PANIC MANAGEMENT SYSTEM

OMISORE, OLUWADAMILOLA OLUWASENI (17/SCI01/068)

JULY, 2021

PANIC MANAGEMENT SYSTEM

OMISORE, OLUWADAMILOLA OLUWASENI (17/SCI01/068)

A Project Submitted to the Department of Mathematical and Physical Sciences, Afe Babalola University, Ado-Ekiti in partial fulfilment of the requirements for the award of degree of Bachelor of Science in Computer Science.

July, 2021

DECLARATION

I hereby declare that this Thesis was written by me and is a correct record of my own research work. It has not been presented in any previous application for any degree of this or any other University. All citations and sources of information are clearly acknowledged by means of references.

……

OMISORE, OLUWADAMILOLA OLUWASENI

DATE……

CERTIFICATION

This is to certify that OMISORE OLUWADAMILOLA OLUWASENI, with matric number 17/SCI01/068 of the Department of Mathematical and Physical Sciences, Bachelor of Sciences in Computer Science Programme, carried out this project work and this project work has been presented elsewhere for the award of the degree or any other purpose.

_____ _____

Omisore Oluwadamilola Oluwaseni Date

(Student)

_____ _____

Dr. O. A. Adeyemo Date

(Supervisor)

_____ _____

Dr. Ajiboye Date

DEDICATION

This project is dedicated to Almighty God and to my family for the continuous support morally, spiritually, emotionally and financially.

ACKNOWLEDGEMENTS

My thanks and gratitude goes to God Almighty for his guidance and infinite mercy in my academic journey.

Special appreciation also goes to my project supervisor, the provost of the college of sciences , Dr. Adeyemo O. A., for his time, effort, support and guidiance in this project and all through my studies in this institution. Appreciation also goes to my lecturers, all teaching and non-teaching staff who contributed positively to my stay in this institution.

I would like to specially thank my parents and my siblings for all their support and care.

I would also like to thank my friends for their constant support and motivation throughout my studies in this institution.

LIST OF FIGURES

Figure 2.1: Speech Recognition Process…….9

Figure 2.2: Interfaces of FEMA application……...……23

Figure 2.3: bSafe main interface and setting of fake call ……...…….……24

Figure 2.4: Setting emergency contact list and a sample of the SOS message ……..……..……25

Figure 2.5: Setting of passcode and shaking pattern……...……26

Figure 2.6: The Panic Button and the trigger options …….……26

Figure 2.7: Main screen of RapidSOS and helping friend request emergency help ……27

Figure 2.8: Check-in to inform user's family and detect current location to request help……...28

Figure 2.9: Option for type of emergency when requested as shown by the left, nearby agencies as shown in the middle, and request SOS as shown by the right……..29

Figure 2.10: Interface of SOS Siren/Whistle, the whistle is triggered or stopped by tapping the green area……...30

Figure 2.11: Main Interfaces for HelpMe application…….31

Figure 2.12: Interface for GreatCall application…….32

Figure 2.13: Main Interface for CFA FireReady application……..33

Figure 2.14: Main Interface for GoSuraksheit application……..34

Figure 2.15: SOSPhone mobile application screenshots…….37

Figure 3.1: Phases in incremental methodology……38

Figure 3.2: Panic Management System Architecture diagram……42

Figure 3.3: Location Tracking Use Case Diagram of Panic Management System……..45

Figure 3.4: Emergency Message Sending Use Case Diagram of Panic Management System…….46

Figure 3.5: Contact Setting Use Case Diagram of Panic Management System……46

Figure 3.6: Register account activity diagram of Panic Management System……47

Figure 3.7: Login activity diagram of Panic Management System…….48

Figure 3.8: Request SOS activity diagram of Panic Management System……..49

Figure 3.9: Settings activity diagram of Panic Management System……..50

Figure 3.10: Panic management system block diagram……..51

Figure 3.11: Panic management system flow chart diagram……...52

Figure 4.1: Home screen of the Panic Management System……...55

Figure 4.2: Switch activity and the respective feedbacks……56

Figure 4.3:Login screen of the Panic Management System……57

Figure 4.4: Login process and the feedback afterwards on the settings screen……57

Figure 4.5: Registration screen of the Panic Management System…….58

Figure 4.6: Registration process and feedback on the login screen after registering……...59

