****

**ANDROID HOME SCREEN EMERGENCY BUTTON**

**WITH GPS MONITORING**

**(YOUR URGENT WIDGET)**

By

Ardy Jaya Putrasalim

001201400007

A Thesis

Submitted to the Faculty of Computing

President University

in Partial Fulfillment of the Requirements

for the Degree of Bachelor of Science

in Information Technology

Cikarang, Bekasi, Indonesia

March 2018

Copyright

Ardy Jaya Putrasalim

2018

**ANDROID HOME SCREEN EMERGENCY BUTTON**

**WITH GPS MONITORING**

**(YOUR URGENT WIDGET)**

By

Ardy Jaya Putrasalim

Approved:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Rila Mandala, Ph.D Nur Hadisukmana, M.Sc

Thesis Advisor Program Head of Information Technology

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Rila Mandala, Ph.D

Dean of Faculty of Computing

**DECLARATION OF ORIGINALITY**

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee, and that this thesis has not been submitted for a higher degree to any other University or Institution.

I have read the Thesis Regulations and I am aware of the potential consequences of any breach of them.

Cikarang*,* March 29, 2018

Ardy Jaya Putrasalim

**ABSTRACT**

Smartphone is one of the most important thing for people nowadays. People use it on daily basis to assist their work and activities. Not only as communication tools, it also has become a tool to help people do their task. Android is one of people’s favorite Operating System which they use on their smartphone. GPS is one of Android feature that helps people with location related things like knowing their own location to show them the geographic around them, or to share their location to other people so the another people can see their precise location.

This thesis, Your Urgent Widgetwill discuss the development of GPS tracking to inform nearby Police Public Officer, Firefighter, or Ambulancethe user’s location. Application will let these nearest Public Officers to track the user depends on which widget the user tap. This thesis purpose is to make emergency call response faster than before so there can be crime prevention, early disaster countermeasure, and fast ambulance service or first aid assistance. The application is based on the Android operating system and uses the GPS (Global Positioning System) in tracking the caller’s position.

# DEDICATION

*To my parents, who believe in me and give endless support in finishing my academic study.*

# ACKNOWLEDGEMENT

Above all, I would like to thank God who gives me the blessing that I am able to develop this thesis. The success of this thesis truly depends on the combination of my efforts along with encouragement and guidelines from many others. I would like to express my gratitude to them who made this thesis possible:

1. Parents and sister who always support me in every condition I may be in.
2. My thesis advisor, Mr. Rila Mandala, who gave me guidance in this research.
3. All of Computing faculty lecturers, especially Mr. Nur Hadisukamana, Mr. Rikip Ginanjar, Mr. Tjong Wan Sen, and Mr. Eddo Fajar Nugroho who gave me knowledge and also Kak Novy Astria and Mba Inda who provided help for all Computing students.
4. Yoshi Toliu for designing the materials for this application.
5. James Victorio, Johanes Parlindungan Hutagalung, and Ricky Sulianto for being brothers, supporters, and roommates.
6. All Computing students, especially students of batch 2014 for a journey together in Computing faculty.
7. And for everyone that I can’t mention one by one which always tell me to do this thesis faster. Thank you help me and bring the joy in my university life. I am glad to know you all.

# Table of Contents

[DEDICATION ii](#_Toc511356027)

[ACKNOWLEDGEMENT iii](#_Toc511356028)

[Table of Contents iv](#_Toc511356029)

[LIST OF FIGURES vii](#_Toc511356030)

[LIST OF TABLES x](#_Toc511356031)

[CHAPTER I INTRODUCTION 1](#_Toc511356032)

[1.1 Background 1](#_Toc511356033)

[1.2 Problem Statement 2](#_Toc511356034)

[1.3 Thesis Objective 3](#_Toc511356035)

[1.4 Scope and Limitation 3](#_Toc511356036)

[1.5 Methodology 4](#_Toc511356037)

[1.6 Thesis Outline 6](#_Toc511356038)

[CHAPTER II LITERATURE STUDY 8](#_Toc511356039)

[2.1 Global Positioning System (GPS) 8](#_Toc511356040)

[2.1.1 GPS System Flow 8](#_Toc511356041)

[2.1.2 GPS Signal 9](#_Toc511356042)

[2.1.3 GPS Accuracy 11](#_Toc511356043)

[2.2 Google Maps 12](#_Toc511356044)

[2.3 Latitude and Longitude 13](#_Toc511356045)

[2.4 Mobile Phone Tracking 14](#_Toc511356046)

[2.4.1 Tracking Mobile Phone Location Technique 15](#_Toc511356047)

[2.5 Firebase 16](#_Toc511356048)

[2.5.1 Firebase Open Source Project 17](#_Toc511356049)

[2.5.2 Firebase Development 18](#_Toc511356050)

[2.6 Test Methodologies 20](#_Toc511356051)

[2.6.1 Black Box Testing 20](#_Toc511356052)

[2.6.2 White Box Testing 20](#_Toc511356053)

[2.7 Related Work 21](#_Toc511356054)

[2.7.1 Grab 21](#_Toc511356055)

[2.7.2 Zenly 21](#_Toc511356056)

[CHAPTER III SYSTEM ANALYSIS 23](#_Toc511356057)

[3.1 System Overview 23](#_Toc511356058)

[3.2 Application Requirement 23](#_Toc511356059)

[3.2.1 Functional Requirement 23](#_Toc511356060)

[3.2.2 System Requirements 24](#_Toc511356061)

[3.3 Comparison Overview 26](#_Toc511356062)

[3.4 Use Case Diagram 26](#_Toc511356063)

[3.5 Use Case Narrative 27](#_Toc511356064)

[3.6 Activity Diagram 31](#_Toc511356065)

[CHAPTER IV SYSTEM DESIGN 33](#_Toc511356066)

[4.1 User Interface Design 34](#_Toc511356067)

[4.2 Database Structure Design 38](#_Toc511356068)

[4.2.1 Initial state 38](#_Toc511356069)

[4.2.2 Officer Available state 40](#_Toc511356070)

[4.2.3 Request State 41](#_Toc511356071)

[4.3 Physical Design 41](#_Toc511356072)

[CHAPTER V SYSTEM DEVELOPMENT 43](#_Toc511356073)

[5.1 User Interface Development 43](#_Toc511356074)

[5.1.1 Splash Screen 43](#_Toc511356075)

[5.1.2 Login Page 44](#_Toc511356076)

[5.1.3 Home 45](#_Toc511356077)

[5.1.4 Update data 46](#_Toc511356078)

[5.1.5 Public Officer Page 47](#_Toc511356079)

[5.1.6 Widget Button 48](#_Toc511356080)

[5.2 Application Details 48](#_Toc511356081)

[5.2.1 Registration and Login Activity 49](#_Toc511356082)

[5.2.2 Google Maps 52](#_Toc511356083)

[5.2.3 Dependencies on build.gradle 53](#_Toc511356084)

[5.2.4 Location sharing 53](#_Toc511356085)

[5.2.5 Cancel a contact by user 56](#_Toc511356086)

[5.2.6 Function runs if the Officer choose completed menu 57](#_Toc511356087)

[5.2.7 Public Officer register an login 57](#_Toc511356088)

[5.2.8 Be available to be contact 61](#_Toc511356089)

[5.2.9 Checking Public Officer Connection 61](#_Toc511356090)

[5.2.10 Generate map 62](#_Toc511356091)

[5.2.11 Scan to see if there is assigned user to Public Officer 62](#_Toc511356092)

[5.2.12 Show user image (profile photo and ID photo) 64](#_Toc511356093)

[5.2.13 Show user info 65](#_Toc511356094)

[5.2.14 Public Officer available 65](#_Toc511356095)

[5.2.15 Public Officer suddenly not available in any reason 66](#_Toc511356096)

[5.2.16 Public Officer completed the request 66](#_Toc511356097)

[5.2.17 User update profile 67](#_Toc511356098)

[5.2.18 Logout 70](#_Toc511356099)

[CHAPTER VI SYSTEM TESTING 71](#_Toc511356100)

[6.1 Testing Environment 71](#_Toc511356101)

[6.2 Testing Scenario 72](#_Toc511356102)

[6.2.1 Login 72](#_Toc511356103)

[6.2.2 Register new account 72](#_Toc511356104)

[6.2.3 Test button 74](#_Toc511356105)

[6.2.4 Widget Button 75](#_Toc511356106)

[6.2.5 Cancel Menu (on homepage) 76](#_Toc511356107)

[6.2.6 Maps view (Public Officer Activity) 77](#_Toc511356108)

[6.2.7 Request done 78](#_Toc511356109)

[CHAPTER VII 80](#_Toc511356110)

[7.1 Conclusions 80](#_Toc511356111)

[7.2 Future Work 81](#_Toc511356112)

[References 83](#_Toc511356113)

# LIST OF FIGURES

[Figure 1.1 Feature Driven Development [14] 4](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356114)

[Figure 2.1 GPS System Flow [12] 9](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356115)

[Figure 2.2 Google Maps 13](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356116)

[Figure 2.3 Longitude and Latitude [13] 14](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356117)

[Figure 3.1 Use Case Diagram 27](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356118)

[Figure 3.2 Register diagram 32](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356119)

[Figure 3.3 Share location diagram 32](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356120)

[Figure 4.1 Registration and Login Page 34](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356121)

[Figure 4.2 Home Page 35](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356122)

[Figure 4.3 Maps Screen 36](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356123)

[Figure 4.4 Widget on Phone’s Home Screen 37](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356124)

[Figure 4.5 Initial State 38](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356125)

[Figure 4.6 Ambulance Child 39](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356126)

[Figure 4.7 Ektp Child with Profile Information 39](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356127)

[Figure 4.8 Generaluser child and the Data Stored 40](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356128)

[Figure 4.9 AmbulanceAvail child 40](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356129)

[Figure 4.11 AmbulanceOnRequest Child 41](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356130)

[Figure 4.10 Request child 41](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356131)

[Figure 5.1 Splash screen 44](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356132)

[Figure 5.2 Login Page 45](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356133)

[Figure 5.3 Home Page 46](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356134)

[Figure 5.4 Update data 47](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356135)

[Figure 5.5 Public Officer main activity (map) 48](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356136)

[Figure 5.6 Widget 49](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356137)

[Figure 5.7 Registration 50](#_Toc511356138)

[Figure 5.8 Copy user data to user account 51](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356139)

[Figure 5.10 Login 52](#_Toc511356140)

[Figure 5.9 Authentication check 52](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356141)

[Figure 5.11 Set Permission on Manifest.xml 53](#_Toc511356142)

[Figure 5.12 Meta-data setting 53](#_Toc511356143)

[Figure 5.13 Key on googlemaps.xml 53](#_Toc511356144)

[Figure 5.15 check location enabled and get location 54](#_Toc511356145)

[Figure 5.14 Build.gradle dependencies 54](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356146)

[Figure 5.16 scan nearby Public Officer and make contact 55](#_Toc511356147)

[Figure 5.17 Check contact status 56](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356148)

[Figure 5.18 Cancel Request by User 57](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356149)

[Figure 5.19 Request completed 58](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356150)

[Figure 5.21 Function of registering 59](#_Toc511356151)

[Figure 5.20 Login for Public Officer 59](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356152)

[Figure 5.22 Authentication check 60](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356153)

[Figure 5.23 Switch 61](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356154)

[Figure 5.24 Checking connection 61](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356155)

[Figure 5.25 generating map 62](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356156)

[Figure 5.26 Set point and call get assigned user function 62](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356157)

[Figure 5.27 get Assigned user 63](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356158)

[Figure 5.28 Show user location on map and show menu 64](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356159)

[Figure 5.29 Get Image from Firebase 64](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356160)

[Figure 5.30 Get user Information from database as a list 65](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356161)

[Figure 5.31 Function to determine where to put the Public Officer location 66](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356162)

[Figure 5.32 Public Officer not available function 67](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356163)

[Figure 5.33 Task completed 68](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356164)

[Figure 5.34 Update phone number 68](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356165)

[Figure 5.35 Add hospital record 69](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356166)

[Figure 5.36 Update profile picture 69](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356167)

[Figure 5.37 Logout 70](file:///A:\print\Thesis%20Fix%20jadi%20dah%20kelar.docx#_Toc511356168)

# LIST OF TABLES

Table 3.1 Software Functional Project 24

Table 3.2 Comparison between Google Maps, Grab, Zenly, and YUW 26

Table 3.3 Use Case Narrative for Register 28

Table 3.4 Use Case Narrative for Login 28

Table 3.5 Use Case Narrative for Share location 29

Table 3.6 Use Case Narrative for Tracking 30

Table 3.7 Use Case Narrative for Cancel Contact 31

Table 4.1 Registration Page Description 35

Table 4.2 Home page description 36

Table 4.3 Maps on page 37

Table 4.4 widget on phone’s home screen description 38

Table 4.5 Software Tool Development 41

Table 4.6 Hardware Requirements 42

Table 6.1 Device Specification 71

Table 6.2 Login scenario 72

Table 6.3 Registration scenario 73

Table 6.4 Testing scenario 74

Table 6.5 Widget scenario 75

Table 6.6 Cancel scenario 77

Table 6.7 Map scenario 77

Table 6.8 Request done scenario 79

# INTRODUCTION

## Background

Smartphone has become a part of human daily needs. Smartphones are developed to help human activities starting from education, communication, hobbies, daily activities, until business purposes. It is equipped with advance technology like camera, touch screen, GPS (Global Positioning System), and others. Among those technology, GPS usually used by people who need to use maps so the application can show the surrounding area and can direct people to certain point from their starting point. GPS can also show the distance between one point to another point, measuring speed, and estimating a trip arrival time. GPS tracking can be used by people to see another people’s position too if they allow it.

Indonesia has a crime index of 49.51 where it is a moderate category of crime compared to other countries. Cases of crime in Jakarta 2016 are recorded an acceleration compared from the previous year for 8 seconds from 12 minutes 26 seconds to 12 minutes 18 seconds. There has also been an increase in some cases of crimes such as robberies, violent, theft, rape, and other cases that undermine the security of Indonesia especially in Jakarta with the most reported crime cases[[1]](#footnote-1).

This thesis will develop an Android Application which has function to track people’s location when they click the home screen widget, so another nearby party (Police Public Officer, Firefighter, or Ambulance) can approach the user faster. This application will give the user an access to set 3 *widgets* to their home screen, so the user can choose whether they want to notify police, firefighter, or ambulance simply by only click the related widget. The nearby Public Officer will track down the user who click the widget by using the user’s real-time location*.* This application will be named “Your Urgent Widgets”.

People will need this application in order to feel safe, because they can call nearby police to track their position and approach them as soon as possible whenever they feel insecure or in danger. In addition, this application will also provide another option like widget to call ambulance for the time when they need it and will also provide a fire fighter widget for emergency matters that related to fire fighter job. This application’s goal is to prevent further damage or loss because of any danger may appears in daily life.

## Problem Statement

Usually, people are panic at certain emergency time, like when they are in hurry or racing against time. People can be panic and being blank, do not know how to do things right or even know what to do in urgent time. It is hard to look for nearby police’s or ambulances or fire fighter’s number and call them, it is even harder to describe to the Public Officer of the current situation that happened. People need to call these Public Officer a soon as possible, but cannot find the fastest way to do it although people need to multitask in order to prevent any further damage may appear. People need the fastest way to make contact to nearby Public Officer they need, is the problem to solve.

## Thesis Objective

The objective of this application is to provide the user a tool to help the user contacting the emergencies number in only one touch. The application can contact police, ambulance, or fire fighter by the widget the user click, and sending the real-time location of the user to the selected party. The aim of this application is to lower crime rate, prevent further damage of fire, and also to save time to save people life. It is easier to only tap one widget to make a contact and tell the user location on the same time, this will make it more efficient in making emergency call.

## Scope and Limitation

This thesis focused on location tracking that required a consistence coordinate of the GPS by developing these several features:

1. Displaying real-time location of transmitter (user device) in range of time,
2. Tracking the movement of user,
3. Scanning of nearby needed apparatus device to notify about the emergency call,
4. Panic Button widget.

However, this application also has some limitation, such as:

1. Cannot turn user device’s GPS on or off automatically, means that the users have to enable GPS on their phone or turn on the GPS to allow the device to be located.
2. Cannot monitor disabled phone, or turned off phone.
3. Both party (transmitter and receiver) have to enable their GPS first.
4. Limited to only 3 kind of service (police, ambulance, fire fighter).
5. Cannot make more than 1 contact in one time.

## Methodology

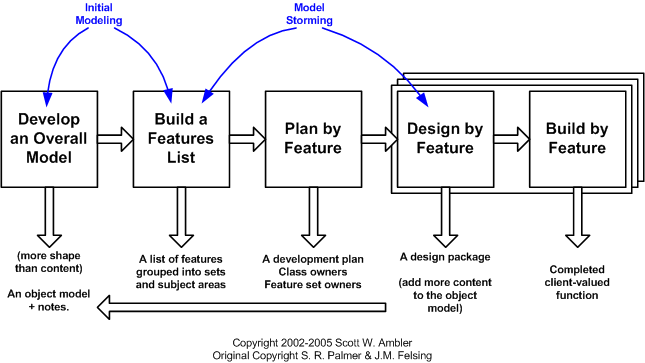
This thesis will use Feature Driven Development (FDD). FDD method accommodate requirement changes during development and, after development, adept a delivering the upgrade caused by software’s rapid software evolution and the customer’s ever-increasing requirements. FDD model reduces the development time which is really useful to save time [1].

Figure 1.1 Feature Driven Development [14]

1. **Develop an Overall Model**

This phase is purposed to make the overall image of the program will be made. In this thesis, the program will make a panic button on Android that implements home screen widget and GPS tracking.

1. **Build a Features List**

By the model image developed on the first phase, then the feature list of the software are listed. The features needed in order to make the program works as the initial goal of the program. The features that helps this panic button to works at full potential and helps the users to utilize the program.

1. **Plan by Features**

This phase will plan each of the features listed. Which feature is the main feature and which feature is the supporting feature. Then categorize each feature by the priority level of the feature.

1. **Design by Feature**

Design by feature is a design package where the feature will be designed of what it will handle and the capability of the feature.

1. **Build by Feature**

Build by feature from what the most important feature the program will have as the core feature will leads to another feature. The core feature of this thesis is the sharing location and the home screen button.

## Thesis Outline

This thesis report consist seven chapters as follows: introduction, literature study, system analysis, system design, system development, system testing, and conclusion.

1. **CHAPTER I INTRODUCTION**

The chapter consists of background, problem statement, thesis objective, scope and limitation, methodology, and thesis outline.

1. **CHAPTER II LITERATURE STUDY**

This chapter consists of theoretical supporting knowledge needed to make this application. This chapter contains explanation about the concept of Mobile Tracking like GPS, Firebase Real-time Database, Google Maps, Longitude and Latitude, and related work.

