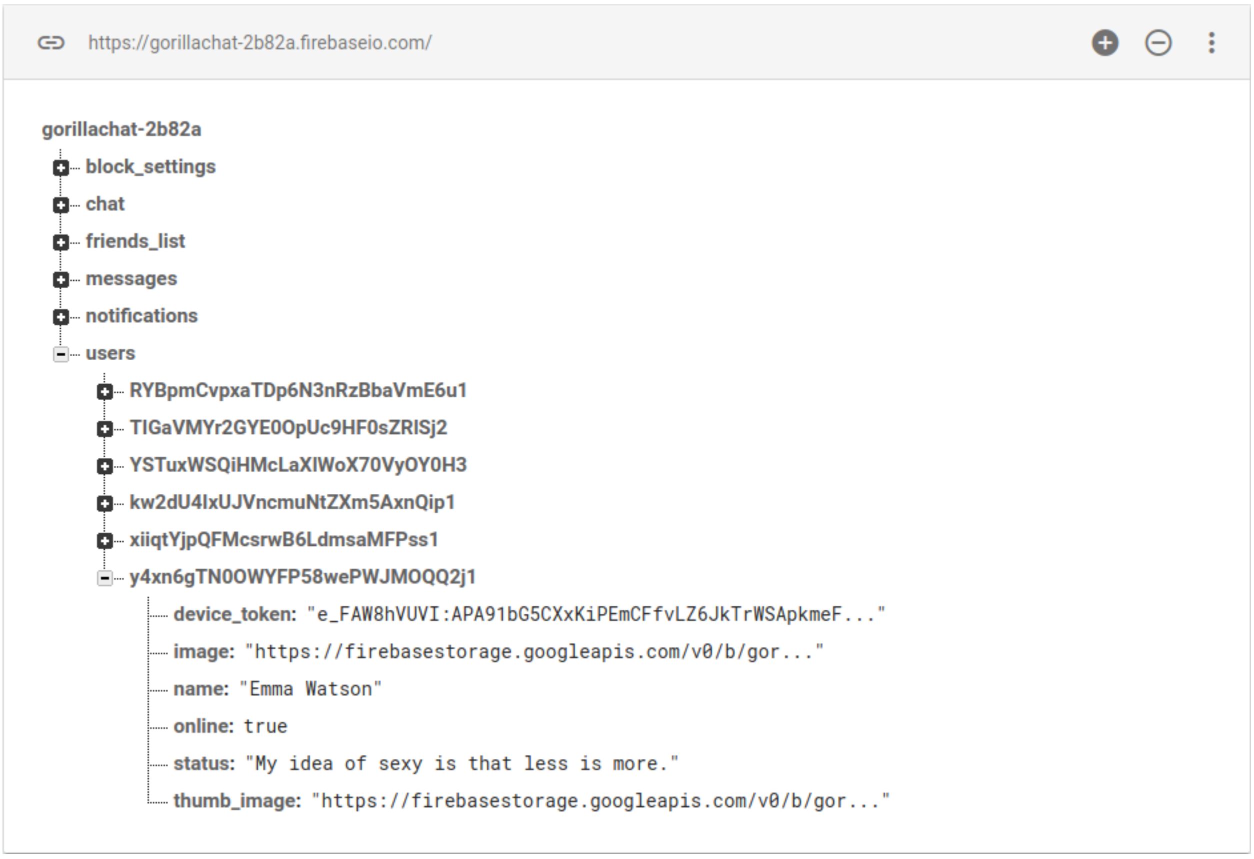


Figure 6.3.1.5
Login Process



```
https://gorillachat-2b82a.firebaseio.com/

gorillachat-2b82a
  block_settings
  chat
  friends_list
  messages
  notifications
  users
    RYBpmCvpxaTDp6N3nRzBbaVmE6u1
    TIGaVMYr2GYE0OpUc9HF0sZRISj2
    YSTuxWSQiHMcLaXIWoX70VyOY0H3
    kw2dU4IxUJVncmuNtZXm5AxnQip1
    xiiqtYjpQFMcsrwB6LdmsaMFPss1
    y4xn6gTN0OWYFP58wePWJMOQQ2j1
      device_token: "e_FAW8hVUVI:APA91bG5CXxKiPEmCFfvLZ6JkTrWSApkmeF..."
      image: "https://firebasestorage.googleapis.com/v0/b/gor..."
      name: "Emma Watson"
      online: true
      status: "My idea of sexy is that less is more."
      thumb_image: "https://firebasestorage.googleapis.com/v0/b/gor..."
```

Figure 6.3.1.6 New user added in Firebase

For registration of new user

The user need to tap on "NEED A NEW ACCOUNT" button to navigate to the registration activity and fill in all the columns that needed, which are the display name, email, and password. Then the user tapped on the "CREATE ACCOUNT" button to create their own account. An alert dialog will pop-out while the user information is sending to the server. In Firebase, the user information will be recorded.

For user to login

The user need to tap on "LOGIN WITH EMAIL" button to navigate to the login activity and fill in all the columns that needed, which are email and password in order to login their account. Then the user tapped on "LOGIN" button and an alert dialog will pop-out while the authentication process is going on. If the user's email and password is correct, user is able to login to their account.