Figure 4.7: Forgot Password screen of the Panic Management System……..60

Figure 4.8: Resetting password process and the feedback on the login screen……60

Figure 4.9: Settings screen of the Panic Management System……61

Figure 4.10: Keywords setting screen of the Panic Management System……...62

Figure 4.11: Keywords setting process and the feedback afterwards on the home screen……...62

Figure 4.12: Contact-setting screen of the Panic Management System……...63

Figure 4.13: Contact and message setting alongside the feedback afterwards on the home screen…….64

Figure 4.14: Maps fragment for the Panic Management System…….…65

Figure 4.15: Marked location on the maps fragment……...65

TABLE OF CONTENTS

DECLARATION……..……..……i

CERTIFICATION……..……ii

DEDICATION……iii

AKNOWLEDGEMENTS……...……..……...……..iv

LIST OF FIGURES……...……v

TABLE OF CONTENTS……..vii

ABSTRACT…….……..xi

CHAPTER ONE: INTRODUCTION……..1

1.1 Background Study……..1

1.2 Statement of Problem…….2

1.3 Aim and Objectives…...…….……2

1.4 Scope of the Project……..……3

1.5 Methodology…….…….3

1.6 Contributions to Knowledge…….…….4

1.7 Definition of Terms……...…….……5

CHAPTER TWO: LITERATURE REVIEW…….……7

2.1 An Overview of Speech Recognition……..……7

2.1.1 Types of Speech Recognition……..……...8

2.1.2 Speech Recognition Strengths……...……9

2.1.3 Speech Recognition Weaknesses……..…….…..10

2.1.4 Factors on the Speech Recognition…..…….....11

2.1.5 Fundamentals of Speech Recognition……...11

2.2 Android……13

2.2.1 Why Android……15

2.2.2 Android Strengths…….16

2.2.3 Android Weaknesses…….17

2.2.4 Layouts……..18

2.2.5 Google Maps Android API v2……..19

2.2.6 Intents……....20

2.2.7 PHP……...21

2.2.8 JSON…….21

2.3 Global Existing Emergency Systems……..22

2.3.1 Federal Emergency Management Agency(FEMA)…….23

2.3.2 bSafe…….….24

2.3.3 SOS- Stay Safe!.....25

2.3.4 RapidSOS+……27

2.3.5 SaveME 999……28

2.3.6 SOS Siren/Whistle……..29

2.3.7 ELERTS…….30

2.3.8 HelpMe……31

2.3.9 GreatCall…….32

2.3.10 CFA FireReady…….32

2.3.11 GoSuraksheit……34

2.3.12 Emergency Medical Service (EMS)……35

2.3.13 Remote Panic Button…….35

2.3.14 GPS and GIS for Ambulance…….36

2.3.15 SOSPhone System…….36

CHAPTER THREE: METHODOLOGY…….38

3.1 Incremental Methodology……38

3.1.1 Phases in Incremental Methodology…….38

3.2 Requirement Analysis of Panic Management System……..39

3.2.1 Major functional requirements for Panic Management System……40

3.2.2 Non-functional requirements for Panic Management System……..40

3.3 Requirement Specifications…….....41

3.3.1 Software Requirements…….41

3.3.2 Hardware Requirements……42

3.4 System Design……..42

3.4.1 System Architecture……..42

3.4.2 System Use Case Diagrams……...45

3.4.3 System Activity Diagrams……47

3.4.4 System Block Diagram…….51

3.4.5 System Flow Chart Diagram……52

CHAPTER FOUR: SYSTEM IMPLEMENTATION…….53

4.1 Software Used……..53

4.1.1 Android Studio……..53

4.1.2 Android SDK…….53

4.1.3 Android Emulator……..54

4.1.4 Lucid Chart……54

4.2 Graphical User Interface (GUI)……55

4.2.1 Home Screen…….55

4.2.2 Login Screen……..56

4.2.3 Registration Screen……58

4.2.4 Forgot Password Screen……59

4.2.5 Settings Screen……..61

4.2.6 Keywords Settings Screen……61

4.2.7 Contact Setting Screen……..63

4.2.8 Maps Fragment Screen……..64

4.3 Testing Methods……...66

4.3.1 Unit Testing……...66

4.3.2 Integration Testing……66

4.3.3 Acceptance Testing……...…67

CHAPTER FIVE: SUMMARY, CONCLUSION AND RECOMMENDATION…….68

5.1 Summary……..68

5.2 Conclusion……69

5.3 Recommendations……69

REFERENCE…..……...……....71

APPENDIX……78

ABSTRACT

The aim of the project is to develop an Android application that lets its users to send notifications in case of any emergency or a panic situation. The drawback of the normal SOS system is that it does not specify the classification of the emergency.