1. **CHAPTER III SYSTEM ANALYSIS**

This chapter consists of the system overview of how the system works, system requirement that consist the hardware and software required to track GPS location and develop the android application, use case diagram of the application feature, use case diagram with narrative description, and swim lane diagram.

1. **CHAPTER IV SYSTEM DESIGN**

This chapter consists of the design of the application, including its architecture, physical, and layout of the User Interfaces of the Urgent Widget application.

1. **CHAPTER V SYSTEM DEVELOPMENT**

This chapter consists of user interface development of Urgent Widget and software details including the coding of the system focusing on the GPS tracking function.

1. **CHAPTER VI SYSTEM TESTING**

This chapter consists of evaluation and testing of the system with several testing scenario from one user to share location and another user role to do the GPS tracking.

1. **CHAPTER VII CONCLUSION AND FUTURE WORK**

This chapter includes the conclusion obtained from the application testing result of the GPS tracking and notification, and possible future works.

# LITERATURE STUDY

## Global Positioning System (GPS)

The Global Positioning System (GPS) is a satellite-based navigation made up of a system of 24 satellites put into space by the U.S. Department of Defense. This system contains three parts which is satellites, ground station, and receivers. Satellites have a function like stars in constellations, the ground stations use radar to make sure they are on the position they are supposed to be, a receiver is constantly listening for a signal from these satellites. The receiver figures out how far they are from them. GPS was originally planned for military application, but in the 1980s [2], government made the system to be available for people to use. GPS works in any weather condition, anyplace in the world, for 24 hours a day every day.

### GPS System Flow

GPS satellites orbiting around the world two times a day and transmit signal information to earth. GPS receivers then take this data and use trilateration to calculate the user’s exact location. Basically, the GPS receiver compares the time a signal was transmitted by a satellite with the time it was received. The time difference tells the GPS receiver how far away the satellite is. Presently, with distance estimations from a few more satellites, the receiver can determine the user's position and show it on the unit's electronic map.

A GPS receiver must be locked on to the signal of at least 3 satellites to calculate a 2D position (latitude and longitude) and track movement [3]. With at least four satellites in view, the receiver can determine the user’s 3D position (latitude, longitude, and altitude). Once the user’s position has been determined, the GPS unit can calculate other data, such as speed, bearing, track, trip distance, distance to destination, sunrise and sunset time and more.

Figure 2.1 GPS System Flow [12]

### GPS Signal

GPS satellites transmit two low power radio signals called L1 and L2. The L1 frequency of 1575.42 MHz in the UHF band can be use by civilian. The signals travel by line of sight, which means they will pass through clouds, glass, and plastic but cannot go through most solid object like buildings and mountains [4].

A GPS signal contains 3 different bits of data: a pseudorandom code, ephemeris data, and almanac data. The pseudorandom is basically an I.D. code that refers to which satellite is transmitting data.

Ephemeris data, which is constantly transmitted by every satellite contains important data about the status of satellite (if it is healthy or unhealthy), current date and time. This part of signal is very important for determining a position.

The almanac data tells the GPS receiver the position of every GPS satellites should be at any time throughout the day. Every satellites transmits almanac data showing the orbital data for that satellite and for very other satellite in the system [3].

There are some factors that can degrade the GPS signal and affect accuracy:

1. Ionosphere and troposphere delays – The satellites signal slow as it goes through the atmosphere. The GPS system uses a built-in model that calculates an average amount of delay to partially correct for this type of error [5].
2. Signal multipath – This happen when the GPS is reflected, for example to tall structure or large rock surface before it reaches the receiver. This expands the travel of the signal, which causing errors.
3. Orbital errors – known as ephemeris errors, these are inaccuracies of satellite’s reported location.
4. Number of satellites visible – The more satellites a GPS receiver can “see”, the better the precision. Structures, terrain, electronic interference, or sometimes even dense foliage can block signal reception, which leads to position errors and possible no positions perusing by any means. GPS unit ordinarily will not work inside, underwater or underground.
5. Satellite geometry or shading – This refers to the relative position of the satellites at any time. Ideal satellite geometry exist when the satellites are located at wide points with respect to each other. Poor geometry result when the satellites are located in a line or in a tight grouping.
6. Intentional degradation of the satellite signal – Selective Availability (SA) is an intentional degradation of the signal once forced by the U.S. Department of Defense. Selective Availability was intended to prevent enemy military from using the highly accurate GPS signals. The government turned off SA in May 2000, which significantly improve the accuracy of civilian GPS receivers.

### GPS Accuracy

GPS needs these five components to be accurate:

* + - 1. Appropriate installation
      2. The level of technology used in the receiver
      3. The number and location of satellites
      4. Errors introduces by Selective Availability (SA), atmospheric condition, the troposphere, the ionosphere, and multi pathing radio signal bouncing off objects in the area
      5. Differential correction

## Google Maps

Google Maps is a Web-based service that has function of providing detailed information about geographical regions around the world, as can be seen at Figure 2.5. Besides the conventional road maps, Google Maps also provide aerial and satellite perspective of many spots. In some area, Google Maps also offers street views with photograph taken from vehicles.

Services that Google Maps offers:

1. A route planner to shows direction for drivers, bikers, walkers, and user of public transportation who want to travel from one specific location to another location.
2. Google Maps application program interface (API) helps website administrators to embed Google Maps into a proprietary site. Google Maps APIs are available for Android, iOS, web browser and via HTTP we services.
3. Google Maps for Mobile provides a location service for driver that uses the GPS location of the mobile device along with data from wireless and cellular networks.
4. Google Street View enables users to view and navigate through horizontal and vertical panoramic street level images of various cities around the world.

## Latitude and Longitude

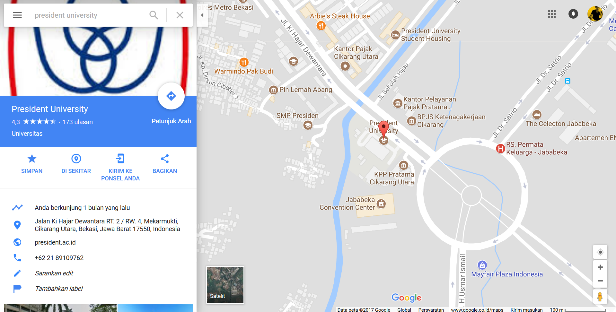
Geographic coordinate system has coordinates that represented by units called Latitude and Longitude. Every single point on the surface of the earth can be determined by the latitude and longitude coordinates, means that latitude and longitude can be used to determine virtually any point on earth.

Figure 2.2 Google Maps

Latitude is the angular distance, in degrees, minutes, and seconds of a point north or south of the Equator (horizontal line in Fig 2.6). Lines of latitude are often called parallels.

Longitude is the angular distance, in degrees, minutes, and seconds, of a point east or west of the Prime (Greenwich) Meridian (Vertical line in Fig 2.6). Lines of longitude can also be called meridians [6].

For precision purpose, degrees (°) of longitude and latitude have been divided into minutes (‘) and seconds (“). There are 60 minutes in each degree and each minute is divided into 60 seconds, and second can be divided further to tenth, hundredths, and even thousandths. Longitude angles can extend up to +180 degrees (180 degrees east) and down to -180 degrees (180 degrees west). The +180 degrees and -180 degrees longitude meridians coincide directly opposite the prime meridian [7].

## Mobile Phone Tracking

Figure 2.3 Longitude and Latitude [13]

Mobile phone tracking is the determining of the position or location of a mobile phone, whether stay on the point or even moving, according to Wikipedia. Localization can happen either by means of multilateration (a route method in view of the estimation of the distinction in separation to two station at known area that communicate signals at known times) of radio signal between (a few) cell towers of the network and the phone, or simply via GPS. To locate mobile phone using multilateration of radio signal, it must emit at least the roaming signal to contact the next nearby antenna tower, but the process does not require an active call. The Global System for Mobile Communication (GSM) is based on the phone’s signal strength to nearby antenna masts.

### Tracking Mobile Phone Location Technique

To determine the location of a mobile phone, there are some technique, such as:

* **Network-based**

The location of a mobile phone can be determined using the service provider’s network infrastructure. The advantage of network-based techniques, from a service provider’s point of view, is that they can implement non-intrusively without affecting handsets.

* **Handset-based**

The location of a mobile phone can be determined using client software installed on the handset. This technique determines the location of the handset by putting its location by cell identification, signal strengths of the home and neighboring cells, which is continuously sent to the carrier. In addition, if the handset is also equipped with GPS then significantly more precise location information can be then form the handset to the carrier.

* **SIM-based**

Using the subscriber identity module (SIM) in GSM and Universal Mobile Telecommunication System (UMTS) handset, it is possible to obtain raw radio measurement from the handset. The type of information obtained via the SIM can differ from that which is available from the handset. For example, it may be possible to obtain any ray measurement from the handset directly, yet still obtain measurement via SIM.

* **Wi-Fi**

Wi-Fi data can also be used to identify a handset’s location. Poor performance of the GPS-based methods in indoor environment and increasing popularity of Wi-Fi have encourages companies to design new and feasible methods to carry out Wi-Fi-based indoor positioning.

* **Hybrid**

Hybrid positioning system use a combination of network-based and handset-based technologies for location determination.

## Firebase

Firebase is a cloud service provider developed by Google as well as a backend business that allows users to obtain organized data for mobile apps. User verification and data updates are important aspects that almost all mobile apps need and this is easier to develop since Firebase provide those development services [8]. Firebase is easy to use and allows quick reading and writing of data even for beginners Firebase can be used to build iOS, Android and even web- based applications with real-time data and storage and makes a variety of other products that application developers can utilize.

Firebase started from Envolve, a startup founded in 2011 by James Tamplin and Andrew Lee. Envolve provided developers an API that enables the integration of online chat functionality into their websites. In October 2014, Firebase was acquired by Google. In October 2015, Google acquired Divshot to merge it with the Firebase team. Since the acquisition, Firebase has grown inside Google and expanded their services to become a unified platform for mobile developers. Firebase now integrates with various other Google services to offer broader products and scale for developers. Firebase would be launched Cloud Firestore, a Document Database, in October 2017.

### Firebase Open Source Project

One of firebase work is an open source project so the developers can use these function on their project.

#### Firepad

Firepad is an [open source](https://en.wikipedia.org/wiki/Open_source) [collaborative real-time editor](https://en.wikipedia.org/wiki/Collaborative_real-time_editor). Released under the MIT License, Firepad is used by several editors, including the [Atlassian](https://en.wikipedia.org/wiki/Atlassian) Stash Real-time Editor and [Koding](https://en.wikipedia.org/wiki/Koding) [[2]](#footnote-2).

#### Firechat

Firechat is an open source real-time chat application. It is released under the MIT License[[3]](#footnote-3).

#### GeoFire

GeoFire is an open source library that makes use of the Firebase real-time database, allowing app developers to store and query a set of keys based on geographic location[[4]](#footnote-4).

### Firebase Development

Firebase develops some services for developer to utilize so developer can use one cloud for some purpose.

#### Firebase Cloud Messaging

Formerly known as [Google Cloud Messaging](https://en.wikipedia.org/wiki/Google_Cloud_Messaging) (GCM), [Firebase Cloud Messaging](https://en.wikipedia.org/wiki/Firebase_Cloud_Messaging) (FCM) is a cross-platform solution for messages and notifications for [Android](https://en.wikipedia.org/wiki/Android_(operating_system)), [iOS](https://en.wikipedia.org/wiki/IOS), and [web applications](https://en.wikipedia.org/wiki/Web_application), which currently can be used at no cost [9].

#### Firebase Authentication

Firebase Authentication is a service that can authenticate users using only client-side code. It supports [social login providers](https://en.wikipedia.org/wiki/Social_login) Facebook, GitHub, Twitter and Google (and [Google Play Games](https://en.wikipedia.org/wiki/Google_Play_Games)). Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase [9].

#### Real-time Database

Firebase provides a real-time database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. The company provides client libraries that enable integration with [Android](https://en.wikipedia.org/wiki/Android_(operating_system)), [iOS](https://en.wikipedia.org/wiki/IOS), [JavaScript](https://en.wikipedia.org/wiki/JavaScript), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [Objective-C](https://en.wikipedia.org/wiki/Objective-C), [swift](https://en.wikipedia.org/wiki/Swift_(programming_language)) and [Node.js](https://en.wikipedia.org/wiki/Node.js) applications. The database is also accessible through a REST API and bindings for several [JavaScript frameworks](https://en.wikipedia.org/wiki/JavaScript_frameworks) such as [AngularJS](https://en.wikipedia.org/wiki/AngularJS), [React](https://en.wikipedia.org/wiki/React_(JavaScript_library)), [Ember.js](https://en.wikipedia.org/wiki/Ember.js) and [Backbone.js](https://en.wikipedia.org/wiki/Backbone.js). The REST API uses the [Server-Sent Events](https://en.wikipedia.org/wiki/Server-sent_events) protocol, which is an API for creating HTTP connections for receiving push notifications from a server. Developers using the real-time database can secure their data by using the company's server-side-enforced security rules. Cloud Firestore which is Firebase's next generation of the Real-time Database was released for beta use [9].

#### Firebase Storage

Firebase Storage provides secure file uploads and downloads for Firebase apps, regardless of network quality. The developer can use it to store images, audio, video, or other user-generated content. Firebase Storage is backed by Google Cloud Storage.

#### Firebase Hosting

Firebase Hosting is a static and dynamic [web hosting service](https://en.wikipedia.org/wiki/Web_hosting_service) that launched on May 13, 2014. It supports hosting static files such as [CSS](https://en.wikipedia.org/wiki/Cascading_Style_Sheets), [HTML](https://en.wikipedia.org/wiki/HTML), [JavaScript](https://en.wikipedia.org/wiki/JavaScript) and other files, as well as [dynamic Node.js support through Cloud Functions](https://firebase.google.com/docs/hosting/functions). The service delivers files over a [content delivery network](https://en.wikipedia.org/wiki/Content_delivery_network) (CDN) through [HTTP Secure](https://en.wikipedia.org/wiki/HTTP_Secure) (HTTPS) and [Secure Sockets Layer](https://en.wikipedia.org/wiki/Secure_Sockets_Layer) encryption (SSL). Firebase partners with Fastly, a CDN, to provide the CDN backing Firebase Hosting. The company states that Firebase Hosting grew out of customer requests; developers were using Firebase for its real-time database but needed a place to host their content [9].

## Test Methodologies

Two general approaches on doing software testing. There are black box testing and white box testing [10].

### Black Box Testing

Using black box testing to test the whether the output of a process is as expected from the input. Also known as Behavioral Testing, this testing method named so because in the perspective of the tester, it is like a black box where tester does not know what happened inside the process, purposely to know whether the software works properly or not. The error categories divided to [11]:

* Incorrect or missing functions
* Interface errors
* Errors in data structures or external database access
* Behavior or performance errors
* Initialization and termination errors

### White Box Testing

Using white box testing will done on the basis of infrastructure as defined by requirement, design, coding standards, and guidelines. White box testing main purpose is to ensure that the software is built correctly [10].

## Related Work

This section of chapter 2 will discuss about some existing application that can be relate to this thesis, will have some nearly similar function since some also use GPS, which will also use the GPS function but have different feature.

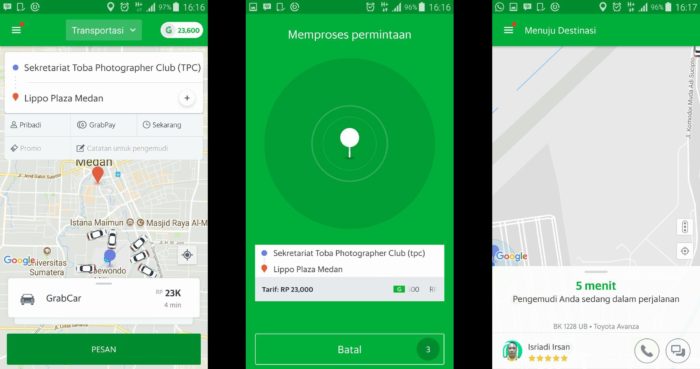
### Grab

Grab is a technology company that offers vast range of ride-hailing and logistic services through Grab application in Southeast Asia, including Indonesia. Grab developed an easy way to find a ride using GPS to track and scan nearby vehicles which are available to take the order.

The similarity of grab and this thesis is, it can scan nearby user when needed. For grab, it means scanning for nearby available driver, while this thesis scan for nearby available police, ambulance, or fire fighter. Similarity also that both send notification to nearby party and if the request accepted, the party can get the user location.

### Zenly

Using GPS technology to determine user location and share it to fellow Zenly friends, Zenly also have similar feature with this thesis, which is sharing user location to other party. This thesis also sharing the location of the related user to related party which help them to monitor the movement of user.



**Figure 2.7 Grab Application Interface**

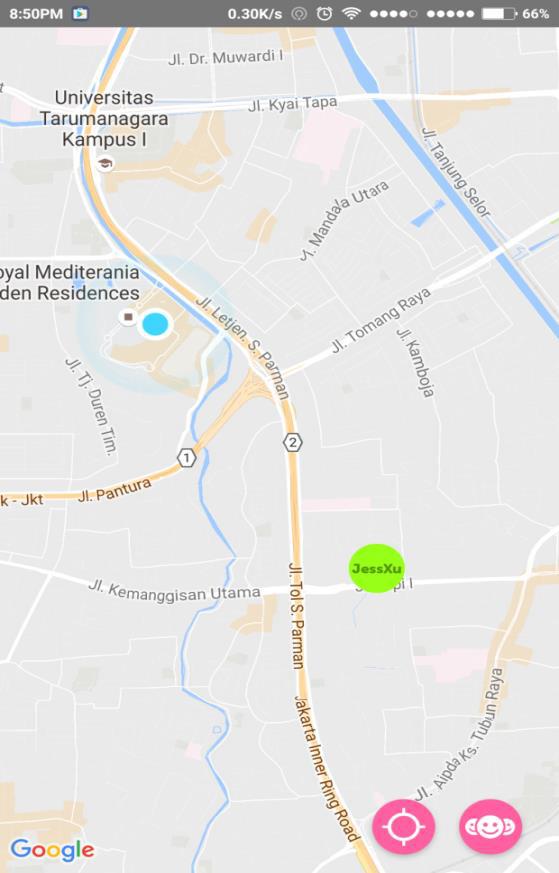
****

Figure 2.8 Zenly Interface

# SYSTEM ANALYSIS

System Analysis explain about the analysis of the application system from its function and utility which has to fit the requirement. This chapter covers of the System Overview, Comparison Overview, Application Requirement, Functional Analysis, Non-functional Analysis, Use Case Diagram, and Use Case Narrative, and Swim Lane Diagram.