6.3.2 Send and Accept Friend's request

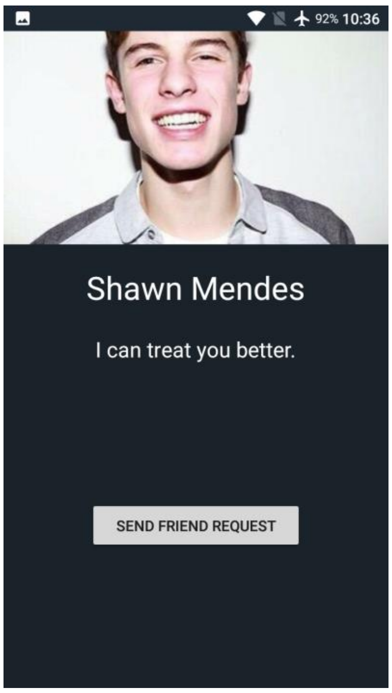


Figure 6.3.2.1
Send Friend Request

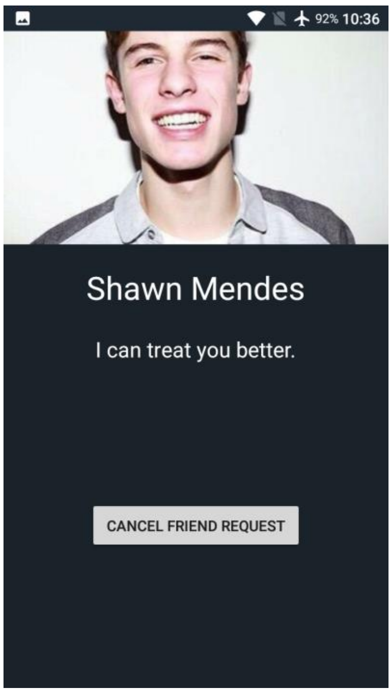


Figure 6.3.2.2
Cancel Friend Request

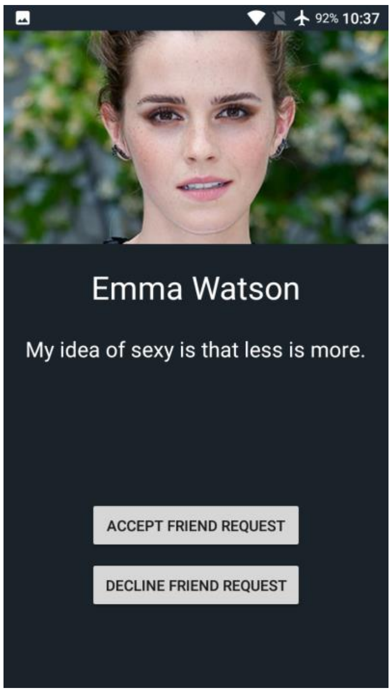


Figure 6.3.2.3
Accept/Decline Friend Request

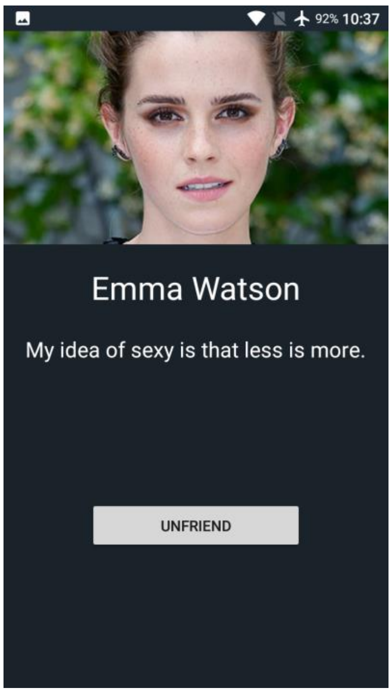
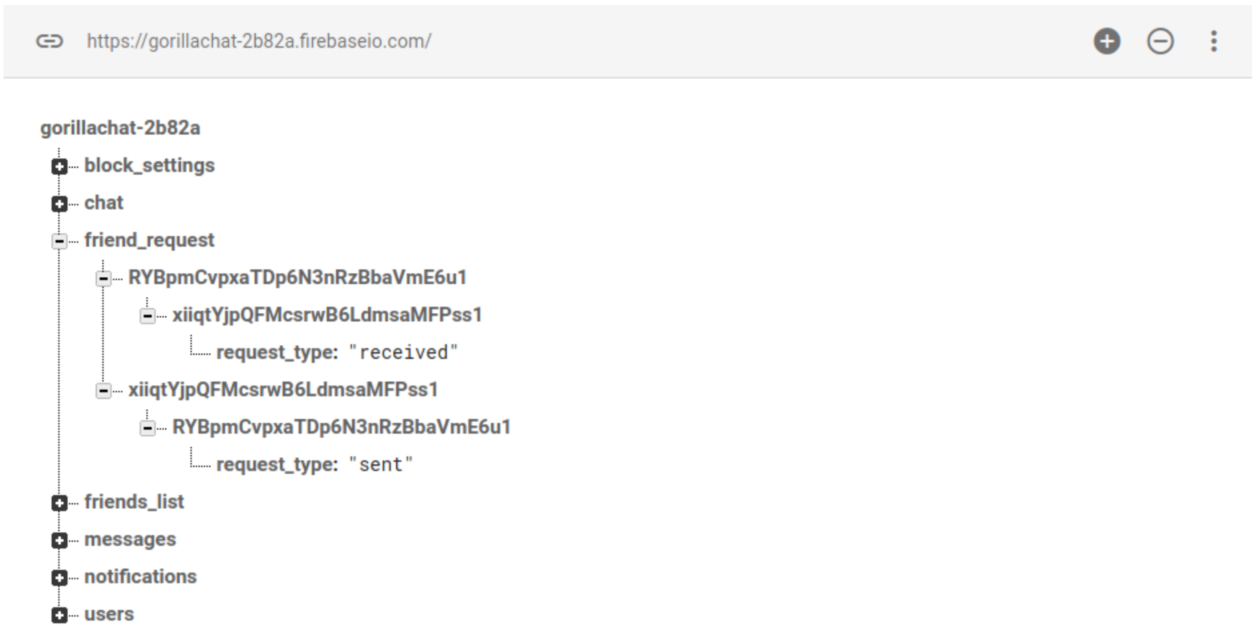