This system is limited to only android-based devices, which is better considering android devices are affordable. The system development life cycle applied for the development of this system will be the Incremental Methodology.

The Panic app is a collection of Activities and Fragments that are presented to the user. These Activities and Fragment have associate XML files (Layouts) declared in the layout folder, which determine the graphical interface for these components. Testing is carried out on the entire project. PANIC app was my first attempt at an Android application. It gave me very good exposure to the Android platform and mobile development in general.

CHAPTER ONE

INTRODUCTION

1.1 Background

In this era of crime, personal safety cannot be guaranteed. There are many emergency cases recently such as robbery, burglary, snatching, rapping, etc and these cases seriously affect social stability and could reduce people stepping out their doors. Furthermore, everyone is worrying about their personal safety and also their loved ones, especially their family and beloved partner. For some of the people, they tend to worry about other people's safety more than they worry about themselves. Nigeria has the highest level of "criminality" in Africa with a score of 7.75/10. Nigeria's criminal markets is as follows: human trafficking 8.5/10, human smuggling 7/10, arms smuggling 7.5/10, Illegal logging 7.0/10, illegal wildlife trafficking 8.0/10, non-renewable resources 9.0/10, and drug trafficking such as heroin trade 6.5/10, cocaine trade 6.5/10, cannabis trade 8.0/10 and synthetic drugs trade 8.5/10 (OC Index, 2019).

As much as we would like to get rid of them, panic or emergency situations are unavoidable and usually unpredictable. The nature and consequences of these situations can vary significantly and in worst cases also be life threatening. Therefore, it would be really nice to have some mechanism by which we can notify certain people about such circumstances and increase the chances of receiving help as soon as possible. The need for such a mechanism increases even more as in this era of technology, platforms exist to support them. One such platform and a very common one in that is a Smartphone. Almost everyone today carries a Smartphone with them as they become more and more affordable and easily available. Also within the Smartphone market Android is the clear leader in terms of market share. According to one report, 78.1% of the total Smart-phones sold in

2013 were the Smartphones that run on Android Operating System (IDC, 2014). Hence, developing an Android application becomes a better choice.

1.2 Statement of the Problem

According to the official data (Shahrudin, 2016), crime index and public fear both shows an upward trend. Public fear of crime is referring to the fear that citizens have as they may be one of the next victim of crime. Hence, some families or parents have limited their children and elderly ones from stepping out the door. Besides that, when people wants to hangout for gatherings or to release stress and relax, they have to worry about their personal safety including their friends. There are a lot of applications available throughout many countries but only a few in Nigeria. This application is to reduce the criminal cases and improve personal safety purpose, but there is still some dissatisfaction and imperfection to the user (Anon, 2016).

A typical existing SOS system uses GPS to track an individual's location. After the emergency call, the device sends your designated contacts a text message containing details of your current location. The drawback of the normal SOS system is that it does not specify the level of emergency i.e. SOS system does not alert your contacts the type of emergency the individual is in so as to ensure proper preparations. Also there is the need for individual to make physical contact with the device being used to send the SOS which may be impossible at times (Shri and Lokesh, 2014).

1.3 Aim and Objectives

The main aim of this research work is to create an android based panic management system using speech recognition to ensure easy notification for help during states of emergency.

The objectives are:

i. to design an android based Panic system for emergency situations,

ii. to include speech recognition in the development of the system,

iii. to include an authentication method in the development of the system, and

iv. to implement (i), (ii) & (iii).

1.4 Scope of the Project

This system is limited to only android-based devices, which is better considering android devices are affordable. This system is only limited to the use of English Language, precisely British English is required for the speech recognition considering English Language is widely used. The system is limited to one pre-set word per situation, which would be defined by the user. Considering how victims may not have the opportunity to speak out for long when in an emergency situation, the victim is limited to using the instance of opportunity wisely and clearly. An adapted architecture from an existing system is also used along with an incremental methodology.