## System Overview

This thesis is intended to help user to feel safe anywhere anytime. The application allows the user to notify nearby police and send the police their location if the user feel like they are insecure or being in danger. Other than just being able to call police, there are also two more option offered by this application, to notify nearby hospital for ambulance and also to notify nearby fire fighter depend on what you need at the time.

## Application Requirement

### Functional Requirement

There are some requirements that should be fulfilled for this application to be categorized as success application. These are the expected capability and core function of the project to be done. The functional requirements of this project are as shown in Table 3.1

Table 3.1 Software Functional Project

|  |  |
| --- | --- |
| **No** | **Functional Requirements** |
| 1 | Can create 3 widgets at home screen |
| 2 | Can track the user’s mobile location |
| 3 | Can send the user location to another party in real-time |
| 4 | Can scan for nearby Public Officer that the user choose to contact |
| 5 | Can cancel the contact |

### System Requirements

Development of this application require some hardware and software in order to be finished and run well. Hardware needed to make this application are:

1. Laptop / Personal Computer

To develop the application, a laptop or personal computer is a must, because this hardware is needed for coding step.

1. Smartphone

Smartphone is the media to run the application after the development step. Smartphone needed has to be android because the application is android application, and must have the GPS feature in it.

Software needed are:

1. Android Studio

Developed by Google for Android platform developer, Android Studio is the official integrated development environment (IDE). Android Studio use Java as the based programming language. This software is simple to use and has smart text editor in it.

1. Google Maps API

To display maps, this application need Google Maps API to use in Android Platform. Google Maps API is an Application Programming Interface (API) that Google provide in order to display maps in several platforms

1. Database

Firebase Database will be used to make the database and the further development of the application that later will also use database to store the data.

1. Web Browser

Web browser will be the media to check the firebase database. Since firebase is the real-time database chosen to be the database Public Officery this application, it need to be display on a media to help the development.

1. Android Emulator

Android emulator like Nox or Genymotion will be used to test the application interface and other function before the GPS test on mobile.

## Comparison Overview

Table 3.2 is the feature comparison between related work from chapter 2 and this thesis application.

Table 3.2 Comparison between Google Maps, Grab, Zenly, and YUW

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Features | Google Maps | Grab | Zenly | Your Urgent Widgets |
| Registration using citizenship ID (KTP) | ✗ | ✗ | ✗ | ✓ |
| Navigation Map | ✓ | ✓ | ✗ | ✓ |
| Scan nearby area | ✓ | ✓ | ✗ | ✓ |
| Share location to police/ambulance/firefighter | ✗ | ✗ | ✗ | ✓ |
| Widget button on home screen to instantly share location | ✗ | ✗ | ✗ | ✓ |

## Use Case Diagram

Use case diagram is a diagram of the interaction between users and the system. The key elements are the user as actor and the process called the use case. The purpose of use case diagram aims to give a picture of the point of view of the system and also to describe the requirements. On the other hand, use case narrative will give the description about the functionalities.

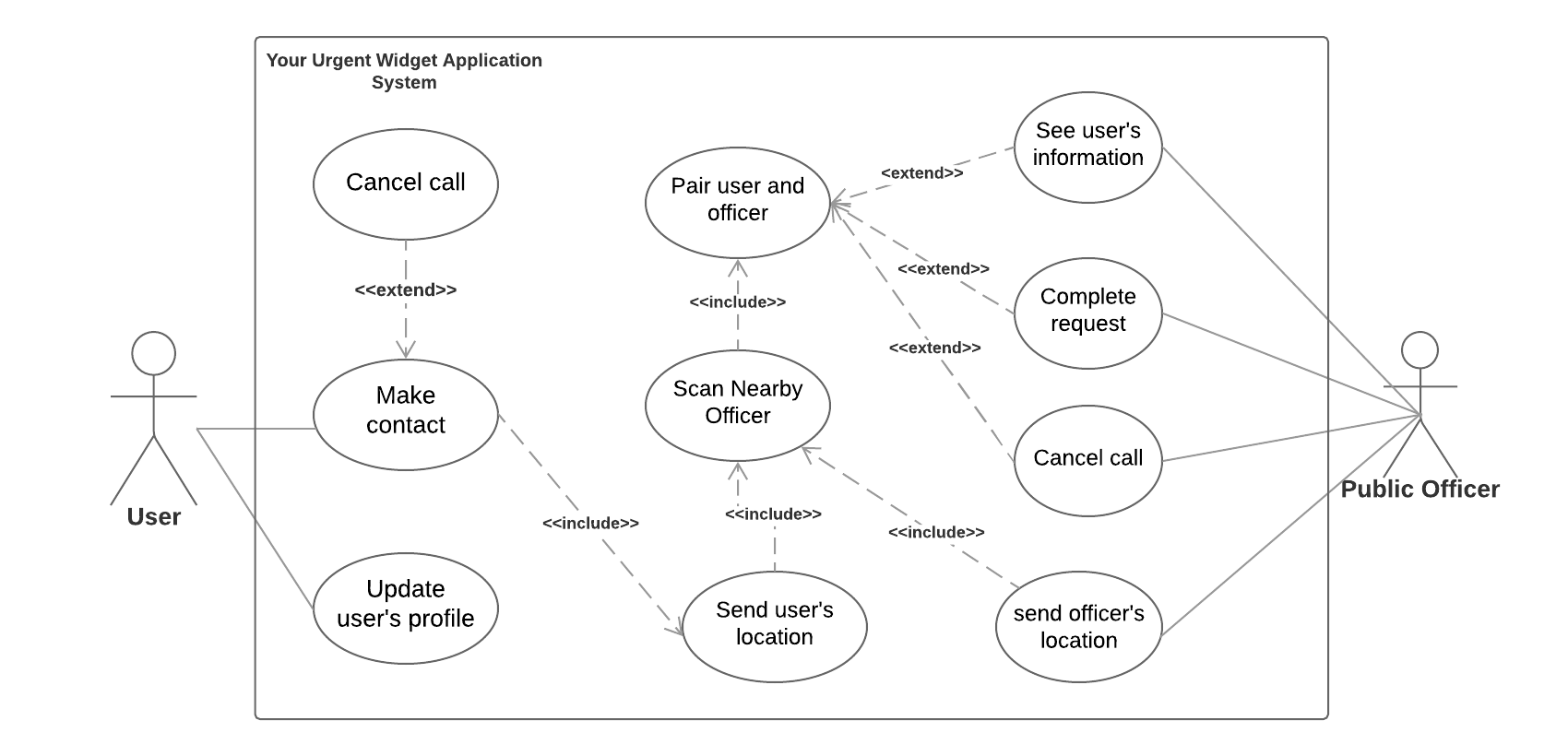
The actor or the one who is having interaction in this application is the user. The use case diagram for Urgent Widgets is shown in Figure 3.1.

Figure 3.1 Use Case Diagram

## Use Case Narrative

Use case diagram narrative is a narration to describe the interaction between actors and system in a form of table. The description of interaction will be describe based on functional or behavior point of view such as name, description case, precondition, trigger, conclusion, post condition, and business rule how to complete the process requirement. The purpose of use case narrative is to help indicating and minimalize any possible misunderstanding from the beginning. Since there are more than 1 type of user, the users will be divided into user and Public Officer (as the Public Officers).

Table 3.3 Use Case Narrative for Register

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Register | |
| **Use Case ID:** | 1 | |
| **Priority:** | High | |
| **Primary Business Actor:** | user | |
| **Primary System Actor:** | System | |
| **Description:** | User register to system with user’s citizenship ID number, email and password | |
| **Precondition:** | Open the application | |
| **Trigger:** | User opens the application | |
| **Typical Course of Events:** | **Actor Action** | **System Response** |
| **Step 1:** User open the application | **Step 2:** The system provide Login / Registration form |
| **Step 3:** User fill the registration form and input their ID | **Step 4:** The system validate whether the ID is valid (exist in the citizen database) or not |
| **Alternate Courses** | ID is not valid (not listed in the database) or ID has been used | |
| **Post condition:** | Application will go to homepage. | |
| **Implementation Constraints and Specifications:** | None | |

Table 3.4 Use Case Narrative for Login

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Login | |
| **Use Case ID:** | 2 | |
| **Priority:** | High | |
| **Primary Business Actor:** | user | |
| **Primary System Actor:** | System | |
| **Description:** | User login using ID, email, and password | |
| **Precondition:** | User has already registered before | |
| **Trigger:** | User choose sign in on registry screen | |
| **Typical Course of Events:** | **Actor Action** | **System Response** |
| **Step 1:** User input ID and password | **Step 2:** system check if ID is already registered and the password match |
| **Alternate Courses** | ID is not registered yet or the password does not match | |
| **Post condition:** | Home page of the application | |
| **Implementation Constraints and Specifications:** | None | |

Table 3.5 Use Case Narrative for Share location

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Share Location | |
| **Use Case ID:** | 3 | |
| **Priority:** | High | |
| **Primary Business Actor:** | user and Public Officer | |
| **Primary System Actor:** | System | |
| **Description:** | user share their location and notify nearby Public Officer that they need at the time | |
| **Precondition:** | The GPS location has already activated by the user | |
| **Trigger:** | User touch the widget button according to their emergency situation | |
| **Typical Course of Events:** | **Actor Action** | **System Response** |
| **Step 1:** user click the button | **Step 2:** system read the user’s location |
|  | **Step 3:** system scan for nearby Public Officer |
|  | **Step 4:** system send the location of userto nearby Public Officer |
| **Alternate Courses** | GPS not active or the signal is not strong enough | |
| **Post condition:** | Public Officer get the notification and rush to the user location | |
| **Implementation Constraints and Specifications:** | None | |

Table 3.6 Use Case Narrative for Tracking

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Tracking | |
| **Use Case ID:** | 4 | |
| **Priority:** | High | |
| **Primary Business Actor:** | Public Officer | |
| **Primary System Actor:** | System | |
| **Description:** | Public Officer (the Public Officer) track the user’s location | |
| **Precondition:** | Public Officer got the location | |
| **Trigger:** | Public Officer open the application and set the switch to on duty | |
| **Typical Course of Events:** | **Actor Action** | **System Response** |
| **Step 1:** Opened the application | **Step 2:** system shows Public Officer the user’s real time location |
| **Alternate Courses** | Poor GPS signal or user cancel the contact | |
| **Post condition:** | Public Officer got the location of user in real-time | |
| **Implementation Constraints and Specifications:** | None | |

Table 3.7 Use Case Narrative for Cancel Contact

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Cancel contact | |
| **Use Case ID:** | 5 | |
| **Priority:** | Medium | |
| **Primary Business Actor:** | user | |
| **Primary System Actor:** | System | |
| **Description:** | user already feel safe or need no Public Officer again | |
| **Precondition:** | user feels safe and no longer need Public Officer to come | |
| **Trigger:** | user click cancel on the menu | |
| **Typical Course of Events:** | **Actor Action** | **System Response** |
| **Step 1:** user open the application and choose cancel call from menu | **Step 2:** System proceed to stop sharing user A’s real-time location |
| **Alternate Courses** | poor internet signal | |
| **Post condition:** | Public Officer no longer get the real time location of user A | |
| **Implementation Constraints and Specifications:** | None | |

## Activity Diagram

Activity Diagram is a diagram to describe a business process or a software workflow to make it easier to understand. Through Activity Diagram, it is intended to show all flow of control of this thesis application. Activity diagrams may be regarded as a form of flowchart. Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

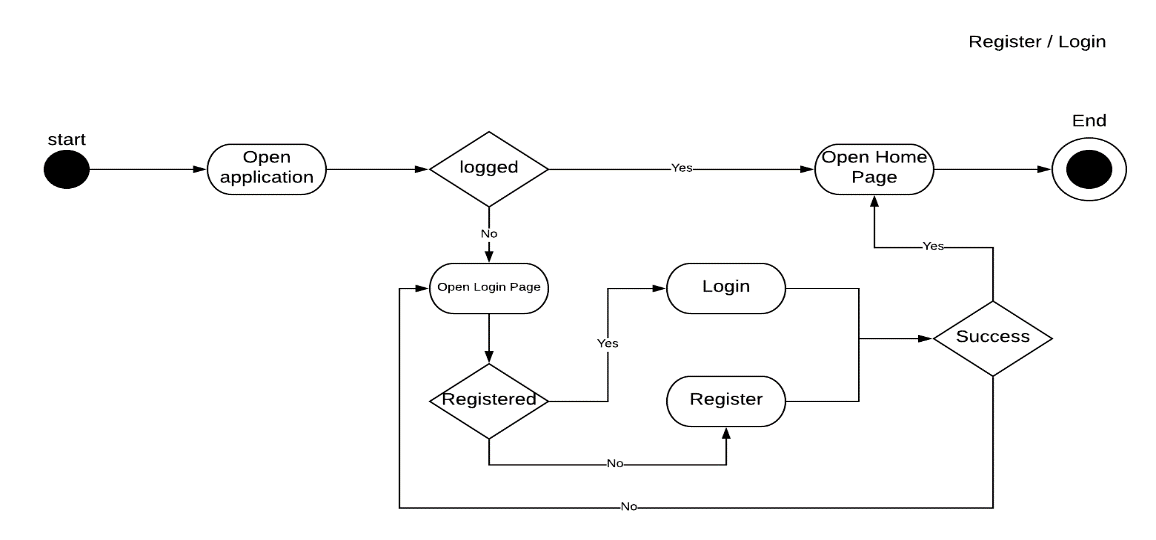
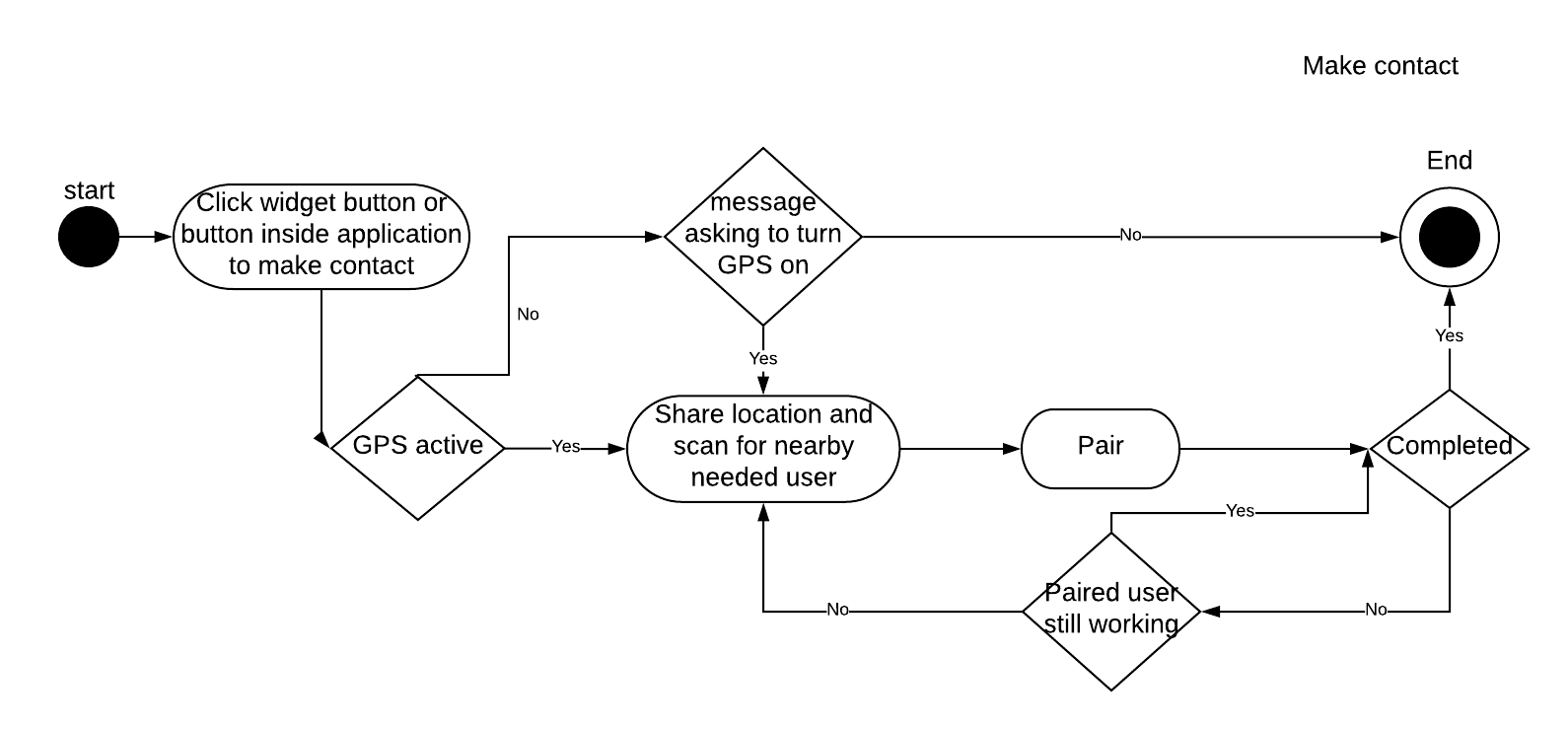
There are 2 activity diagram for this application.

Figure 3.2 Register diagram

Figure 3.3 Share location diagram

# SYSTEM DESIGN

System design is the development process of design of the application that meets the requirement from system analysis. System Design gives the description about the visualization of the application. Divided into 3 section of process, system design covers the process of user interface design, physical design, class diagram, and also database diagram.

The user interface design describe each component of the application layout and explain the use of each component. User interface’s purpose is to improve the user experience and make the interaction between system and Public Officer become efficient and simple as the user easily understand the use of each function and not confused of the system.

The physical design later contains the information of the software and hardware that the application required. All about the required component specification and tools will be describe in this phase.

Database diagram will explain all the relationship of database table that this application need. The database schema will also show the primary key and the foreign key of the tables. Class diagram shows the relation of the classes and function that the application require when running.

## User Interface Design

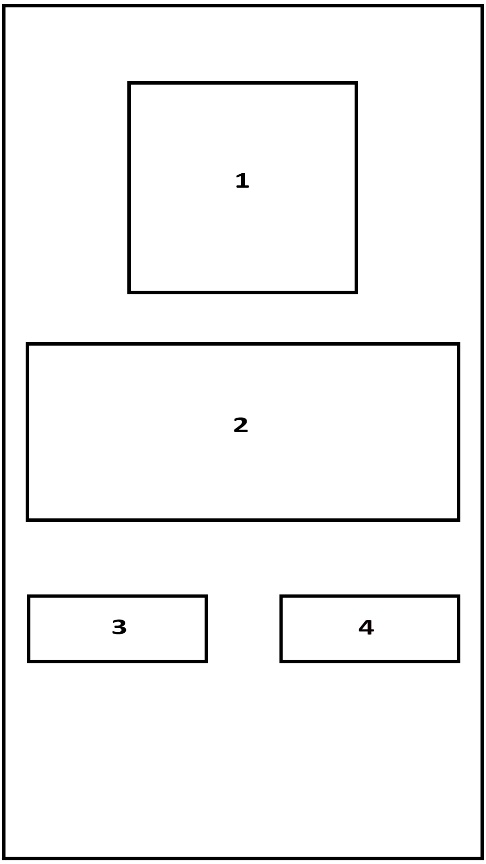
The user interface of this application will be simple because the main function of this application is on the widget button, not inside the application. The user interface layout of the application will be as shown in the figure 4.1 for the registration form, figure 4.2 for the login page, and figure 4.3 for the homepage of the application, while figure 4.4 shows the interface of the widget on the user home screen.