Figure 6.3.2.4
Unfriend



https://gorillachat-2b82a.firebaseio.com/

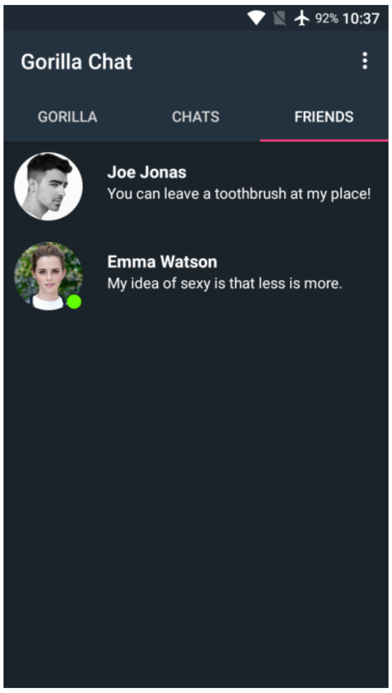
```
gorillachat-2b82a
  block_settings
  chat
  friend_request
    RYBpmCvpxaTDp6N3nRzBbaVmE6u1
      xiiqtYjpQFMcsrwB6LdmsaMFPss1
        request_type: "received"
    xiiqtYjpQFMcsrwB6LdmsaMFPss1
      RYBpmCvpxaTDp6N3nRzBbaVmE6u1
        request_type: "sent"
  friends_list
  messages
  notifications
  users
```

Figure 6.3.2.5 Friend request is created in Firebase

When the user need to add friend, he or she need to tap on the “SEND FRIEND REQUEST” button to perform the action. The “SEND FRIEND REQUEST” button will turn to “CANCEL FRIEND REQUEST” after the friend request was sent and the record was created on the database.

The person that received the friend request is able to perform 2 actions which are accept or decline friend request. It the “ACCEPT FRIEND REQUEST” button is tapped, the ‘ACCEPT FRIEND REQUEST” button will turn into “UNFRIEND” button, it is used to remove the friendship in the future.

6.3.3 Start a conversation



Joe Jonas
You can leave a toothbrush at my place!

Emma Watson
My idea of sexy is that less is more.

Figure 6.3.3.1
Friends Fragment

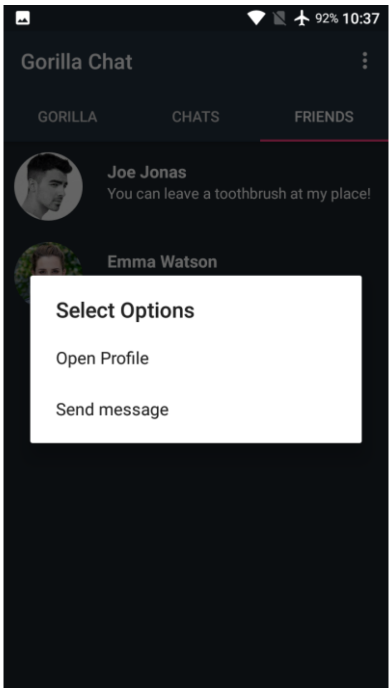
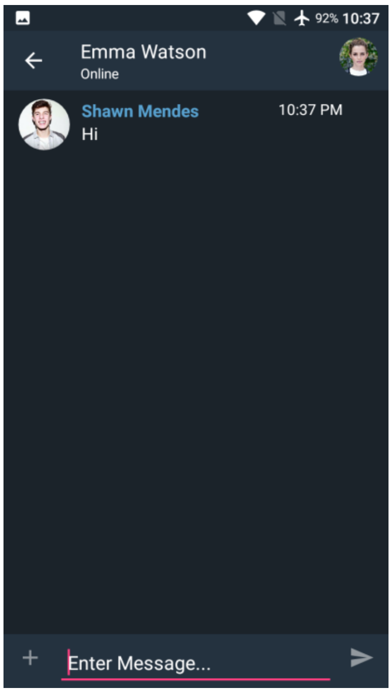


Figure 6.3.3.2
Dialog Options



Shawn Mendes 10:37 PM
Hi

Figure 6.3.3.3
Chat Activity

To start a new conversation, the user need to navigate to the “FRIENDS” tab, and tap on a friend, a dialog will pop-out to prompt the user for the next move. To check the profile, the user can tap on “Open Profile” option. To proceed on to a new conversation, user need to choose the “Send message” option. Then it will navigate to the chat activity, user can enter the message and send it to their friends.