1.5 Methodology

The SDLC applied for the development of this system will be the Incremental Methodology. This is a software development process where all the requirements are broken down into various modules of software development cycle. Each process development of this proposed application is based on each repeated authorization starting with requirements, design, coding and testing phases. Each of the subsequent releases of the application add functions into the previously released till all of the designs have been implemented. Usually, the first incremental release is a core product where all user basic requirements are attended. The phases are as follows:

- The requirements and specifications of an existing android based emergency system are collected.

- High-end functions will then be designed after the data gathered and solutions from the requirement phases are clearly defined and understood.
- The coding of the software will be carried out using some development tools and implantation methods such as: AndroidSDK, Android Studio, Java Programming Language, APIs, Android Emulator.
- Once the system is developed, it will go through the testing phase.

The proposed system will be implemented using Android Studio. An Android Speech API will also be used which provides access to the Speech Recognizer service, which converts speech into the text. We need to start Intent (RecognizerIntent. ACTION_RECOGNIZE_SPEECH) which recognizes speech through a mic dialog box. This activity converts speech into text and provides result to the calling activity. In addition, "android.speech.tts." Text To Speech provides access to Text To Speech service, which converts synthesized speech from the text for immediate playback. Users need to talk in front of the device. Voice is given as input through the mic and it converts into text by the API. Generated text available for shareable on social platforms. It is also stored on device storage on application storage or in File system.

1.6 Contributions to Knowledge

Although there are numerous emergency applications, but most of them need to be purchased to enjoy better features. For example, RapidSOS+ requires a purchase of $49.99 US dollar annually to enjoy the family packet or $29.99 US dollar to enjoy the individual packet. By using this application, users can get help immediately when they are faced problem or in trouble. This application also features with SOS emergency alert which ensures victims do not miss out on their golden opportunity to get instant help to prevent loss of life. This emergency alert can help users

escape from worst situation or even can prevent sad or unhappy cases from happening. Also considering the speech recognition function which make physical contact with phone irrelevant. This helps in critical times like when a victim is being robbed or has been involved in a damaging accident whereby he/she cannot make use of their hands. This application is suitable for all ages, either the old or young generation. For some families, parents can reduce their worries when their children or loved ones are outside. With using this application, people can be rest assured with their family. Public fear will also decrease at the same time (Mattias, 2014).

1.7 Definition of Terms

1) **API**

   Application Programming Interface; can be described as a group of pre-constructed software components that developers can combine and use for creating new software. A collection of algorithms, classes and/or data structures for e.g. performing specific tasks or communicating with other software (Mattias, 2014).

2) **DescribeFeatureType request**

   A type of request standard used by WFS clients to retrieve information about a specific layer offered by the WFS (OGC, 2013).

3) **EDCA**

   The Android application developed as a case study during this thesis project; "the Emergency Data Collector for Android™" (Mattias, 2014).

4) **EOC**

   Emergency Operation Centre, a location where emergency management leadership can gather to receive and analyse information, including spatial data, and coordinate rescue and relief efforts (Cutter 2003).

5) **GetCapabilities request**

A type of request standard and that is sent to WMS or WFS services to query the service for available layers, options and capabilities in general (OGC, 2013).

6) **GetMap request**

A type of request standard that is used for requesting map images from a WMS (OGC, 2013).

7) **GIS**

Geographic Information System; a system capable of managing and using spatial data, aiding in activities such as data collection and storage, viewing, map creation, manipulation and analysis (Mattias, 2014).

8) **GML**

Geography Markup Language, a spatial data standard (OGC, 2013).

9) **GPS**

The Global Positioning System; a system of satellites that broadcast signals which allow devices with GPS receivers to calculate their position on the Earth (Mattias, 2014).

10) **Layer**

A layer is a digital representation of a collection of physical features, such as roads, buildings, lakes etc. Each layer consists of a specific geometric type such as a Point, Line or Polygon and has common attributes, such as road length, building use category or lake area. A layer can be displayed on a map e.g. by querying a geospatial server (Mattias, 2014).