Figure 4.1 Registration and Login Page

Figure 4.1 shows the application layout design for the registration and login page. For user A, the form consist of 3 textbox and for Public Officer the form consist of 1 dropdown and 3 textbox. There are 2 buttons, one login, and one for registration. The layout is explained on Table 4.1.

Table 4.1 Registration Page Description

|  |  |
| --- | --- |
| 1 | Application logo |
| 2 | Registration and login form (email, citizenship id, and password) |
| 3 | Submit button |
| 4 | To login page button |

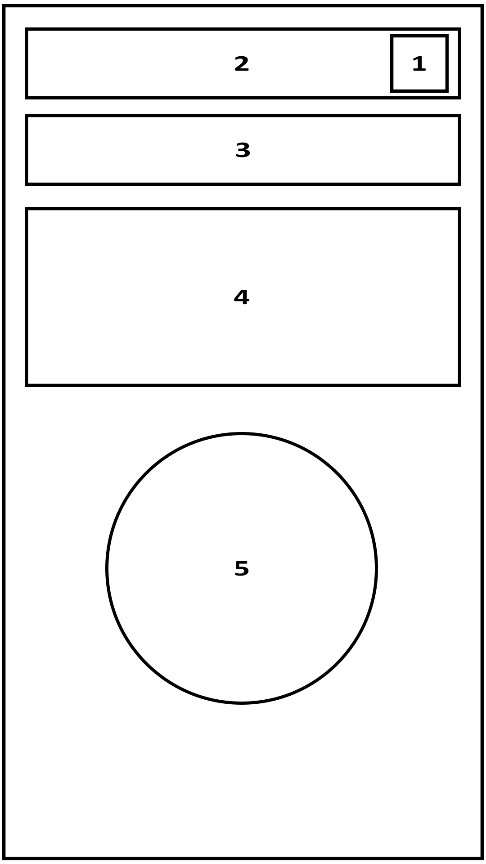
****After the user registered or after the user login, the application shows the home page. As shown on figure 4.3, there are only one urgent button and one menu bar.

Figure 4.2 Home Page

Table 4.2 Home page description

|  |  |
| --- | --- |
| 1 | Menu |
| 2 | Menu bar |
| 3 | tab bar |
| 4 | Status information |
| 5 | Button |

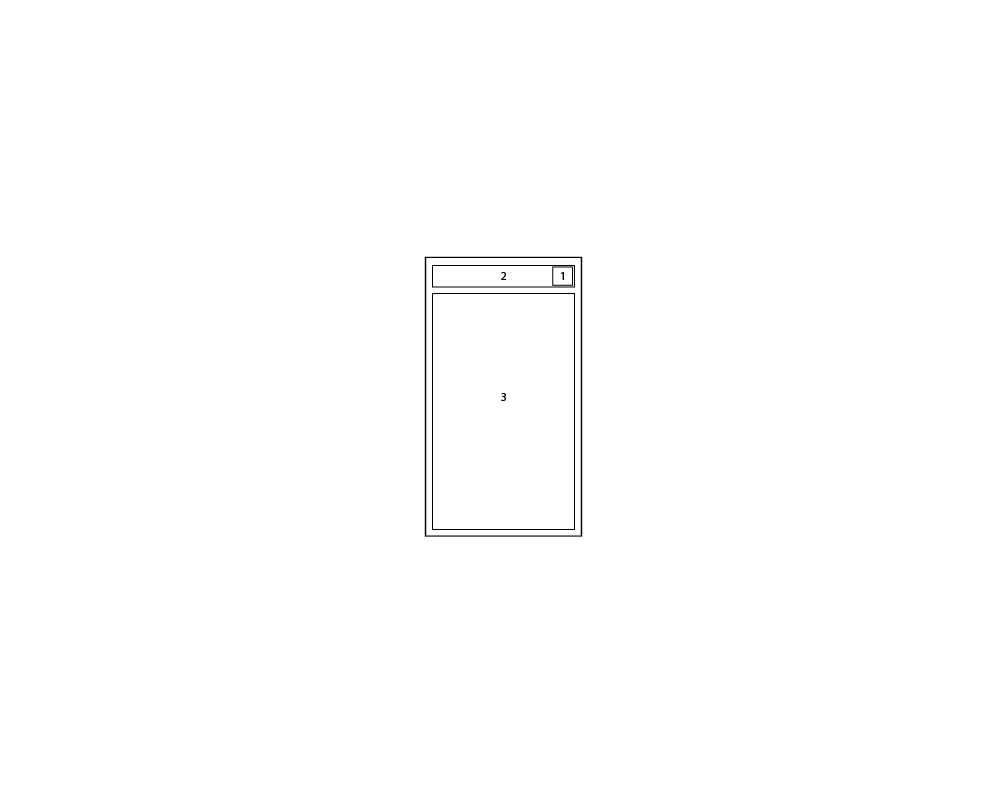
****

Figure 4.3 Maps Screen

Figure 4.4, shows the screen which containing the maps screen. The menu button can show user’s information and complete request.

Table 4.3 Maps on page

|  |  |
| --- | --- |
| 1 | Switch |
| 2 | Menu bar |
| 3 | Maps |

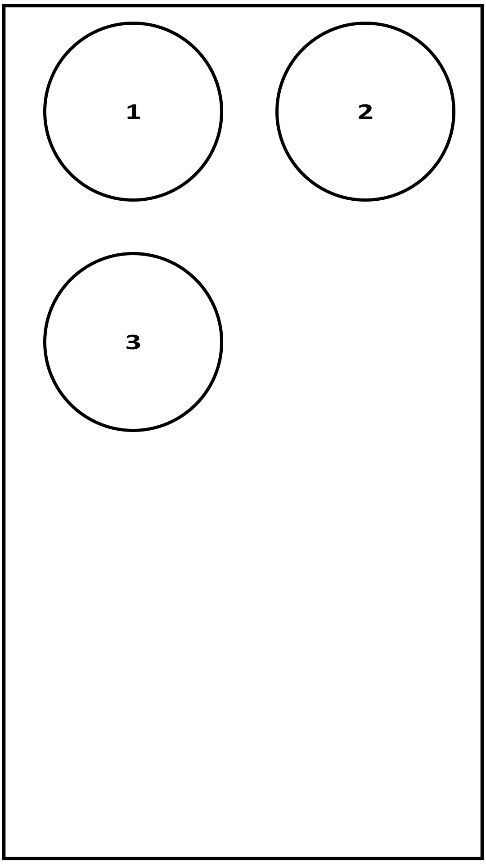


Figure 4.4 Widget on Phone’s Home Screen

In figure 4.4, there are three buttons, actually the arrangement is on the user and not set by the application. So the user is free to arrange how they want the button to be at or which widget button they want on home screen. For example, the arrangement can be like as shown in figure 4.4 and describe on table 4.4.

Table 4.4 widget on phone’s home screen description

|  |  |
| --- | --- |
| 1 | Urgent button to notify Police |
| 2 | Urgent button to notify Ambulance |
| 3 | Urgent button to notify Fire Fighter |

## Database Structure Design

The database structure design on Firebase Real-time Database is in a form of tree notes. The tree notes will have their own childs as each of tree notes have their own purpose in storing data.

### Initial state

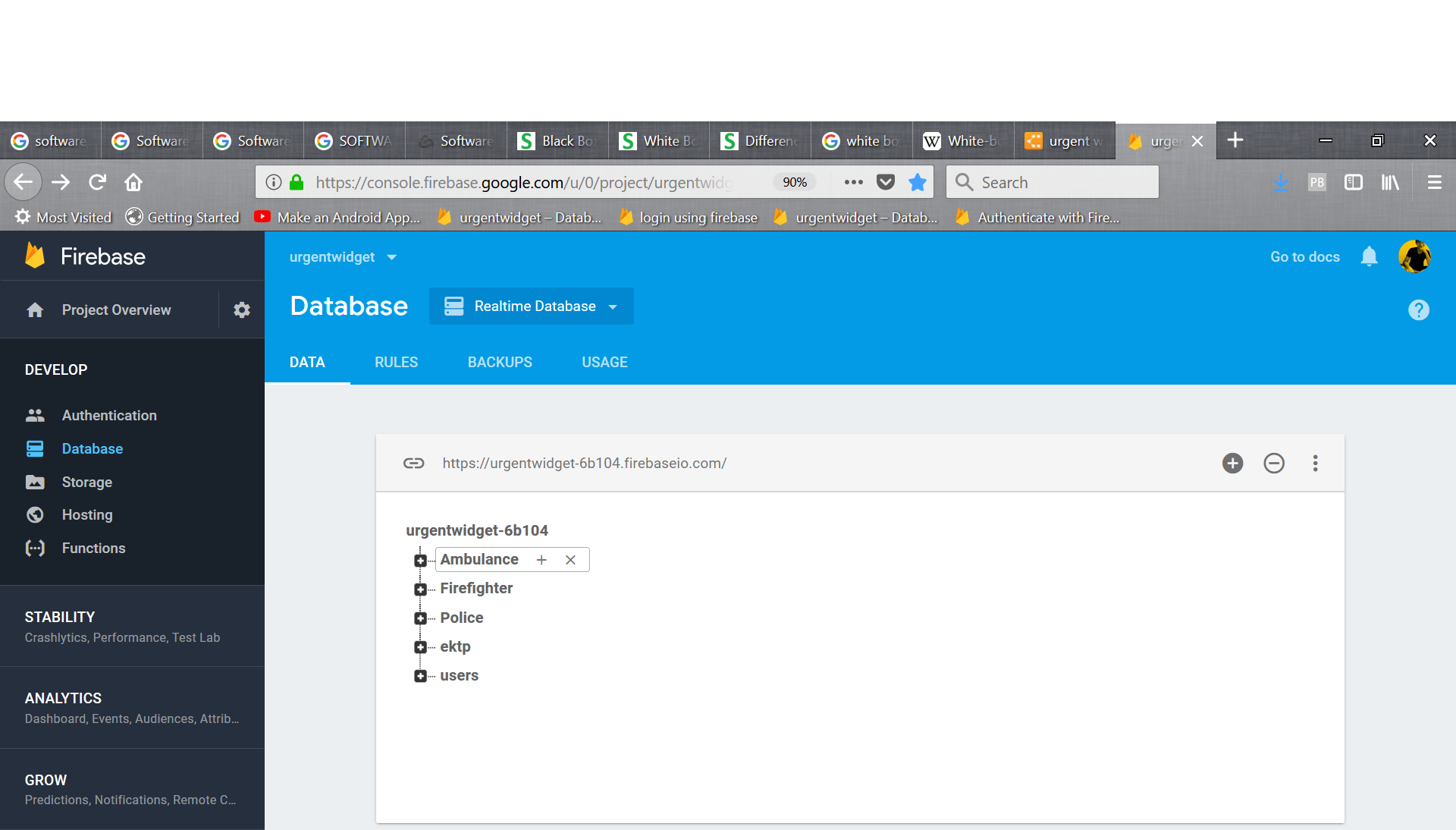
On the initial state, the data on the tree notes as shown in Figure 4.5 have ambulance, firefighter, and police child to store the officer’s work id to verify on login phase, and ektp child to store citizenship id to verify user on login phase, users child is to store the registered account.

Figure 4.5 Initial State

#### Officer Child

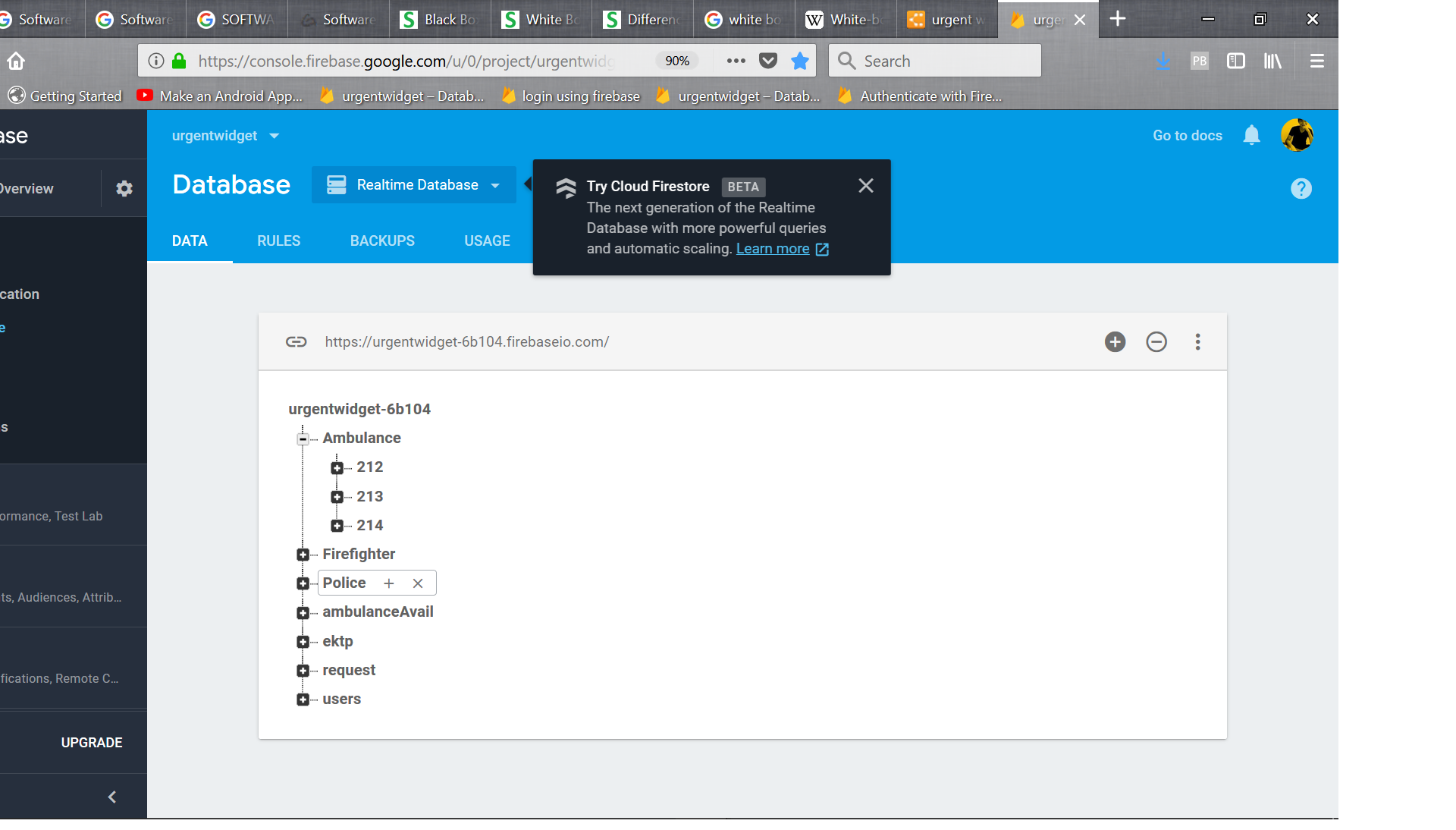
These child made as a dummy data that have a function to store the work id of officers, for example the work id of ambulances as shown on Figure 4.6.

Figure 4.6 Ambulance Child

#### Ektp Child

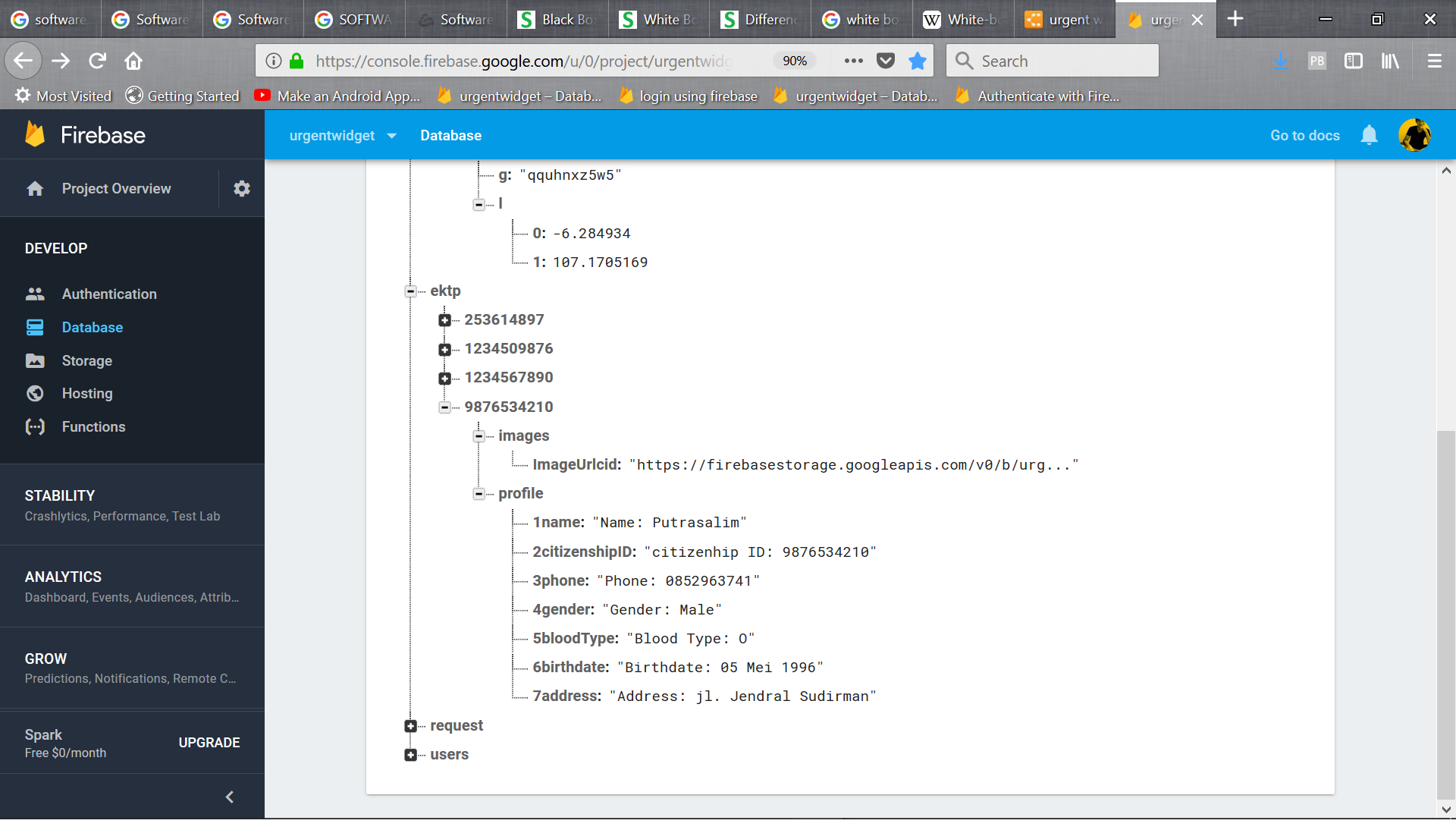
Ektp child has citizenship ids that each contains image url and user’s profile information.