6.3.4 Auto Message Deletion

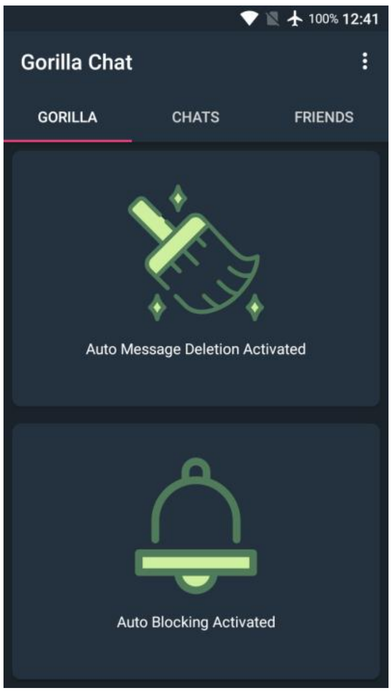


Figure 6.3.4.1
Gorilla Fragment

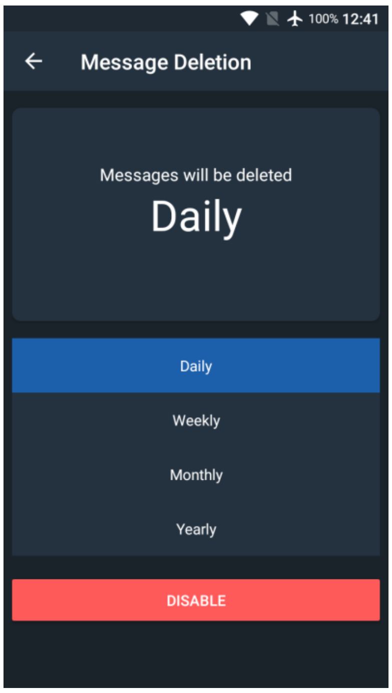


Figure 6.3.4.2
Message Deletion Activity

It is easy to perform the auto message deletion function, user need to choose the first option with a broom icon from the "GORILLA" tab, then it will navigate to the "Message Deletion" activity. The system allow user to choose the message deletion frequency by daily, weekly, monthly, and yearly. After choosing, the option will be saved. To disable the function, user just need to tap on the disable button.