CHAPTER TWO

LITERATURE REVIEW

2.1 An overview of Speech Recognition

Speech recognition is a technology that enables a computer to capture the words spoken by an individual with the help of a microphone. These words are recognized by speech recognizer and processed, then the system outputs the recognized words. An ideal situation in the process of speech recognition is that, a speech recognition engine recognizes all words uttered by an individual however, the performance of a speech recognition engine depends on range of factors. Vocabularies, multiple users and noisy environment are the major depending factors for a speech recognition engine. When one thinks about speaking to computers, the primary idea is usually speech recognition, the conversion of an acoustic signal to a stream of words. After many years of research and analysis, speech recognition technology began to pass the threshold of practicality. The last decade has witnessed dramatic improvement in speech recognition technology, to the extent that high performance algorithms and systems are becoming available and accessible. Wide varieties of techniques with completely different levels of speech recognition are used to perform speech recognition. The speech recognition method is performed by a software component known as the speech recognition engine. The primary function of the speech recognition engine is to process spoken input word and translate it into text that an application can understand. The application can be in two different modes, which are the “Command and Control mode” some times referred as “Voice navigation and Dictation mode” (Mansour, 2018).

2.1.1 Types of Speech Recognition

Speech recognition systems can be categorized into the number of classes based on their ability to recognize words or list of words. Few classes of speech recognition are classified as under:

i. **Isolated Speech**

Isolated words typically involve a pause between two vocalizations. It does not mean that it only accepts a single word however instead it requires one utterance at a time (Mansour, 2018).

ii. **Connected Speech**

Connected words or connected speech is comparable to isolated speech but permits separate utterances with minimal pause between them (Mansour, 2018).

iii. **Continuous speech**

Continuous speech permits the user to speak almost naturally, it is also referred to as the Computer dictation (Mansour, 2018).

iv. **Spontaneous Speech**

At a basic level, it is thought of as speech that is natural sounding and not rehearsed. An ASR system with spontaneous speech ability ought to be able to handle a variety of natural speech features like words being run together, “ums” and “ahs”, and even slight stutters (Mansour, 2018).

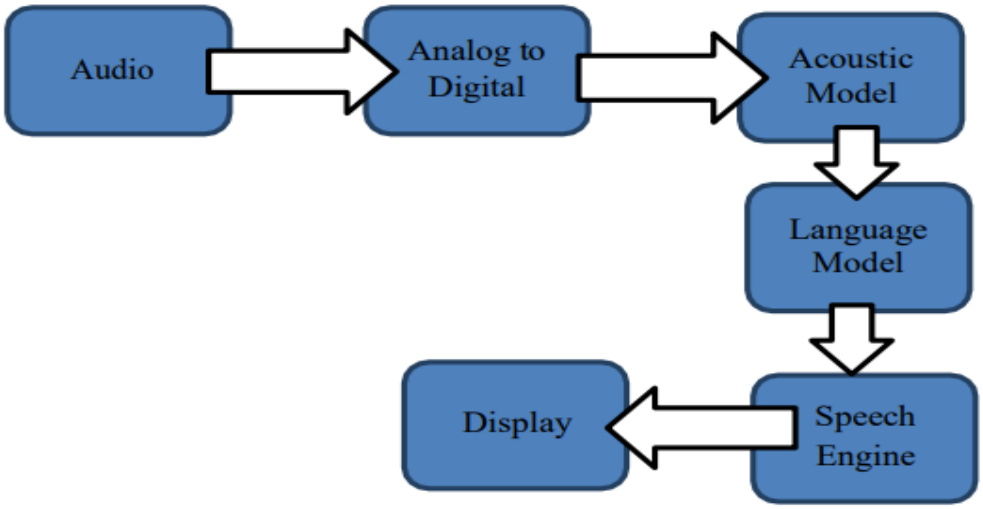


Figure 2.1: Speech Recognition Process (Mansour, 2018).

2.1.2 Speech Recognition Strengths

Speech recognition technology has become an increasingly progressive concept in recent years. From organizations to individuals, the technology is widely used for various benefits it provides. One of the most notable benefits of speech recognition technology includes the dictation ability it provides.