Figure 4.7 Ektp Child with Profile Information

#### Users child

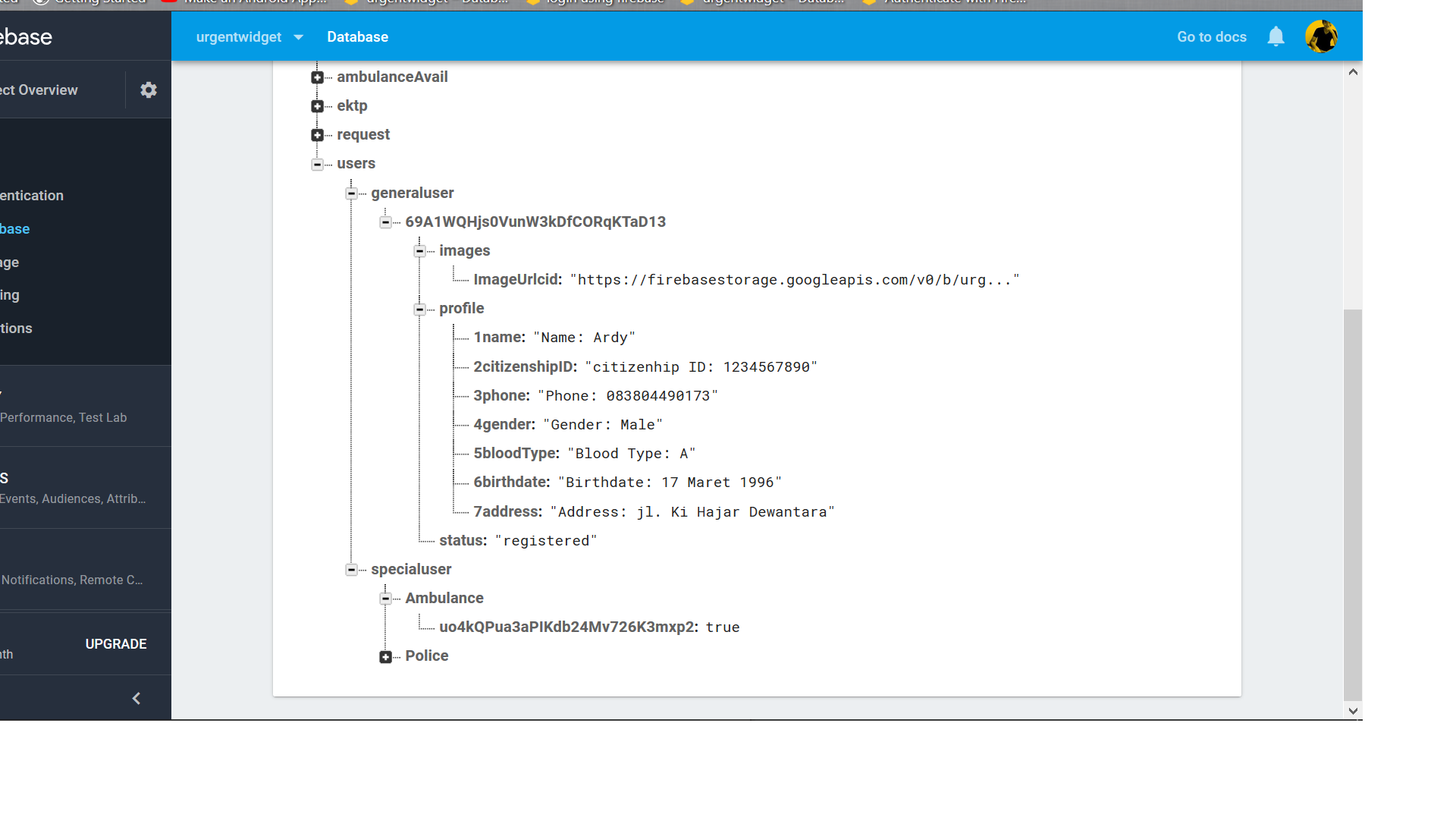
Users child is to store the data of users who registered on the application. Users child has 2 child which are generaluser (as shown in Figure 4.8) and specialuser. Can be seen that the data inside the user id is a copy of the dummy data inside ektp.

Figure 4.8 Generaluser child and the Data Stored

### Officer Available state

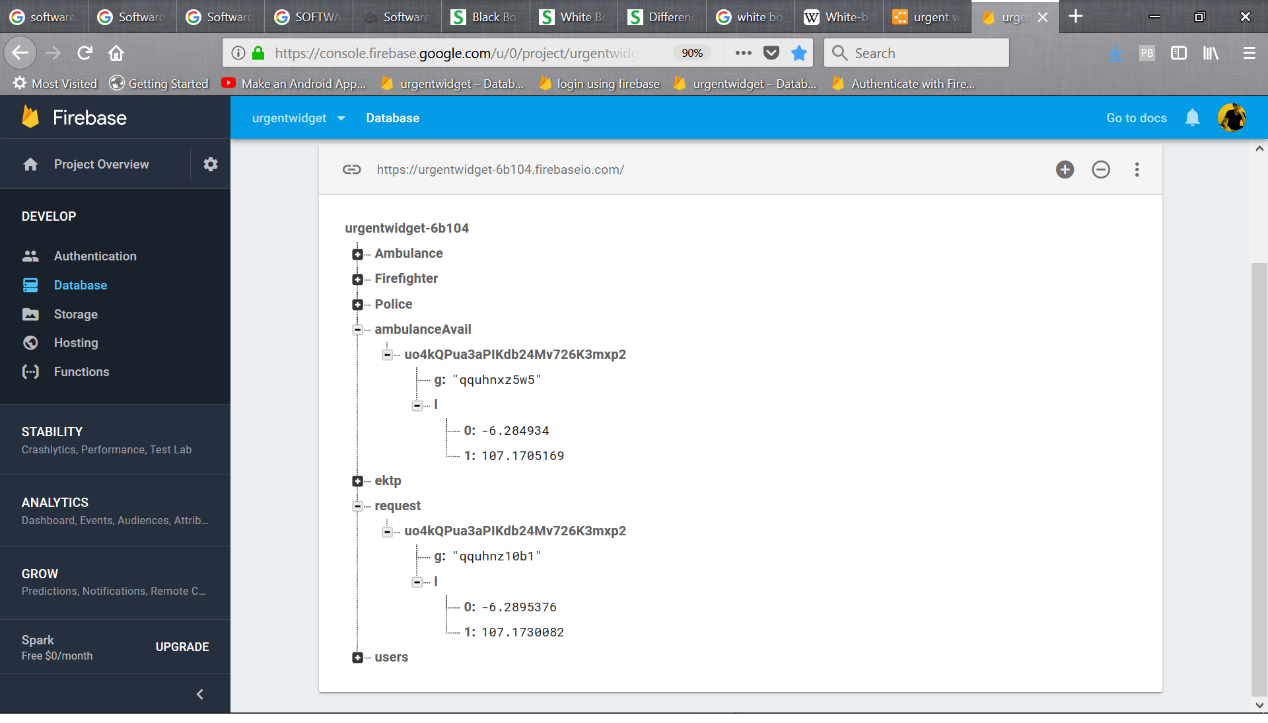
Officer can switch between available and not available with a switch button. When the officer switch to available state, then system will put their user id and store their real-time location to their related work available child. (Figure 4.9 shows ambulanceavail child)

Figure 4.9 AmbulanceAvail child

### Request State

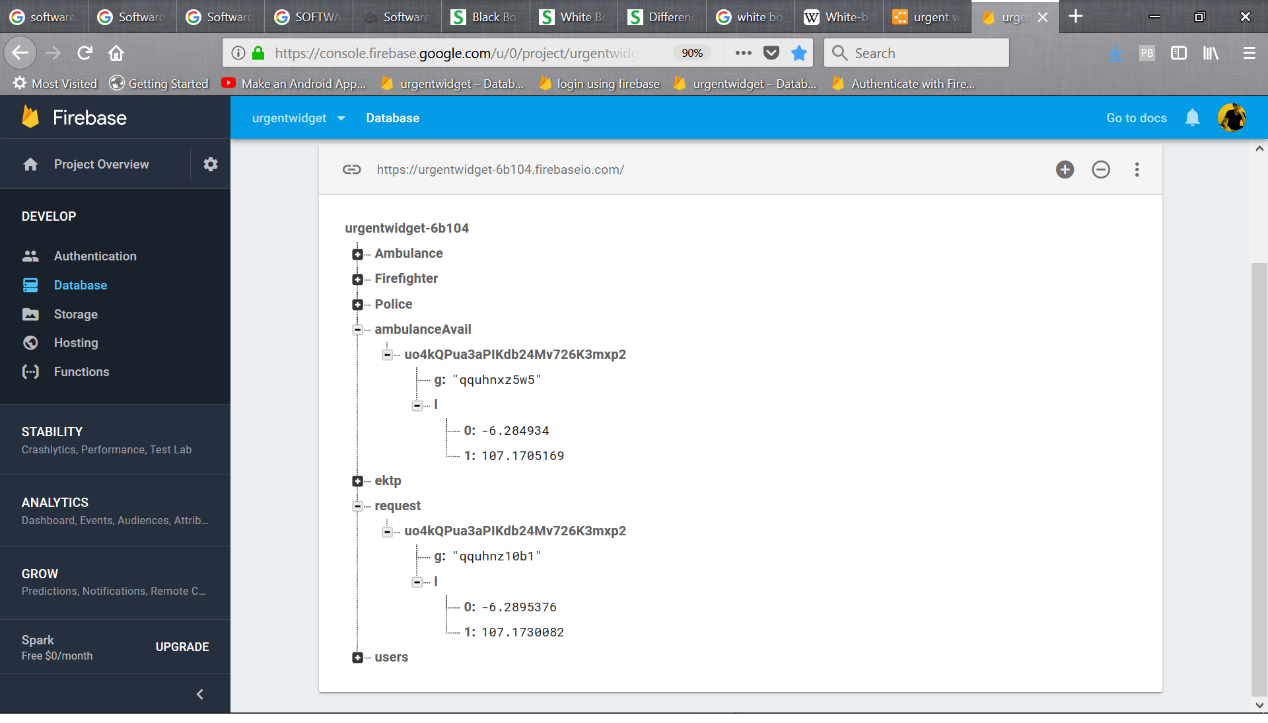
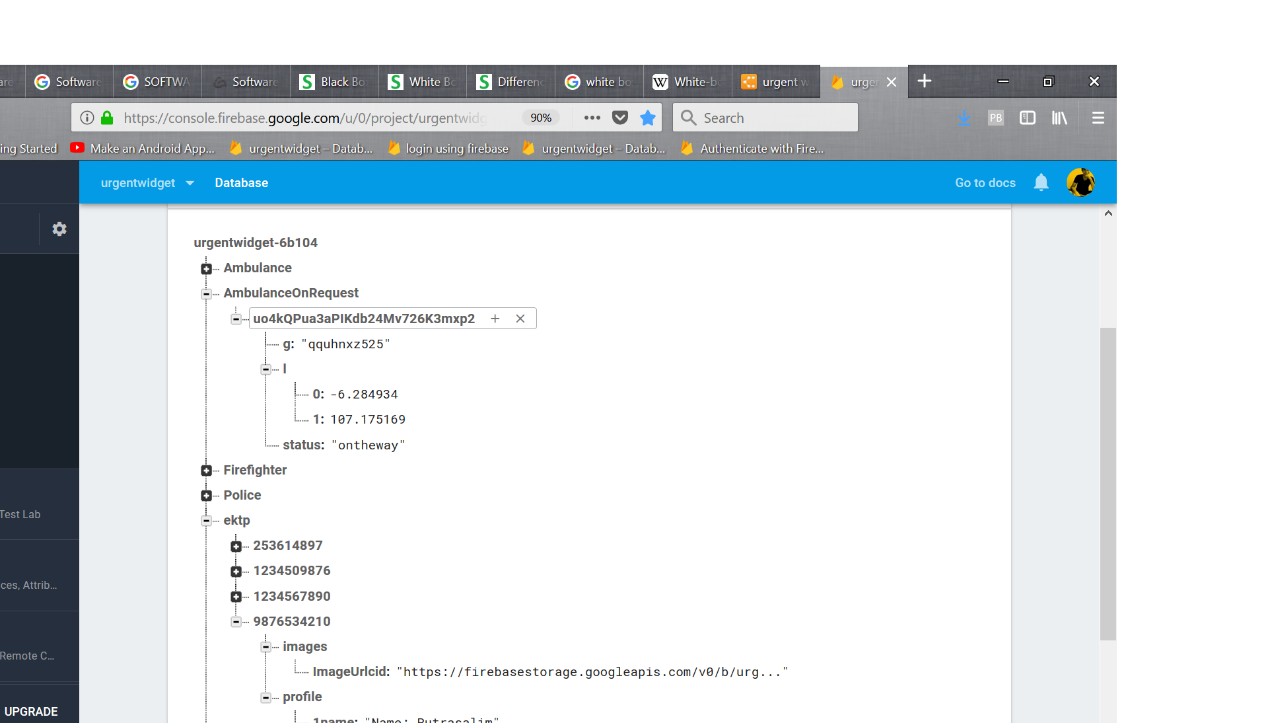
On the request state there are 2 things happen. First, the user’s user id and real-time location will be stored temporarily on request child (as shown in Figure 4.10) and scan for nearby related officer available from the officer available child. Second, when the officer found, the officer’s user id will be moved from available into officer onrequest child (as shown on Figure 4.11).

Figure 4.10 AmbulanceOnRequest Child

Figure 4.11 Request child

## Physical Design

Physical design will specify the tools of software development and the specification of hardware tools that is required for this application. There are minimum requirements for the system to run and fulfill its function. The development tools define the software that will be used in application development. Hardware devices are needed to run the application.

Table 4.5 Software Tool Development

|  |  |
| --- | --- |
| Operating System | Windows 10 64-bits is the platform chosen to develop the program. |
| Programming Language | Java is the programming language to develop android application |
| Integrated Development Environment | Android Studio is the IDE used as the programming environment because it is the IDE google develop for programmer to develop android application |
| Microsoft Office | Microsoft Office is needed to create documentation paper document in this application. Microsoft Office Word is used to write the development. |
| Firebase | Firebase is a cloud database that can store data or take data from it in real-time and firebase also supported on Android Studio, so there is no need of another way to connect to the database like using php or other. |

Table 4.6 Hardware Requirements

|  |  |
| --- | --- |
| Smartphone Operating System | At least Intel Core i3 processor, or above, is required for development and to run the program. |
| Memory | Minimum requirement is 512 MB RAM or above is recommended. |
| Hard Drive | Minimum requirement is 50MB of hard disk free size or above is recommended for application installation. |

# SYSTEM DEVELOPMENT

System development phase is continuation of the previous phase where the design of the system is created. A well-designed system can be implemented and developed in form of code. The programming language used is Java, developed using Android Studio and Google Maps API. System development will be divided to User Interface Development and Application Details.

## User Interface Development

The development in User Interface aims to give the picture of the real application appearance. The User Interface Development will follows the layout design made before with some improvements. There are some pages (activity) for this application which are Register, Login, Home (Tabbed Activity), Public Officer Page (contains Maps and User Profile Information whom pushed the button), and also the Widgets on home screen.

### Splash Screen

The application runs with splash screen as the first page. The splash screen shows the application logo and application name.

### Login Page

Figure 5.1 Splash screen

General user can do login if they already registered before or their parents already have an account under their name (for those who have not have citizenship ID will use their parent’s ID instead). Public Officer can login with their name, ID, and password.