6.3.5 Auto Notification Blocking

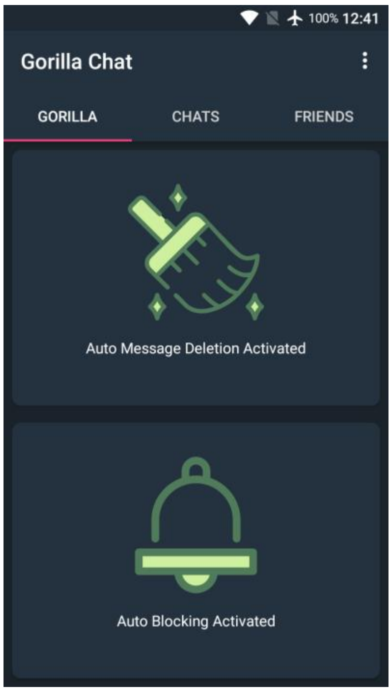


Figure 6.3.5.1
Gorilla Fragment

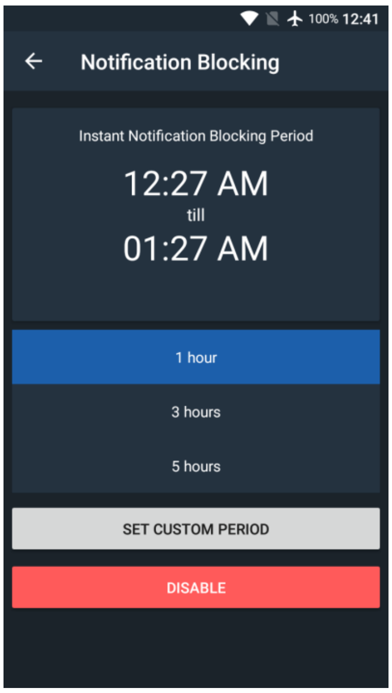


Figure 6.3.5.2
Instant Blocking Period

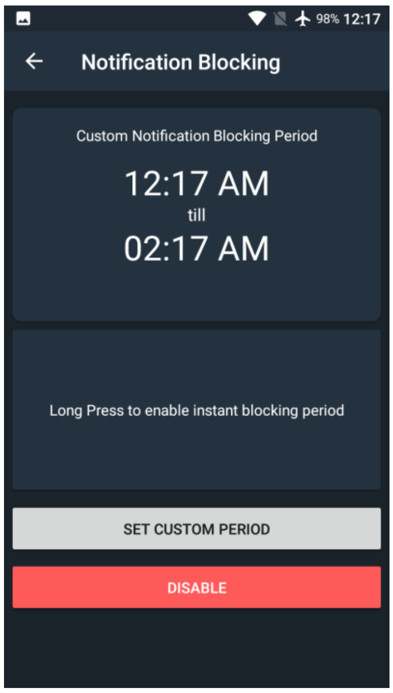

98% 12:17
Notification Blocking
Custom Notification Blocking Period
12:17 AM
till
02:17 AM
Long Press to enable instant blocking period
SET CUSTOM PERIOD
DISABLE

Figure 6.3.5.3
Custom Blocking Period

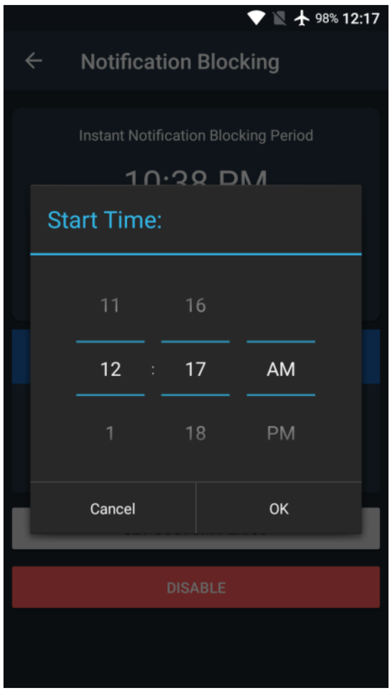


Figure 6.3.5.4
Set Start Time

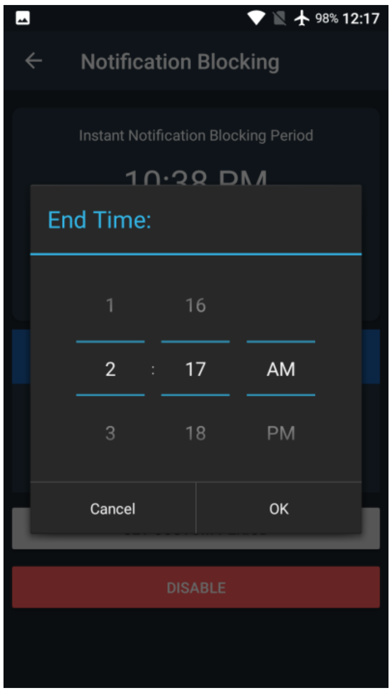
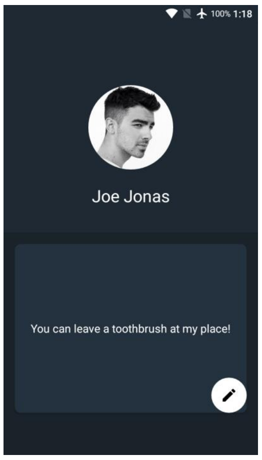
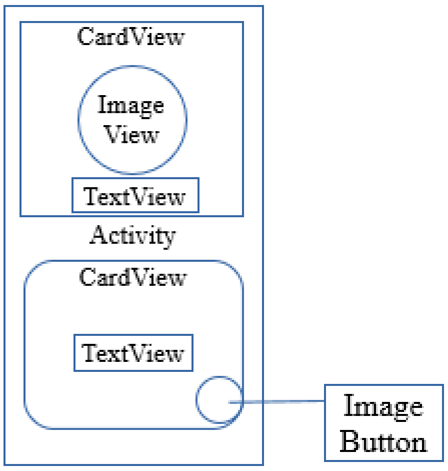


Figure 6.3.5.5
Set End Time

To perform auto notification blocking, the user need to choose the option with a bell icon on the “GORILLA” tab, then the notification blocking activity will show the instant notification blocking period in 1 hour by default. User can choose 1 hour, 3 hours, or 5 hours to block the message notification instantly for once. If the user decided to block it every day, the user should tap on “SET CUSTOM PERIOD” button, and a time picker dialog will pop-out and prompt user for the start and end time for the blocking period. After setting, the period will be saved. To switch back to instant block period, user needs to hold on the “Long press to enable instant block period” button. To disable the function, user needs to tap on the “DISABLE” button.

6.4 UI Structure



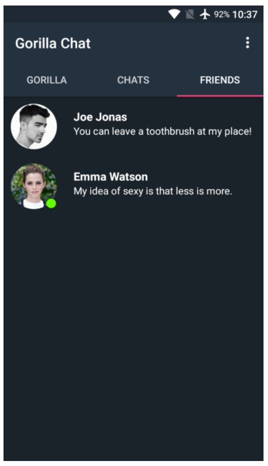


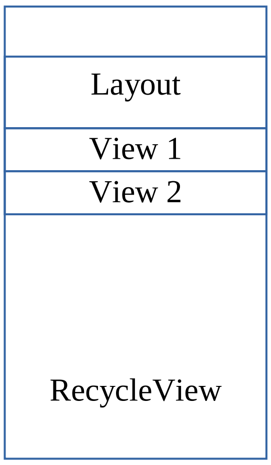
The user activity is made up by 1 imageview, 2 textview, 1 imagebutton, 2 cardviews in a Relative layout.