With the assistance of technology, users can easily control or manage devices and create documents by speaking. Speech recognition allows documents to be created faster as a result of the software generally producing words as quickly as they are uttered, which is usually much faster than an individual can type. Dictation solutions are not only employed by individuals but also by organizations that require huge transcription tasks like healthcare and legal.

Speech recognition technology also makes valuable contributions to organizations. Businesses that offer customer services gain from the technology to improve self-service

in a way that enriches the customer experience and reduces organizational costs. With the use of the voice recognition technology, callers can input information like name, account number, the reason for their call, etc. without interacting with a live agent. Instead of having callers stay idly on hold while agents are busy, organizations can engage their callers without live customer representatives. That is why speech recognition technology contributes to cost savings by minimizing or even eliminating the need for live agents whereas also improving customer experience.

2.1.3 Speech Recognition Weaknesses

Besides all these advantages and benefits, yet a fully perfected speech recognition system is yet to be developed. There are range of factors that can reduces the accuracy and performance of a speech recognition program. Speech recognition process is easy for an individual, however, it is a difficult task for a machine. Compared to a human mind, speech recognition programs seem less intelligent and this is due to that fact that the human mind is God gifted thing and the capability of thinking, understanding and reacting is natural, whereas for a computer program it is a complicated task, first it needs to understand the spoken words in relation to their meanings, and it has to make a sufficient balance between the words, noise and spaces. A human has a built-in capability of filtering the noise form a speech whereas a machine needs training, computer requires help for separating the speech sound from the other sounds (Mansour, 2018).

2.1.4 Factors on the Speech Recognition

i. **Homonyms**

These words are spelt the same and sound the same but have different meanings. For example, “there” and “their”, “be” and “bee”, “cool” and “coal”. This is a challenge for computer machine to distinguish between such types of phrases that sound alike (Mansour, 2018).

ii. **Overlapping speeches**

Another challenge in the process is to understand the speech uttered by different users, current systems have a difficulty in separating simultaneous speeches from multiple users (Mansour, 2018).

iii. **Noise factor**

The program requires hearing the words uttered by An individual distinctly and clearly. Any extra sound can create interference, first you need to place system away from noisy environments and then speak clearly otherwise the machine will confuse and will mix up the words (Mansour, 2018).

2.1.5 Fundamentals to Speech Recognition

This is basically the science of speaking with the computer, and having the words correctly recognized. To elaborate on it we have to understand the following terms.

i. **Utterance**

When user say some things, this is seen as an utterance in other words. Speaking a word or a combination of words that means something to computer is called an utterance. Utterances heard then sent to speech engine to be processed.

ii. **Pronunciation**

A speech recognition engine uses an individual's pronunciation to process words, then it give output based on what the speech engine recognizes as the sound of the word. Words can have multiple pronunciations associated with them.

iii. **Grammar**

Grammar make us of unique set of guidelines in order to define the words and phrases that are going to be recognized by speech engine. More concisely, grammar defines the domain with which the speech engine works. Grammar can be simple as list of words or flexible enough to support the diverse tiers of variations.

iv. **Accuracy**

The overall performance of the speech recognition system is measurable. The ability of recognizer can be measured by calculating its accuracy. It is useful to perceive an utterance.

v. **Vocabularies**

Vocabularies are the list of words that can be identified by the speech recognition engine. Generally, the smaller vocabularies are less difficult to

identify by a speech recognition engine, while a large list of words is a difficult task to be identified by engine.

vi. **Training**

The users who have difficulty of speaking or pronouncing certain words can utilize training, speech recognition systems with training should be capable of adapting.

2.2 Android

Android is one of the most widely used Mobile Operating System today. It is based on the modified version of Linux kernel and other open source software which is designed primarily for touchscreen mobile devices such as smartphones and tablets. Some of the most important features of an Android operating system is that it allows reuse and replacement of components, it is based on the open source Web kit engine and supports 2-D and 3-D graphics using OpenGL-ES standard. The Android operating system is implemented as a stack of different layers of software (Suryawanshi, 2010). The following are the different layers:

i) **Linux Kernel** – This is the layer at the very bottom of the Android architecture and this layer contains all the vital hardware drivers which assist to manipulate and communicate with the hardware. All other layers run on top of the Linux kernel and depend on this kernel to interact with the hardware. The Linux kernel is a free and open source, monolithic, modular, multitasking, Unix-like operating system kernel. It provides the primary

capabilities like Process management, Memory management and Device management like Camera, Display, Flash etc.