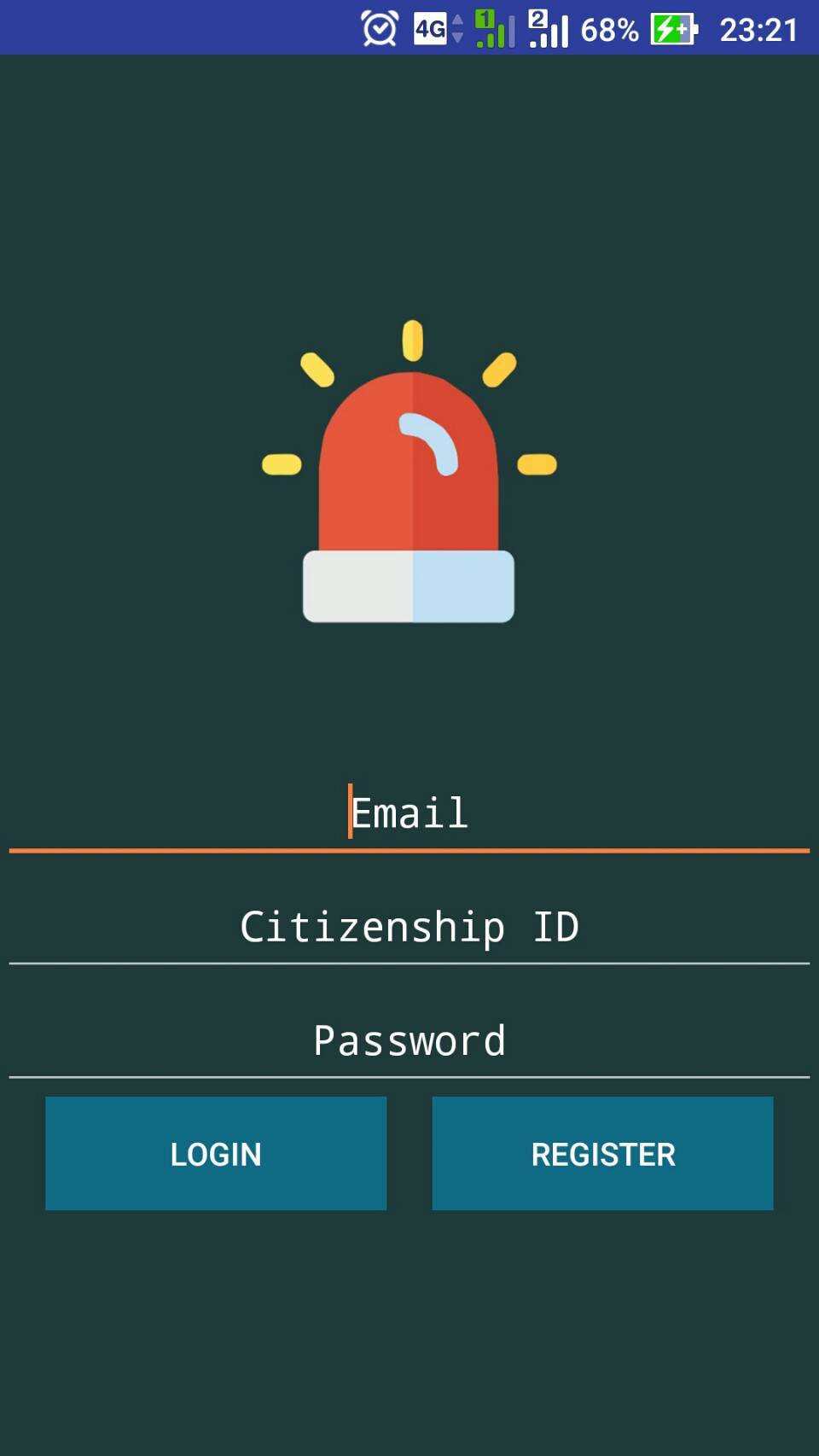


Figure 5.2 Login Page

### Home

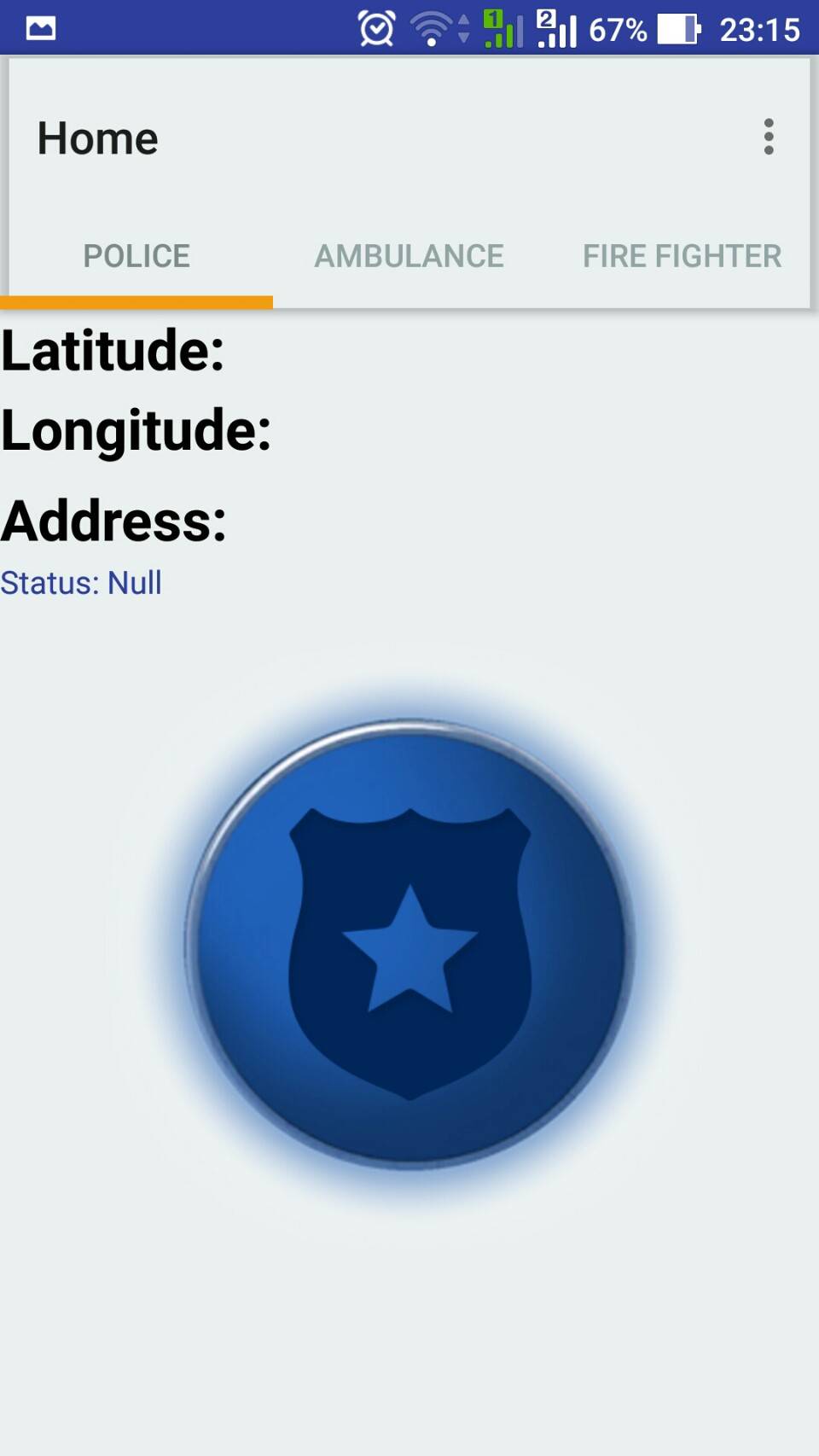
This page contains swipe activity where user can swipe through the page to choose the button they want to test. These buttons are to test if they work as what they should do. There will also toggle button to switch the button to actually really send notification to corresponding user. The home screen also contains a menu which have the task to cancel call (if call was performed) or to change the canceling password (if the call was not performed before) and adding medical record information (optional if the user have any account on several hospital).

Figure 5.3 Home Page

### Update data

There is an activity where user can change their phone number and upload their profile picture to help Public Officer to recognize their appearance. Also, there is add hospital and patient ID function to help ambulance get the patient medical record from related hospital the patient took treatment before. There is also list of hospital the user has added.

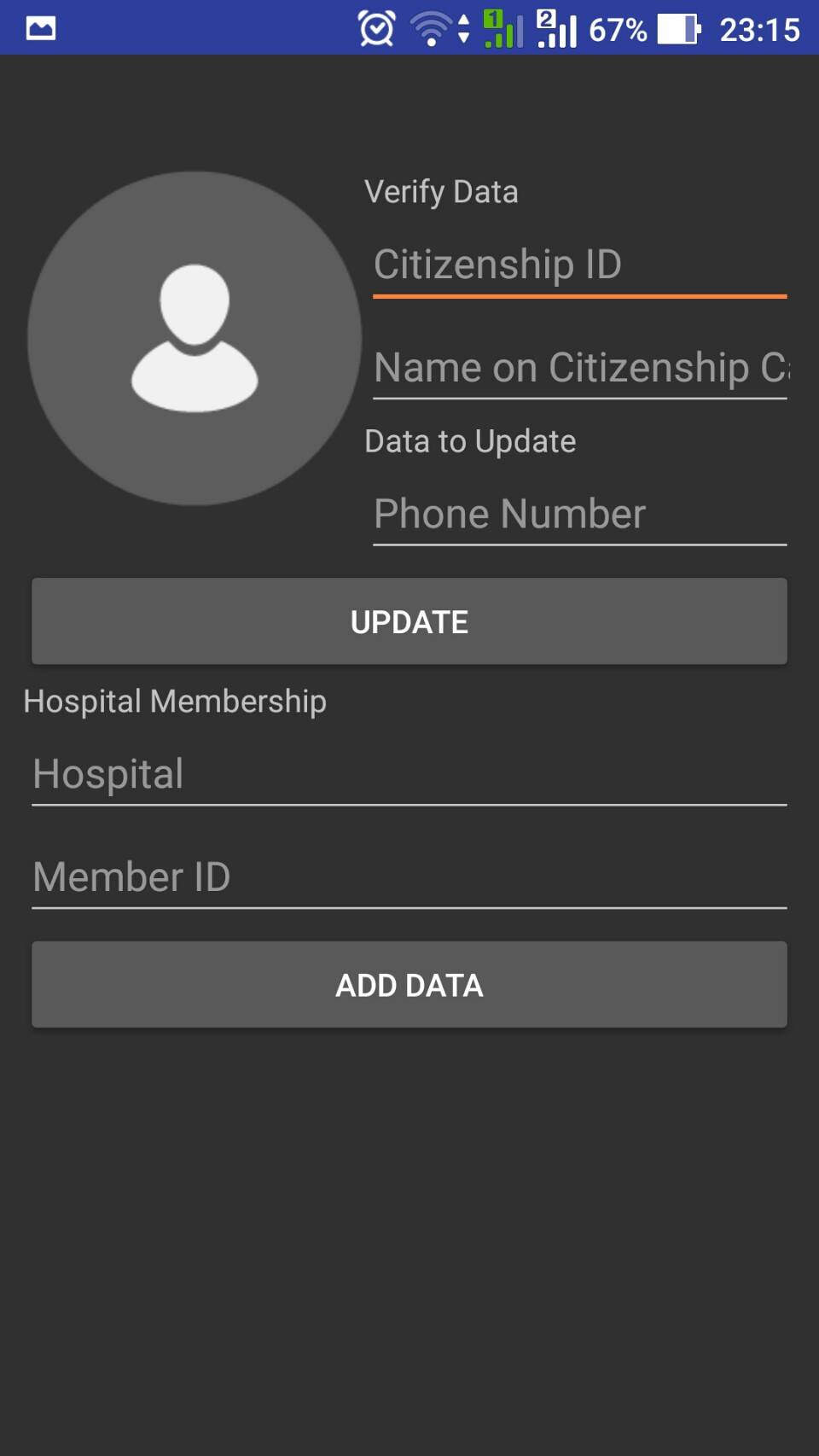


Figure 5.4 Update data

### Public Officer Page

For the user which have the role of Police, Ambulance, or Firefighter, they are not having the home page like the general user, rather these user have a special page that contains maps and toggle button to on or off notification of their account, screen that shows the user information (of user who calls them).



Figure 5.5 Public Officer main activity (map)

### Widget Button

Widget button appear on the home screen as the user set it to. User can choose whether they wants all the widget button to be on the screen or just some of it.

## Application Details

Software Details are describing the code that builds the program. The main code which important as the core code will be explain.

### Registration and Login Activity

Figure 5.6 Widget

These lines (Figure 5.7) are works to read data from database (determine whether the user is a legal citizen of Indonesia) and store user’s data to database (user data like password and other information that later will be used to inform the Public Officer about the user).

Figure 5.7 Registration

**final** DatabaseReference checkid = FirebaseDatabase.*getInstance*().getReference().child(**"ektp"**).child(mktp);

checkid.addValueEventListener(**new** ValueEventListener() {

@Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if**(dataSnapshot.exists() && dataSnapshot.getChildrenCount()>0){  
 Map<String,Object> map = (Map<String,Object>) dataSnapshot.getValue();  
  
 **if**(map.get(**"status"**)==**null**){  
 **mAuth**.createUserWithEmailAndPassword(username, pass).addOnCompleteListener(Login.**this**, **new** OnCompleteListener<AuthResult>() {

@Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
 **if** (!task.isSuccessful()) {  
 Toast.*makeText*(Login.**this**, **"error to register"**, Toast.***LENGTH\_SHORT***).show();  
 } **else** {  
 String user\_id = **mAuth**.getCurrentUser().getUid();  
 DatabaseReference current\_user\_db = FirebaseDatabase.*getInstance*().getReference().child(**"users"**).child(**"generaluser"**).child(user\_id);  
 current\_user\_db.setValue(**true**);  
 Map status = **new** HashMap();  
 status.put(**"status"**, **"registered"**);  
 checkid.updateChildren(status);  
 copyandpaste(checkid,current\_user\_db);  
 }  
 }  
 });  
  
 }**else if**(map.get(**"status"**)==**"registered"**){  
 Toast.*makeText*(Login.**this**, **"this ID is registered already! if it's not you, please contact admin.."**, Toast.***LENGTH\_LONG***).show();  
 }  
  
 }**else**{  
 Toast.*makeText*(Login.**this**, **"this ID is not registered as Indonesia citizen"**, Toast.***LENGTH\_LONG***).show();  
 }  
 }

It is not only to determine whether the ID exist but also to check if the ID has been registered before. If the account is succeeded to be registered, then system

will read the user profile data from database and copy them to be the account info. (Figure 5.8)

Figure 5.8 Copy user data to user account

**public void** copyandpaste(DatabaseReference fromPath, **final** DatabaseReference toPath)  
{  
 fromPath.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 toPath.setValue(dataSnapshot.getValue(), **new** DatabaseReference.CompletionListener() {  
 @Override  
 **public void** onComplete(DatabaseError databaseError, DatabaseReference databaseReference) {  
 **if** (databaseError != **null**)  
 {  
 Toast.*makeText*(Login.**this**, **"Failed to add user profile"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 **else** {  
 Toast.*makeText*(Login.**this**,**"Success"**,Toast.***LENGTH\_LONG***).show();  
 }  
 }  
 });  
 }

**Indonesia citizen"**, Toast.***LENGTH\_LONG***).show();  
 }  
 }

In order to use the application, user have to log in first. User have to login with citizenship ID to avoid the misapplication or misuse of the application from irresponsible party (Figure 5.9). These lines (Figure 5.10) are to check whether the user already login before, if the user already logged in in this device previously, the user does not have to login anymore the other time they run the application.

**final** DatabaseReference checkid = FirebaseDatabase.*getInstance*().getReference().child(**"ektp"**).child(mktp);  
checkid.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if**(dataSnapshot.exists() && dataSnapshot.getChildrenCount()>0){  
 Map<String,Object> map = (Map<String,Object>) dataSnapshot.getValue();  
  
 **if**(map.get(**"status"**).toString().equals(**"registered"**)){  
 **mAuth**.signInWithEmailAndPassword(username, pass).addOnCompleteListener(Login.**this**, **new** OnCompleteListener<AuthResult>() {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
 **if** (!task.isSuccessful()) {  
 Toast.*makeText*(Login.**this**, **"Login error"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
 }**else if**(map.get(**"status"**)==**""**){  
 Toast.*makeText*(Login.**this**, **"Account not Registered"**, Toast.***LENGTH\_SHORT***).show();  
 }  
  
 }**else**{  
 Toast.*makeText*(Login.**this**, **"No ID found in database"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }

Figure 5.10 Login

Figure 5.9 Authentication check

**mAuth** = FirebaseAuth.*getInstance*();  
**firebaseauthlistener** = **new** FirebaseAuth.AuthStateListener() {  
 @Override  
 **public void** onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {  
 FirebaseUser user = FirebaseAuth.*getInstance*().getCurrentUser();  
 **if** (user != **null**) {  
 Intent intent = **new** Intent(Login.**this**, Home.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 }  
};

### Google Maps

To use Google Maps API, there are some configuration to declare the usage of Google Maps API on the application tag (Figure 5.12) and set for permissions on the manifest.xml (Figure 5.11) and Google Maps key on the googlemaps.xml (Figure 5.13).

Figure 5.11 Set Permission on Manifest.xml

<**meta-data  
 android:name="com.google.android.geo.API\_KEY"  
 android:value="@string/google\_maps\_key"** />

<**uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"** />  
<**uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"** />  
<**uses-permission android:name="android.permission.INTERNET"** />

Figure 5.12 Meta-data setting

<**string name="google\_maps\_key" templateMergeStrategy="preserve" translatable="false"**>AIzaSyCEXt5btITpEHU\_ATi4hvto3RSbnRoQSks</**string**>

Figure 5.13 Key on googlemaps.xml

### Dependencies on build.gradle

To include Firebase and google maps service, there are some dependencies to declare on the application build.gradle (Figure 5.14).

### Location sharing

For the Public Officer to be able to track the user, this application needs to update the location of the user after they click the button. Then after clicking the button, and check for location enabled, if the location is not enabled yet, then there will be a pop up message and if user click ok then the application will direct user to location setting. If the location enabled, then system will run a function to check for nearby police available. (Figure 5.15)

Figure 5.15 check location enabled and get location

**reqstatus**.setText(**"Status: looking for Police.."**);  
 **gps** = **new** request\_service(getActivity());  
  
 **if** (**gps**.CanGetLocation()) {  
 **double** latitude = **gps**.getLatitude();  
 **double** longitude = **gps**.getLongitude();  
  
 **geocoder** = **new** Geocoder(getActivity(), Locale.*getDefault*());  
 **try** {  
 **addr** = **geocoder**.getFromLocation(latitude, longitude, 1);  
 String addrs = **addr**.get(0).getAddressLine(0);String fulladdress = addrs;  
 **addres**.setText(**"Address :"** + fulladdress);  
 **addres**.setMovementMethod(**new** ScrollingMovementMethod());  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **mLocReq** = **new** LocationRequest();  
 **mLocReq**.setInterval(2000);  
 **mLocReq**.setFastestInterval(2000);  
 **mLocReq**.setPriority(LocationRequest.***PRIORITY\_HIGH\_ACCURACY***);  
 nearestPolice();  
  
 } **else** {  
 **gps**.showSettingAlert();  
 }

Figure 5.14 Build.gradle dependencies

dependencies {  
 implementation **'com.google.android.gms:play-services-maps:11.8.0'** *//ini ada error tapi di surpress - implementation 'com.android.support:support-v13:23.4.0'  
 //noinspection GradleCompatible* implementation **'com.android.support:support-v13:23.4.0'** implementation **'com.android.support.constraint:constraint-layout:1.0.2'** compile fileTree(**dir**: **'libs'**, **include**: [**'\*.jar'**])  
 testCompile **'junit:junit:4.12'** *//noinspection GradleCompatible* compile **'com.android.support:appcompat-v7:23.4.0'** compile **'com.android.support:design:23.4.0'** compile **'com.android.support:support-v4:23.4.0'** compile **'com.google.firebase:firebase-core:11.8.0'** compile **'com.google.firebase:firebase-database:11.8.0'** compile **'com.google.firebase:firebase-auth:11.8.0'** compile **'com.google.firebase:firebase-storage:11.8.0'** compile **'com.github.bumptech.glide:glide:4.0.0'** compile **'com.google.android.gms:play-services-location:11.8.0'** compile **'com.firebase:geofire-android:2.2.0'**}  
apply **plugin**: **'com.google.gms.google-services'**

The function to scan for nearby Public Officer and assign Public Officer to this user.