Code Layout

```
<RelativeLayout>

    <android.support.v7.widget.CardView>
        <RelativeLayout>

            <de.hdodenhof.circleimageview.CircleImageView />
            <TextView />

        </RelativeLayout>
    </android.support.v7.widget.CardView>

    <android.support.v7.widget.CardView>
        <RelativeLayout>

            <TextView />
            <ImageButton/>

        </RelativeLayout>
    </android.support.v7.widget.CardView>

</RelativeLayout>
```

Recycle View


Gorilla Chat
GORILLA
CHATS
FRIENDS
Joe Jonas
You can leave a toothbrush at my place!
Emma Watson
My idea of sexy is that less is more.



The list of the users is setup with a recycleview that can hold a number of single piece of rectangular view with the viewholder method. View 1 indicates the first view that populate with the information of the users in an array. It is similar to Chat Fragment.

Code Layout

```
<FrameLayout

        <android.support.v7.widget.RecyclerView />

</FrameLayout>
```

Method that use to populate the view into recycleview

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

        Code definition refer to bitbucket
        This method is used to create a view for the friends fragment to hold
        the single rectangular view.
}


@Override
public void onStart() {
     super.onStart();
FirebaseRecyclerOptions<Friends> friendsRecycleViewOptions = new
FirebaseRecyclerOptions.Builder<Friends>().setQuery(dFriendsDatabase,
Friends.class).build();

FirebaseRecyclerAdapter<Friends, FriendsViewHolder> friendsRecycleViewAdapter =
new FirebaseRecyclerAdapter<Friends, FriendsViewHolder>(friendsRecycleViewOptions)
{
```

```
            @NonNull
            @Override
public FriendsViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
                  View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.users_single_layout,parent,f
alse);
                  return new FriendsViewHolder(view);
            }
            @Override
            protected void onBindViewHolder(@NonNull final FriendsViewHolder
friendsViewHolder, int position, @NonNull Friends friends) {
```

The FriendsViewHolder method is called to pass the data from Firebase Database into the TextView and ImageView and populate the viewholder into the RecycleView.

```
}
                        @Override
                        public void onCancelled(DatabaseError databaseError) {
                        }
                  });
            }
      };
      friendsRecycleViewAdapter.startListening();
      friendList.setAdapter(friendsRecycleViewAdapter);
}



public static class FriendsViewHolder extends RecyclerView.ViewHolder{
```

//List all the 'View'(TextView, ImageView, CardView) in this method to be refer by the onCreateViewHolder.

```
         View mView;
      private TextViewHelper textViewHelper;
      public FriendsViewHolder(View itemView) {
            super(itemView);
            mView = itemView;
      }
      public void setStatus(String status){
            TextView userStatusView = (TextView)
mView.findViewById(R.id.userliststatus);
            userStatusView.setText(status);
      }
      public void setName(String name){
            TextView userNameView = (TextView) mView.findViewById(R.id.username);
            userNameView.setText(name);
            userNameView.setTypeface(userNameView.getTypeface(), Typeface.BOLD);
      }
      public void setUserImage(String name, String thumb_image, String image,
Context context){
            CircleImageView userImageView = (CircleImageView)
mView.findViewById(R.id.userListImage);
            textViewHelper = new TextViewHelper();
            TextDrawable placeHolder = textViewHelper.textDrawableRandColor(64,64,
name);
            if (!image.equals("default")){

Picasso.with(context).load(thumb_image).placeholder(placeHolder).into(userImageView);
            } else {
                  userImageView.setImageDrawable(placeHolder);
            }
      }
      public void setUserOnline(String onlineStatus){
            ImageView friendsOnlineIcon = (ImageView)
mView.findViewById(R.id.online_icon);
            if(onlineStatus.equals("true")){
                  friendsOnlineIcon.setVisibility(View.VISIBLE);
            }else{
                  friendsOnlineIcon.setVisibility(View.INVISIBLE);
            }
      }
```

6.5 Auto Blocking (How it works?)

The instant blocking period will be chosen by the users (1, 3, 5 hours) and the period will save into SharedPreference by calling the Editor method. For example, if the current time is 12.00pm, and the instant blocking period is 3 hours, then the period of blocking will be 12.00pm to 3.00pm. The time format will be saved as timestamp and then post to the Firebase database.

```
public void savePeriod(final int hour){
   //Get start time in timestamp
     final Calendar calendar = Calendar.getInstance();
     editor.putLong("startTime", calendar.getTimeInMillis());
     //Post period type to Firebase database
   mBlockDatabase.child("period_type").setValue("instant");
     //Save the start and end timestamp to SharedPreference and Firebase
Database
mBlockDatabase.child("startTimestamp").setValue(calendar.getTimeInMillis());
     calendar.add(Calendar.HOUR, hour);
     editor.putLong("endTime",calendar.getTimeInMillis());
     mBlockDatabase.child("endTimestamp").setValue(calendar.getTimeInMillis());
     editor.putInt("hour",hour);
     editor.apply();
}
```

Then the blocking period will get from Firebase database and use to verify that the needs of sending out a notification. If the user that is having a blocking period, and the current time is in between the blocking period, then the notification will not be created in the Firebase database.

```
this.mBlockDatabase.addValueEventListener(new ValueEventListener() {

     @Override
     public void onDataChange(DataSnapshot dataSnapshot) {
          if (dataSnapshot.hasChild(mChatUser)){
               Long startTime = (Long)
dataSnapshot.child(mChatUser).child("startTimestamp").getValue();
               Long endTime = (Long)
dataSnapshot.child(mChatUser).child("endTimestamp").getValue();
               Long currentTime = System.currentTimeMillis();

               if (currentTime > startTime && currentTime < endTime){
                    Toast.makeText(ChatActivity.this,"No Notification",
Toast.LENGTH_SHORT).show();
               }else{
                    HashMap<String, String> msgNotifiction = new HashMap<>();
                    msgNotifiction.put("from", mCurrentUserID);
                    msgNotifiction.put("type", "message");

mMsgNotificationDatabase.child(mChatUser).push().setValue(msgNotifiction);
               }
          }else {
               HashMap<String, String> msgNotifiction = new HashMap<>();
               msgNotifiction.put("from", mCurrentUserID);
```

```
                msgNotifiction.put("type", "message");

mMsgNotificationDatabase.child(mChatUser).push().setValue(msgNotifiction);
          }
      }
      @Override
      public void onCancelled(DatabaseError databaseError) {
      }
});
```

Below is the JavaScript function that written to detect the notification, if the notification request is created in the Firebase database

```
'use strict'
```

//get require library from firebase functions and admin

```
const functions = require('firebase-functions');

const admin = require('firebase-admin');

admin.initializeApp(functions.config().firebase);
```

//Export the function 'sendNotification' to Firebase

```
exports.sendNotification =
```

//Refer to database and listen to *notifications*/user_id/notification_id when it is created

```
functions.database.ref('/notifications/{user_id}/{notification_id}').onWrite(event => {
```

// get the user_id of the receiver and the notification_id

```
    const user_id = event.params.user_id;

    const notification_id = event.params.notification_id;

    if(!event.data.val()){

        return console.log('A Notification has been delete', notification_id);

    }

    const fromUser =
admin.database().ref(`/notifications/${user_id}/${notification_id}`).once('value');
```

//Return the notification the the user's device

```
    return fromUser.then(fromUserResult => {

        const from_user_id = fromUserResult.val().from;

        const type = fromUserResult.val().type;
        console.log('You have new notification from ', from_user_id);

        console.log("The type of notification", type);
```

//Get user’s name value and deviceToken value

```
        const userQuery =
admin.database().ref(`/users/${from_user_id}/name`).once('value');

        const deviceToken =
admin.database().ref(`/users/${user_id}/device_token`).once('value');

        return Promise.all([userQuery, deviceToken]).then(result => {

            const userName = result[0].val();

            const token_id = result[1].val();

            var payload = {};
```

//the friend request notification will be sent if the type of notification is ‘request’

```
            if (type === 'request') {

                console.log('It a friend request');

                payload = {

                    notification: {

                        title : "Friend Request",

                        body  : `${userName} has sent you request`,

                        icon  : "default",

                        click_action : "com.gorillachat.gorillachat_TARGET_NOTIFICATION"

                    },

                    data : {

                        user_id : from_user_id

                    }

                };

            }
```

//the message notification will be sent if the type of notification is ‘message’

```
            if (type === 'message'){
                console.log('It a msg notification');
                payload = {
                    notification: {
                        title : "New Message",
                        body  : `${userName} has sent you a message`,
                        icon  : "default",
                        click_action : "com.gorillachat.gorillachat_MESSAGE_NOTIFICATION"
                    },
                    data : {
                        user_id : from_user_id,
                        userName : userName
                    }
                };
            }
            return admin.messaging().sendToDevice(token_id, payload).then(response =>
{
                return console.log('Notification');
            });
        });
    });
});
```

6.6 History Deletion (How it works?)

There are 4 options (Daily, Weekly, Monthly, Yearly) for user to choose for the auto history deletion function.

```
public void saveAutoCleanSettings(String cleanFreq){
//The time is store in timestamp in order to convert to day, month, and
year easily
Long currentTimestamp = System.currentTimeMillis();
      SharedPreferences sharePref = getSharedPreferences("cleanInfo",
Context.MODE_PRIVATE);
      SharedPreferences.Editor editor = sharePref.edit();
      //The cleanFreq will record the user preference whether it is daily,
weekly, monthly or yearly and the timestamp will also save with
SharedPreference.
   editor.putString("cleanFreq", cleanFreq);
      editor.putLong("savedTimestamp", currentTimestamp);
      editor.apply();
      Toast.makeText(AutoCleanActivity.this, "Saved", Toast.LENGTH_SHORT).show();
      String cleanFreqValue = sharePref.getString("cleanFreq","");
      deletionFrequency.setText(cleanFreqValue);
}
```

The class below is the class that will perform message deletion automatically once the user enter the app.