GeoQuery **geoQuery**;  
**public void** nearestPolice(){  
 **reqstatus**.setText(**"Status: Finding Police"**);  
 **final double** lat = **gps**.getLatitude();  
 **final double** lo = **gps**.getLongitude();  
 DatabaseReference policeLoc = FirebaseDatabase.*getInstance*().getReference().child(**"policeAvail"**);  
 **final** GeoFire geoFire = **new** GeoFire(policeLoc);  
  
 **geoQuery** = geoFire.queryAtLocation(**new** GeoLocation(lat, lo), **radius**);  
 **geoQuery**.removeAllListeners();  
  
 **geoQuery**.addGeoQueryEventListener(**new** GeoQueryEventListener() {  
 @Override  
 **public void** onKeyEntered(String key, GeoLocation location) {  
 **if**(!**policeFound**){  
 **policeFound** = **true**;  
 **policeFoundID** = key;

DatabaseReference policeRef = FirebaseDatabase.*getInstance*().getReference().child(**"users"**).child(**"specialuser"**).child(**"Police"**).child(**policeFoundID**);

String userID = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 HashMap map = **new** HashMap();  
 map.put(**"requestID"**, userID);  
 policeRef.updateChildren(map);  
  
 DatabaseReference requesttt = FirebaseDatabase.*getInstance*().getReference().child(**"request"**).child(userID).child(**"status1"**);  
 requesttt.setValue(**"on progress"**);**latd**.setText(**"Latitude: "**+ lat);  
 **longit**.setText(**"Longitude: "**+ lo);  
 *req\_police* = **true** ;  
 checkpolice();  
 }  
 }  
 @Override  
 **public void** onGeoQueryReady() {  
 **if**(!**policeFound**){  
 **radius**++;  
 nearestPolice();  
 }  
 }  
 });  
 **reqstatus**.setText(**"Status: Got Nearby Police"**);  
 checkpolice();  
}

Figure 5.16 scan nearby Public Officer and make contact

After making contact, there is a function to check if the contact has completed or the Public Officer suddenly not available which will leads to making another contact. (Figure 5.17)

Figure 5.17 Check contact status

**public void** checkpolice(){  
 String userID = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 **if** (**policeFoundID** != **null**){  
 **final** DatabaseReference checkPoliceStatus = FirebaseDatabase.*getInstance*().getReference(**"request"**).child(userID).child(**"status1"**);  
 checkPoliceStatus.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()){  
 String value = dataSnapshot.getValue().toString();  
 String done = **"done"**;  
 String notdone = **"notdone"**;  
 **if**(value.equals(done)) {  
 requestcompleted();  
 }**else if** (value.equals(notdone)){  
 **radius** =1;  
 **policeFoundID**=**null**;  
 **policeFound** = **false**;  
 nearestPolice();  
 }**else** {  
 checkpolice();  
 }  
 }  
 }

### Cancel a contact by user

If the user already feel safe or the just clicked the button by accident and want to cancel it, they also able to cancel a contact by choose cancel menu from menu bar and the cancel function will run and stop sharing the user location to assigned officer if contact has been made, or delete the request from database if there is no officer assigned to this request before.

### Function runs if the Officer choose completed menu

Figure 5.18 Cancel Request by User

**public void** cancelRequest(){  
 **geoQuery**.removeAllListeners();  
 **if**(**policeFoundID** != **null**){  
 DatabaseReference policeRef = FirebaseDatabase.*getInstance*().getReference().child(**"users"**).child(**"specialuser"**).child(**"Police"**).child(**policeFoundID**);  
 policeRef.setValue(**true**);  
 **policeFoundID** = **null**;  
 }  
 **policeFound** = **false**;  
 **radius** = 1;  
 String userID = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 DatabaseReference refReq = FirebaseDatabase.*getInstance*().getReference(**"request"**);  
 GeoFire geoFireReq = **new** GeoFire(refReq);  
 geoFireReq.removeLocation(userID);**addres**.setText(**"Address:"**);  
 **latd**.setText(**"Latitude:"**);  
 **longit**.setText(**"Longitude:"**);  
 **reqstatus**.setText(**"Status:"**);  
 *req\_police* = **false**;  
 *// btnlocation.setPressed(false);* }

ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()){  
 String value = dataSnapshot.getValue().toString();  
 String done = **"done"**;  
 String notdone = **"notdone"**;  
 **if**(value.equals(done)) {  
 requestcompleted();  
 }**else if** (value.equals(notdone)){  
 **radius** =1;  
 **policeFoundID**=**null**;  
 **policeFound** = **false**;  
 nearestPolice();  
 }**else** {  
 checkpolice();  
 }  
 }  
 }

If Public Officer clicks the request completed, then user will catch the response and run this function (Figure 5.19).

### Public Officer register an login

Almost the same like user Register and login, but for Public Officer as Public Officer is divided into 3 different category of account so the function might have a bit different. For registering, Public Officer has to choose on dropdown menu which category they are. And system will check the working ID database according to the account they choose. (Figure 5.20)

If the user has been registered and wish to login, then they have to still choose the user category and fill the data needed to login. (Figure 5.21)

**breg**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
 **final** String emaill = **email**.getText().toString();  
 **final** String pass = **password**.getText().toString();  
 **final** String mworkid = **workid**.getText().toString();  
 **datacheckid** = FirebaseDatabase.*getInstance*().getReference().child(**selectedItem**).child(mworkid);  
  
 **datacheckid**.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if**(dataSnapshot.exists() && dataSnapshot.getChildrenCount()>0){  
 **mAuth**.createUserWithEmailAndPassword(emaill, pass).addOnCompleteListener(Login.**this**, **new** OnCompleteListener<AuthResult>() {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {

**if** (!task.isSuccessful()) {  
Toast.*makeText*(Login.**this**, **"error to register"**, Toast.***LENGTH\_SHORT***).show();

} **else** {

String user\_id = **mAuth**.getCurrentUser().getUid();

Figure 5.19 Request completed

**public void** requestcompleted(){  
  
 **geoQuery**.removeAllListeners();  
  
 **if**(**policeFoundID** != **null**){  
 **policeFoundID** = **null**;  
 }  
 **policeFound** = **false**;  
 **radius** = 1;  
 String userID = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
  
 DatabaseReference refReq = FirebaseDatabase.*getInstance*().getReference(**"request"**);  
 GeoFire geoFireReq = **new** GeoFire(refReq);  
 geoFireReq.removeLocation(userID);  
 DatabaseReference refRequser = FirebaseDatabase.*getInstance*().getReference(**"request"**).child(userID);  
 refRequser.removeValue();  
 *req\_police* = **false**;  
 }

Figure 5.21 Function of registering

**blogin**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **final** String emaill = **email**.getText().toString();  
 **final** String pass = **password**.getText().toString();  
 **final** String mworkid = **workid**.getText().toString();  
  
 **datacheckid** = FirebaseDatabase.*getInstance*().getReference().child(**selectedItem**).child(mworkid);  
  
 **datacheckid**.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if**(dataSnapshot.exists() && dataSnapshot.getChildrenCount()>0){  
 **mAuth**.signInWithEmailAndPassword(emaill, pass).addOnCompleteListener(Login.**this**, **new** OnCompleteListener<AuthResult>() {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
 **if** (!task.isSuccessful()) {  
 Toast.*makeText*(Login.**this**, **"Login error"**, Toast.***LENGTH\_SHORT***).show();  
 }**else if**(task.isSuccessful()){  
 Toast.*makeText*(Login.**this**, **"Login Success"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
 }**else**{  
 Toast.*makeText*(Login.**this**, **"cannot Login"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });

Figure 5.20 Login for Public Officer

DatabaseReference current\_user\_db = FirebaseDatabase.*getInstance*().getReference().child(**"users"**).child(**"specialuser"**).child(**selectedItem**).child(user\_id);  
 current\_user\_db.setValue(**true**);  
 }  
 }  
 });  
  
 }**else**{  
 Toast.*makeText*(Login.**this**, **"cannot Register"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
  
 });  
 }  
});

Also if there the user already login before and reopen the application, they do not have to re login. (Figure 5.22)

Figure 5.22 Authentication check

**mAuth** = FirebaseAuth.*getInstance*();  
**firebaseauthlistener** = **new** FirebaseAuth.AuthStateListener() {  
 @Override  
 **public void** onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {  
 **if**(firebaseAuth.getCurrentUser()!=**null**){  
 String userID = firebaseAuth.getCurrentUser().getUid();  
 **checkuser** = FirebaseDatabase.*getInstance*().getReference().child(**"users"**).child(**"specialuser"**).child(**"Police"**).child(userID);  
 **checkuser2** = FirebaseDatabase.*getInstance*().getReference().child(**"users"**).child(**"specialuser"**).child(**"Ambulance"**).child(userID);  
 **checkuser3** = FirebaseDatabase.*getInstance*().getReference().child(**"users"**).child(**"specialuser"**).child(**"Firefighter"**).child(userID);  
  
 **checkuser**.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()){  
 startActivity(**new** Intent(Login.**this**,policemapActivity.**class**));  
 }  
 }  
 });  
 **checkuser2**.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()){  
  
 startActivity(**new** Intent(Login.**this**,ambulancemapActivity.**class**));  
 }  
 }  
 });  
 **checkuser3**.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()){  
  
 startActivity(**new** Intent(Login.**this**,firemapActivity.**class**));  
 }  
 }  
 });  
  
 }  
 }  
 };

### Be available to be contact

For the Public Officer activity, there will be a switch if the user is available to take a request. If the switch is on the on state, then it will run check connection function and if it returns true, then it will run generate map function. If the switch changed into off state, then it will run not available function and stopping GPS checking. (Figure 5.23)

Figure 5.23 Switch

**ssonoff**.setOnCheckedChangeListener(**new** CompoundButton.OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(CompoundButton buttonView, **boolean** isChecked) {  
 **if** (isChecked){  
 checkconnection();  
 **if**(**connected**){  
 generateMap();  
 }**else**{  
 **ssonoff**.setChecked(**false**);  
 }  
 }  
 **else if**(!isChecked){  
 **gps**.stopGPS();  
 notAvail();  
 }  
 }  
});

**ssonoff**.setOnCheckedChangeListener(**new** CompoundButton.OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(CompoundButton buttonView, **boolean** isChecked) {  
 **if** (isChecked){  
 checkconnection();  
 **if**(**connected**){  
 generateMap();  
 }**else**{  
 **ssonoff**.setChecked(**false**);  
 }  
 }  
 **else if**(!isChecked){  
 **gps**.stopGPS();  
 notAvail();  
 }  
 }  
});

### Checking Public Officer Connection

**private void** checkconnection(){  
 ConnectivityManager connectivityManager = (ConnectivityManager)getSystemService(Context.***CONNECTIVITY\_SERVICE***);  
**if**(connectivityManager.getNetworkInfo(ConnectivityManager.***TYPE\_MOBILE***).getState() == NetworkInfo.State.***CONNECTED*** ||  
 connectivityManager.getNetworkInfo(ConnectivityManager.***TYPE\_WIFI***).getState() == NetworkInfo.State.***CONNECTED***) {**connected** = **true**;  
 }  
 **else**{  
 Toast.*makeText*(**this**, **"no internet access"**, Toast.***LENGTH\_SHORT***).show();  
 **connected** = **false**;}

Figure 5.24 Checking connection

### Generate map

After the switch turned on, the map will proceed and show the officer’s location.

Figure 5.25 generating map

**public void** generateMap(){  
 **if** (**gps**.**canGetLocation**){  
 SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
 .findFragmentById(R.id.***map***);  
 mapFragment.getMapAsync(**this**);  
 }**else** {  
 **ssonoff**.setChecked(**false**);  
 **gps**.showSettingAlert();  
 }  
}

This on map ready called to show Public Officer position and call getassigneduser() function.

Figure 5.26 Set point and call get assigned user function

**public void** onMapReady(GoogleMap googleMap) {  
 **mMap** = googleMap;  
  
 **if** (ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {**return**;  
 }  
 buildGoogleAPIClint();  
 **mMap**.setMyLocationEnabled(**true**);  
 **mMap**.animateCamera(CameraUpdateFactory.*zoomTo*(19));  
 **ssonoff**.setChecked(**true**);  
 getAssignedUser();  
}

### Scan to see if there is assigned user to Public Officer

A function to see if there is a request from user and if there is, then the system will store user’s data to a list. After that run a function to get user’s location.

To get user’s location and show it on the Public Officer map as a target. (Figure 5.28)

**private void** getUserLocation(){  
 **assignedUserLocationRef** = FirebaseDatabase.*getInstance*().getReference().child(**"request"**).child(**generaluserID**).child(**"l"**);  
 **assignedUserLocationRefListener** = **assignedUserLocationRef**.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if**(dataSnapshot.exists()){  
 List<Object> map = (List<Object>) dataSnapshot.getValue();  
 **double** locationLat = 0;  
 **double** locationLng = 0;  
 **if**(map.get(0) != **null**){  
 locationLat = Double.*parseDouble*(map.get(0).toString());

**private void** getAssignedUser(){  
 String policeID = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 **final** DatabaseReference assignedUserRef = FirebaseDatabase.*getInstance*().getReference().child(**"users"**).child(**"specialuser"**).child(**"Police"**).child(policeID).child(**"requestID"**);  
 assignedUserRef.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if**(dataSnapshot.exists() && **generaluserID**.equals(**""**)){  
 **generaluserID** = dataSnapshot.getValue().toString();  
 **mProfile** = FirebaseDatabase.*getInstance*().getReference().child(**"users"**).child(**"generaluser"**).child(**generaluserID**).child(**"profile"**);  
 **if**(**listlist**.isEmpty()){  
 getinfo();  
 }  
 }**else** {  
 **generaluserID** = **""**;  
 **if** (**mUserMarker** != **null**) {  
 **mUserMarker**.remove();  
 }  
 **if** (**assignedUserLocationRefListener** != **null**) {  
 **assignedUserLocationRef**.removeEventListener(**assignedUserLocationRefListener**);  
 **listlist**.clear();  
 **showuserinfo**.setVisibility(**showuserinfo**.***INVISIBLE***);  
 }  
  
 }  
 }  
  
 });  
}

Figure 5.27 get Assigned user

### Show user image (profile photo and ID photo)

Figure 5.28 Show user location on map and show menu

}  
 **if**(map.get(1) != **null**){  
 locationLng = Double.*parseDouble*(map.get(1).toString());  
 }  
  
 LatLng userLatLng = **new** LatLng(locationLat,locationLng);  
 **if** (**mUserMarker** != **null**){  
 **mUserMarker**.remove();  
 }  
 **mUserMarker** = **mMap**.addMarker(**new** MarkerOptions().position(userLatLng).title(**"Target"**).icon(BitmapDescriptorFactory.*fromResource*(R.mipmap.***sos\_marker***)));  
 **showuserinfo**.setVisibility(**showuserinfo**.***VISIBLE***);  
 **menu**.setVisibility(**menu**.***VISIBLE***);  
 **logout**.setVisibility(**logout**.***INVISIBLE***);  
 }  
 }  
 });  
}

On user info shown for Public Officer after getting the user assigned to them, there are 2 photos, which are from their ID and from what user upload.

DatabaseReference images = FirebaseDatabase.*getInstance*().getReference().child(**"users"**).child(**"generaluser"**).child(**generaluserID**).child(**"images"**);  
images.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if**(dataSnapshot.exists() && dataSnapshot.getChildrenCount()>0){  
 Map<String,Object> map = (Map<String,Object>) dataSnapshot.getValue();  
 **if**(map.get(**"ImagesUrlprofile"**)!=**null**){  
 **mProfileImageUrl** = map.get(**"ImagesUrlprofile"**).toString();  
 Glide.*with*(getApplication()).load(**mProfileImageUrl**).into(**image\_profile**);  
 }  
 **if**(map.get(**"ImageUrlcid"**)!=**null**){  
 **mCidImageUrl** = map.get(**"ImageUrlcid"**).toString();  
 Glide.*with*(getApplication()).load(**mCidImageUrl**).into(**image\_cid**);  
 }  
 }  
 }

Figure 5.29 Get Image from Firebase

### Show user info

After getting the assigned user, Public Officer get the user personal info in order to make Public Officer easier to recognize the user. (Figure 5.30)

Figure 5.30 Get user Information from database as a list

**mProfile**.addChildEventListener(**new** ChildEventListener() {  
@Override  
**public void** onChildAdded(DataSnapshot dataSnapshot, String s) {  
 **if**(dataSnapshot.exists()){  
 String value= dataSnapshot.getValue(String.**class**);  
 **listlist**.add(value);  
 }  
}

### Public Officer available

When Public Officer clicks set the switch to on state, it automatically read Public Officer location and store to database in order for user to scan nearby Public Officer they need. So system will put the Public Officer to Available table when there are no any user assigned to them and put them to on request when Public Officer is already assigned to a user. (Figure 5.31)

**mlocation** = location;  
 LatLng latLng = **new** LatLng(location.getLatitude(), location.getLongitude());  
 **mMap**.moveCamera(CameraUpdateFactory.*newLatLng*(latLng));String userID = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
  
 DatabaseReference refPoliceAvali = FirebaseDatabase.*getInstance*().getReference(**"policeAvail"**);  
 DatabaseReference refPoliceNotAvali = FirebaseDatabase.*getInstance*().getReference(**"policeOnRequest"**);  
  
 DatabaseReference listenerrequestID = FirebaseDatabase.*getInstance*().getReference(**"users"**).child(**"specialuser"**).child(**"Police"**).child(userID).child(**"requestID"**);  
 listenerrequestID.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if**(dataSnapshot.exists()){

### Public Officer suddenly not available in any reason

Figure 5.31 Function to determine where to put the Public Officer location

**generaluserID** = dataSnapshot.getValue().toString();  
 }  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
 });  
  
  
 GeoFire geoFireAvail = **new** GeoFire(refPoliceAvali);  
 GeoFire geoFireNotAvail = **new** GeoFire(refPoliceNotAvali);  
  
 **if**(**ssonoff**.isChecked()) {  
 **switch** (**generaluserID**) {  
 **case ""**:  
 geoFireNotAvail.removeLocation(userID);  
 geoFireAvail.setLocation(userID, **new** GeoLocation(location.getLatitude(), location.getLongitude()));  
 **break**;  
 **default**:  
 geoFireAvail.removeLocation(userID);  
 geoFireNotAvail.setLocation(userID, **new** GeoLocation(location.getLatitude(), location.getLongitude()));  
 getUserLocation();  
 **break**;  
 }  
 }  
 }

There will be a time when Public Officer get a request but suddenly he cannot do the request or when he want to switch off but the request get in at the same time. So this function runs to read if the request not done but Public Officer suddenly switch off the application. (Figure 5.32)

### Public Officer completed the request

When the Public Officer has done the job and now can be available again, there is an option on the menu that will be appear only when there is assigned user. This function (Figure 5.33) resets all the changes and sending signal to user to stop sharing location due to the completed assignment.