```
public class MessageDeletionService extends Activity{
      private FirebaseAuth mAuth;
      private DatabaseReference mMessageDatabase;
      private DatabaseReference mChatDatabase;
      private SharedPreferences sharedPref;
      private Long currentTimestamp = System.currentTimeMillis();
      public MessageDeletionService(SharedPreferences sharedPref){
            this.sharedPref = sharedPref;
      }
      public void messageDeletion(){
            SimpleDateFormat simpleDateFormat;
            mAuth = FirebaseAuth.getInstance();
            String currentUserID = mAuth.getCurrentUser().getUid();
            mMessageDatabase =
FirebaseDatabase.getInstance().getReference().child("messages").child(currentUserID);
            mChatDatabase =
FirebaseDatabase.getInstance().getReference().child("chat").child(currentUserID);
            Context appContext = MainActivity.getContextOfApp();
            String clearFreq = sharedPref.getString("cleanFreq","");
            Long savedTimestamp = sharedPref.getLong("savedTimestamp", 0);
            Toast.makeText(appContext, "Clean Frequency : " +
clearFreq ,Toast.LENGTH_SHORT).show();
            switch (clearFreq){

                  case "Daily":
                        simpleDateFormat = new SimpleDateFormat("dd");
                        String currentDate = simpleDateFormat.format(currentTimestamp);
```

```
                String savedTimestampToDD =
simpleDateFormat.format(savedTimestamp);
//If the current Date is 1, it will not greater than 30, or 31, then we need to
update the saveTimestamp with currentTimestamp to make the date start again
from 1
                if (currentDate == "1"){
                    updateSharedPref();
                }else {
                    if (Integer.parseInt(currentDate) >
Integer.parseInt(savedTimestampToDD)){
                        updateSharedPref();
                    }
                }
                break;
//If the current week is 1, it will not greater than 52, so we need to update the
saveTimestamp with currentTimestamp to make the date start again from 1
            case "Weekly":
                simpleDateFormat = new SimpleDateFormat("ww");
                String currentWeek = simpleDateFormat.format(currentTimestamp);
                String savedTimestampToWW =
simpleDateFormat.format(savedTimestamp);

                if (currentWeek == "1"){
                    updateSharedPref();
                }else {
                    if(Integer.parseInt(currentWeek) >
Integer.parseInt(savedTimestampToWW)){
                        updateSharedPref();
                    }
                }
                break;
//If the current month is 1, it will not greater than 12, so we need to update
the saveTimestamp with currentTimestamp to make the date start again from 1
            case "Monthly":
                simpleDateFormat = new SimpleDateFormat("MM");
                String currentMonth =
simpleDateFormat.format(currentTimestamp);
                String savedTimestampToMM =
simpleDateFormat.format(savedTimestamp);
                if (currentMonth == "1"){
                    updateSharedPref();
                }else {
                    if(Integer.parseInt(currentMonth) >
Integer.parseInt(savedTimestampToMM)){
                        updateSharedPref();
                    }
                }
                break;
            case "Yearly":
                simpleDateFormat = new SimpleDateFormat("yyyy");
                String currentYear = simpleDateFormat.format(currentTimestamp);
                String savedTimestampToyyyy =
simpleDateFormat.format(savedTimestamp);
                if(Integer.parseInt(currentYear) >
Integer.parseInt(savedTimestampToyyyy)){
                    updateSharedPref();
                }
                break;
                default:
                    break;
        }
    }
//Delete the message and update sharepreference
    private void updateSharedPref(){
//Remove messages
        mMessageDatabase.removeValue();
        mChatDatabase.removeValue();
//Update savedTimestamp with currentTimeStamp
        SharedPreferences.Editor editor = sharedPref.edit();
        editor.putLong("savedTimestamp", currentTimestamp);
        editor.apply();
    }
}
```

CHAPTER 7
CONCLUSION AND RECOMMENDATION

7.1 Conclusion

There are a few of messaging app in the market currently, but it is rare to see the messaging that implemented the auto message deletion function. Most of the messaging app in the market is letting their user to delete chat messages manually, it is considered as time consuming for the process of message deletion. Therefore, to ease user for doing such a repetitive process, the auto message deletion function is implemented to the messaging app. The auto message deletion function can help user to remove their message by daily, weekly, monthly, or yearly. Besides that, the period auto notification blocking function is also able to fully customize by the user. The mobile app user can set the blocking period once and use it until they want to change it.

It is easy to setup a real-time database with Google Firebase for a messaging app. Firebase provides a NoSQL database for storing the data, therefore it is easy to store and access the data in real-time.

7.2 Recommendation

In order to in sync with the technology in the future, the massaging app should implement with block-chain technology in order to increase the security and integrity of massaging data. It can overcome the limitations that related to identity control, eliminates intermediaries and third parties and allows any party involved to send and share value with other elements on the network, and ensuring speed, authenticity and verifiability.

For the automation of the notification blocking and message deletion, it should implement with machine algorithm in the future in order to track user habit and perform the relevant action. The messages also should be backup before the action of message deletion.