Figure 5.32 Public Officer not available function

**private void** notAvail(){  
 **mUserMarker**.remove();  
 **mGoogleApiClient**.disconnect();  
  
 String userID = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
  
 DatabaseReference noreq = FirebaseDatabase.*getInstance*().getReference(**"policeAvail"**);  
 DatabaseReference onreq = FirebaseDatabase.*getInstance*().getReference(**"policeOnRequest"**);  
 DatabaseReference reqID = FirebaseDatabase.*getInstance*().getReference(**"request"**).child(**generaluserID**).child(**"status1"**);  
 DatabaseReference request = FirebaseDatabase.*getInstance*().getReference(**"users"**).child(**"specialuser"**).child(**"Police"**).child(userID);  
  
 GeoFire geoFireonreq = **new** GeoFire(onreq);  
 GeoFire geoFirenoreq = **new** GeoFire(noreq);  
  
 **switch** (**generaluserID**) {  
 **case ""**:  
 geoFirenoreq.removeLocation(userID);  
  
 **break**;  
 **default**:  
 geoFireonreq.removeLocation(userID);  
 reqID.setValue(**"notdone"**);  
 request.setValue(**true**);  
 **menu**.setVisibility(**menu**.***INVISIBLE***);  
 **break**;  
 }  
}

**private void** notAvail(){  
 **mUserMarker**.remove();  
 **mGoogleApiClient**.disconnect();  
  
 String userID = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
  
 DatabaseReference noreq = FirebaseDatabase.*getInstance*().getReference(**"policeAvail"**);  
 DatabaseReference onreq = FirebaseDatabase.*getInstance*().getReference(**"policeOnRequest"**);  
 DatabaseReference reqID = FirebaseDatabase.*getInstance*().getReference(**"request"**).child(**generaluserID**).child(**"status1"**);  
 DatabaseReference request = FirebaseDatabase.*getInstance*().getReference(**"users"**).child(**"specialuser"**).child(**"Police"**).child(userID);  
  
 GeoFire geoFireonreq = **new** GeoFire(onreq);  
 GeoFire geoFirenoreq = **new** GeoFire(noreq);  
  
 **switch** (**generaluserID**) {  
 **case ""**:  
 geoFirenoreq.removeLocation(userID);  
  
 **break**;  
 **default**:  
 geoFireonreq.removeLocation(userID);  
 reqID.setValue(**"notdone"**);  
 request.setValue(**true**);  
 **menu**.setVisibility(**menu**.***INVISIBLE***);  
 **break**;  
 }  
}

### User update profile

In this activity, user can update their phone number (Figure5.34) and upload profile picture (Figure5.35) and add some medical data (hospital name and patient membership id) (Figure5.36) that will be shown to the ambulance assigned to user that get assigned to them. But first, user has to fill their full name and citizenship ID as identification and to prevent false data update, means to user to be fully aware of the update.

**if** (**checkname**.equals(**dataname**) && **checkID**.equals(**datacitizenID**)){  
 Map userprofileInfo = **new** HashMap();  
 **if**(**phone**!=**null**) {  
 userprofileInfo.put(**"3phone"**, **"Phone: "**+**newphone**);  
 **mUserprofileDatabase**.updateChildren(userprofileInfo);  
 }

Figure 5.33 Task completed

Figure 5.34 Update phone number

**if** (**result\_uri** != **null** ){  
 StorageReference filepath = FirebaseStorage.*getInstance*().getReference().child(**"profileImages"**).child(**generaluserID**);  
 Bitmap bitmap = **null**;  
  
 **try** {  
 bitmap = MediaStore.Images.Media.*getBitmap*(getApplication().getContentResolver(), **result\_uri**);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }

**private void** requestcomplete() {  
 **mUserMarker**.remove();  
  
 **menu**.setVisibility(**menu**.***INVISIBLE***);  
 **popup\_menu**.setVisibility(**menu**.***INVISIBLE***);  
 String userID = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
  
 DatabaseReference refPoliceAvali = FirebaseDatabase.*getInstance*().getReference(**"policeAvail"**);  
 DatabaseReference policeonreq= FirebaseDatabase.*getInstance*().getReference().child(**"policeOnRequest"**);  
 GeoFire geoFire = **new** GeoFire(policeonreq);  
 GeoFire geoFireAvail = **new** GeoFire(refPoliceAvali);  
 geoFire.removeLocation(userID);  
 geoFireAvail.setLocation(userID, **new** GeoLocation(**gps**.getLatitude(), **gps**.getLongitude()));  
  
 DatabaseReference statustrue = FirebaseDatabase.*getInstance*().getReference(**"users"**).child(**"specialuser"**).child(**"Police"**).child(userID);  
 statustrue.setValue(**true**);  
  
 generateMap();  
}

Figure 5.35 Add hospital record

**if** (**checkname**.equals(**dataname**) && **checkID**.equals(**datacitizenID**)){  
 **if** (**datahospitalname** != **""** && **datauserpatientID** != **""**){  
 Map userHospitalMember = **new** HashMap();  
 userHospitalMember.put(**"hospital"**+ **datacounthospital**, **datahospitalname**+**" = "** + **datauserpatientID**);  
 **mUserMedicarRecord**.updateChildren(userHospitalMember);  
 Toast.*makeText*(**this**,**"Data added:"** +**datahospitalname** +**" = "**+**datauserpatientID**,Toast.***LENGTH\_SHORT***).show();  
 **hospitalName**.setText(**""**);  
 **userpatientID**.setText(**""**);  
 **citizenshipID**.setText(**""**);  
 **name**.setText(**""**);  
 hidekeyboard();  
 getInfo();  
 }**else** {  
 Toast.*makeText*(**this**,**"Please fill the data"**,Toast.***LENGTH\_SHORT***).show();  
 }  
}**else**{  
 Toast.*makeText*(**this**, **"Cannot add data due to verification failure"**, Toast.***LENGTH\_SHORT***).show();  
}

Figure 5.36 Update profile picture

ByteArrayOutputStream baos = **new** ByteArrayOutputStream();  
 bitmap.compress(Bitmap.CompressFormat.***JPEG***, 20, baos);  
 **byte**[] data = baos.toByteArray();  
 UploadTask uploadTask = filepath.putBytes(data);  
  
 uploadTask.addOnFailureListener(**new** OnFailureListener() {  
 @Override  
 **public void** onFailure(@NonNull Exception e) {  
 finish();  
 **return**;  
 }  
 });  
 uploadTask.addOnSuccessListener(**new** OnSuccessListener<UploadTask.TaskSnapshot>() {  
 @Override  
 **public void** onSuccess(UploadTask.TaskSnapshot taskSnapshot) {  
 Uri downloadUri = taskSnapshot.getDownloadUrl();  
 DatabaseReference profileImagedatabse = FirebaseDatabase.*getInstance*().getReference().child(**"users"**).child(**"generaluser"**).child(**generaluserID**).child(**"images"**);  
 Map newImage = **new** HashMap();  
 newImage.put(**"ImagesUrlprofile"**, downloadUri.toString());  
 profileImagedatabse.updateChildren(newImage);  
 finish();  
 **return**;  
 }  
 });  
}**else** {  
 finish();  
}

### Logout

Originally, this application does not have a logout option whether for user nor Public Officer. But, in the development of the application there has to be logout option in order to test feature. So this is the logout code.

Figure 5.37 Logout

FirebaseAuth.*getInstance*().signOut();  
intent = **new** Intent(Home.**this** , Login.**class**);  
startActivity(intent);  
finish();

# SYSTEM TESTING

In order to know whether the application is already fulfill the original function and requirement, it has to be tested. The system has to be tested on its capability to run process as expected or even better. The application will be tested in some different scenario and possibility that can happened at the time the application run.

## Testing Environment

Testing environment here is the description of the devices that will be use to run the application testing. The application can be categorize as fully functional if the application can be run without any error on this device as the application was build base on this device’s SDK.

Table 6.1 Device Specification

|  |  |  |
| --- | --- | --- |
| **No** | **Device 1** | **Device 2** |
| 1. | Asus Zenfone 2 (ZE551ML) | Asus Zenfone 2 (ZE551ML) |
| 2. | Marshmallow Operating System  (Android API 23) | Android Operating System  (Android API 23) |
| 3. | Ram 64-bit Dual-Channel 4 GB | Ram 64-bit Dual-Channel 4 GB |
| 4. | CPU 64-bit Intel Atom 2.3G | CPU 64-bit Intel Atom 2.3G |

## Testing Scenario

Testing scenario is based on the main functionalities of this program that are checking identity, scan location, and send notification to nearby party needed. The testing scenario and the result will be describe as follows:

### Login

Login activity have to be done in order to access the application homepage. This activity can be done if the user is already registered as application user.

Table 6.2 Login scenario

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Scenario** | **Expected Result** | **Result** |
| 1. | Valid name, ID, and password | Directed to homepage | As expected |
| 2. | Invalid name, ID, or password | Display login error message | As expected |
| 3. | Click the register button | User registered and directed to homepage | As expected |

### Register new account

This event of testing held when user click the Register text on Login page. Aims to add new user information by verify first whether the user is real person and registered as citizen of Indonesia then check if the ID has been registered as application user.

Table 6.3 Registration scenario

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Scenario** | **Expected Result** | **Result** |
| 1. | Internet is on | Registration complete and user profile stored on application user database, directed to homepage | As expected |
| ID is verified as Indonesian |
| ID has not registered as  application user |
| Password and confirmation password match |
| 2. | Internet is on | Displaying message that the password not match | As expected |
| ID is verified as Indonesian |
| ID has not registered as  application user |
| Password and confirmation password does not match |
| 2. | Internet is on | Displaying message that the user is already registered | As expected |
| ID is verified as Indonesian |
| ID already registered as application user |
| 3. | Internet is on | Displaying message that register error. | As expected |
| ID not found on Indonesian e-ktp database |
| 4. | Internet is off | Displaying no internet connection toast | As expected |

### Test button

Each of the button has the test mode on the application homepage which has its own unique function. Like for police button, it will show the user ID profile information, for the ambulance, it will show the medical record of the user.

Table 6.4 Testing scenario

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Scenario** | **Expected Result** | **Result** |
| 1. | Police Button | Displaying longitude and latitude of the user location and the contacting status as label and send location data to database in real-time | As expected |
| Internet connection on |
| GPS on |
| 2. | Ambulance Button | Displaying the contacting status as label and send location data to database in real-time | As expected |
| Internet connection on |
| GPS on |
| 3. | Fire fighter Button | Displaying the contacting status as label and send location data to database in real-time | As expected |
| Internet connection on |
| GPS on |
| 4. | Any Button | Displaying pop up to activate GPS first which direct to location setting if user click ok | As expected |
| Internet connection on |
| GPS off |
| 5. | Any Button | Displaying pop up to activate mobile data | As expected |
| Internet is off |

### Widget Button

There are 3 types of the widget which call 3 different Public Officer as well. This button works on condition that the user already registered, GPS is on, and Internet on also.

Table 6.5 Widget scenario

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Scenario** | **Expected Result** | **Result** |
| 1. | Police widget | Scan location, scan nearby active police, and then send notification, profile information, and location to the nearby police who respond the notification. | As expected |
| Internet on |
| GPS on |
| 2. | Ambulance widget | Scan location, scan nearby active ambulance, and then send notification, patient information (registered hospital and patient ID) and location to the nearby stand by ambulance who responds the notification. | As expected |
| Internet on |
| GPS on |
| 3. | Fire fighter widget | Scan location, scan nearby active fire fighter, and then send notification and user’s location to the nearby stand by fire fighter who responds the notification. | As expected |
| Internet on |
| GPS on |
| 4. | Any widget | Show notification to activate GPS first and direct to GPS setting page. | As expected |
| Internet on |
| GPS off |
| 5. | Any widget | Show notification to activate mobile data first and direct to mobile data setting page. | As expected |
| Internet off |

### Cancel Menu (on homepage)

There is a cancel menu on the homepage menu set, which will cancel current contact the user made if there is a contact. If there is no contact has been made yet, there will be a message saying that there’s no contact yet.

Table 6.6 Cancel scenario

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Scenario** | **Expected Result** | **Result** |
| 1. | Contact event activated | Contact canceled | As expected |
| Cancelation menu clicked |
| 2. | Contact event activated | Contact event still running | As expected |
| Cancelation menu not clicked |
| 3. | Contact event not active | Show message that there is no contact yet | As expected |

### Maps view (Public Officer Activity)

This page can only be access by the Public Officers as their job is to find the general user who called them and give them assistance.

Table 6.7 Map scenario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Actor** | **Scenario** | **Expected Result** | **Result** |
| 1. | Police | Internet on | Map appears with the targeted use being on point. The User profile also can be access through menu that appears if there is contact. | As expected |
| GPS on |
| Receive location |
| 2. | Ambulance | Internet on | Map appears with the targeted Public Officer being on point. The User profile also can be access through menu that appears if there is contact and the User’s registered patient ID on certain hospital also appears on other menu. | As expected |
| GPS on |
| Receive location |
| 3. | Fire Fighter | Internet on | Map appears with the targeted Public Officer being on point. | As expected |
| GPS on |
| Receive location |
| 4. | Any actor | Internet on | Cannot receive location | As expected |
| GPS off |
| 5. | Any actor | Internet off | Cannot receive location | As expected |

### Request done

When the request is done, the Public Officer can click the request done on their menu.

Table 6.8 Request done scenario

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Scenario** | **Expected Result** | **Result** |
| 1. | Public Officer click the complete button. | The event is done and Public Officer is ready to take another request | As expected |

# 

**CONCLUSION**

## Conclusions

This thesis aims to provide a new way of utilizing the combination of home screen widget and mobile GPS function in order to share user’s real-time location with a button on user’s home screen. The goal of the thesis has been successfully achieved since all the function are working as expected.

The system used to make the real-time data sharing possible is the utilization of Google’s Firebase which taking a big part to store data such as location, profile data, and storing image. Can be concluded that Firebase is suitable to be a cloud service for any online application that needs online storage because of the easy development Firebase offer, since Android Studio support Firebase even has the Firebase Assistant to make it more efficient.

From the research for android home screen widget, it can be concluded that there are still a few of application that utilize home screen widget to the fullest capability. Most of the application are only use the widget to open the main application or only to show data (for example: clock, weather, etc.).

For the location scanning, it appears that the accuracy of GPS also affected by the internet network intensity.

## Future Work

Several things can be accounted as an improvement in the near future in order to enhance the use of this project:

* 1. Improve the user interface of the application

Improving the user interface using the method that designer user like color theory, layout theory, and else in order to help user understand the application and ease them to user the application. Maybe make the widget button more eye-catching without making it too animated.

* 1. Develop multi-platform application

To maximize the utilization of this application, not to be discriminative to other platform. Not only for android, later has had had the application had to make a way to reach to be in apple platform and windows platform.

* 1. Validate user’s ID card and Public Officer member ID by accessing government database

The database this thesis use is just temporary and limited. In order to really check the user’s identity, this application needs access to government database of citizen ID and classified data of special unit’s member ID.

* 1. Call and message system

If implemented, this system can help Public Officer to interact directly to general user.

* 1. Heat map

If possible, this will helps firefighters to determine the area effected by fire and prevent the spreading of fire more effective.

* 1. Video call

For the ambulance, it will be more effective if there is a first aid button which will provide a video call function to help user do the first aid before ambulance arrive.

* 1. Bluetooth Low Energy (BLE)

Bluetooth Low Energy will effectively determine the user’s location in indoor area. The technology utilize the Beacon devices installed on the building. It works in a way user’s device detect the signal from the beacon then calculate roughly the distance to the beacon and hence the estimate location.

# References

|  |  |
| --- | --- |
| [1] | C. P. Puri, Agile Management: Feature Driven Development, Global India Publications, 2009. |
| [2] | J. B.-Y. Tsui, "Fundamentals of Global Positioning System Receivers: A Software Approach," in *Fundamentals of Global Positioning System Receivers: A Software Approach*, John Wiley & Sons, 2005, p. 352. |
| [3] | A. El-Rabbany, "Introduction to GPS: The Global Positioning System," in *Introduction to GPS: The Global Positioning System*, Artech House, 2002, p. 176. |
| [4] | F. S. T. V. Diggelen, "A-GPS: Assisted GPS, GNSS, and SBAS," in *A-GPS: Assisted GPS, GNSS, and SBAS*, Artech House, 2009, p. 380. |
| [5] | W. Shi, "Principles of Modeling Uncertainties in Spatial Data and Spatial Analyses," in *Principles of Modeling Uncertainties in Spatial Data and Spatial Analyses*, CRC Press, 2009, p. 432. |
| [6] | B. Williams, "Latitude and Longitude," in *Latitude and Longitude*, Smart Apple Media, 2002, p. 32. |
| [7] | M. Beaulieu, "Wireless Internet Applications and Architecture: Building Professional Wireless Applications Worldwide," in *Wireless Internet Applications and Architecture: Building Professional Wireless Applications Worldwide*, Addison-Wesley Professional, 2002, p. 632. |
| [8] | B. Stonehem, Google Android Firebase: Learning the Basics, First Rank Publishing, 2016. |
| [9] | L. Moroney, "Build Android Apps on Google's Mobile Platform," in *The Definitive Guide to Firebase*, Apress, 2017, p. 275. |
| [10] | M. G. Limaye, Software Testing, Tata McGraw-Hill Education, 2009, p. 523. |
| [11] | "Black Box Testing," STF, Dcember 19 2017. [Online]. Available: http://softwaretestingfundamentals.com/black-box-testing/. [Accessed 11 April 2018]. |
| [12] | "GPS Technology," [Online]. Available: http://www.geneko.rs/en/gps-technology. [Accessed 27 March 2018]. |
| [13] | "Latitude and Longitude Facts," 12 July 2016. [Online]. Available: https://www.worldatlas.com/aatlas/imageg.htm. [Accessed 27 March 2018]. |
| [14] | S. R. Palmer, A Practical Guide to Feature-Driven Development, Prentice Hall, 2002. |

1. R, Mei Amelia. “Kapolda Metro: Kejahatan Di Jakarta Terjadi Tiap 12 Menit 18 Detik.” *Detik.com*, Detik, 30 Dec. 2016, 08:39, news.detik.com/berita/d-3384009/kapolda-metro-kejahatan-di-jakarta-terjadi-tiap-12-menit-18-detik. [↑](#footnote-ref-1)
2. [*"Firepad - An open source collaborative code and text editor"*](https://firepad.firebaseapp.com/). Firebase, Inc*. June 12, 2014*, https://firepad.firebaseapp.com/ [↑](#footnote-ref-2)
3. [*"Firechat - open source realtime chat built on Firebase"*](https://firechat.firebaseapp.com/). Firebase, Inc*. June 12, 2014*, https://firechat.firebaseapp.com/ [↑](#footnote-ref-3)
4. [*"GeoFire — Realtime location queries with Firebase"*](https://github.com/firebase/geofire)*. Retrieved December 20, 2016*, https://github.com/firebase/geofire [↑](#footnote-ref-4)