ii) **Libraries** – This is a set of common functions of the application framework that enables the device to handle different types of data. Some of the most important set of libraries are Web kit which is the browser engine to display HTML, OpenGL used to render 2- D or 3-D graphics on to the screen, SQLite which is useful for storing and sharing of application data.

iii) **Android Runtime** – The Android runtime mainly consist of the Dalvik Virtual Machine (DVM). Dalvik Virtual Machine is very much like the standard Java Virtual Machine (JVM) except that it is optimized for mobile devices that have low processing power and low memory. DVM generates a .dex file from the .class file at compile time and provides higher efficiency in low resources devices of which each application has its own process and an instance of DVM. Android runtime also provides core libraries that allows the Android developers to create applications using the Java language.

iv) **Application Framework** – These are some standard class files that are available to the developer for use. Applications can directly interact with them and make use of them. The application framework provides basic functionalities of the phone like Location Manager, Content Providers etc.

v) **Applications** – This is the topmost layer in the architecture and the layer where the application that we develop fits in. This layer provides several

pre-installed applications that are default for certain things like Contacts Books, Browser etc.

2.2.1 Why Android?

i) **Zero start-up cost to begin development with**

The tool required to develop any android applications if free of cost and Google charge very small fee deal out application in the market. (Grell, 2010)

ii) **Freedom to innovate**

Android operating system is an open source platform which is based on Linux kernel and some other open source libraries. Moreover, developers are free to build applications which runs on android devices and even free to extend platform as well. (Grell, 2010)

iii) **Freedom to collaborate**

Android developers are encouraged to share code with others and they don't even have to sign an agreement to do this. According to a survey conducted by Black Duck Software, the amount of open source libraries and mobile applications grew at a rate of 168% faster on Android compared to any other platform from year 2008-2009. (Grell, 2010)

iv) **Multi-platform support**

Android operating system is supported on several hardware devices including various phones and tablets.

v) **Multi-carrier support**

Android powered smart phones are offered by most of the carrier services.

2.2.2 Android Strengths

Google is well known and respected by technology companies all over the world. This is mainly due to the Android platform that gives it credibility and viability to the developers, potential partners and vendors. Google also in collaboration with the Open Handset Alliance (OHA) which is a group of technology companies that work together and developing an open and free Android mobile platform.

The Android platform is also integrated with a number of the Google application and services like GMAIL, GPS, Maps, Read and many others which able to reinforce Android platform capability and effectiveness. Besides, as the leader in the internet search and the YouTube, an online video browsing service, provide Android platform more competitive advantages.

Android operating system is an open-source software, it is quite different from the Apple IPhone where you only can get the hardware only from the Apple which is why hardware manufacturers choose the Android operating system of which they can apply to their handset. Therefore, consumer can choose from various brands of mobile phone manufactured, for example Samsung, HTC, etc.

Android operating system is more cost effective and highly transparent in licensing agreements. Therefore, mobile phone manufacturers are able to utilize the Android

operating system and tend to release a price for the smart phone far cheaper compared to the Apple IPhone. This will allow the increase in population of the Android operating system user. Lastly, there are just few steps then a single people become an Android application developer. The people just need to download a JAVA compiler and the Android SDK then he/she can start to developer application. Google also provide tutorials how to getting started to develop an application and there is many available source code at the website. Besides, there are many people discussing about the Android application on the internet, developer can simply share their experience with each other. This is the main reason that, Google Play Store got many powerful applications.

2.2.3 Android Weaknesses

Android operating system is just less mature compare to the iPhone operating system. Thus, the level or overall user friendliness that Apple iPhone operating system has achieved, the Android operating system is yet to achieve and still need keep improving to achieve that.

Another weakness of the Android operating system is the multimedia support. Apple has a very popular application called iTunes and marketplace, but Google does not have a centralized multimedia material and the multimedia player. This makes the Android operating system user have difficulties in downloading the multimedia or playing it.

Some of the users might go to unauthorized sites to download, it might cause infection of the viruses and other malicious attacks. Android phone always prone to external threats, this is because the friendliness of application development and the manipulation of source code by somebody that means to do something bad. Besides, many available applications in the Google Store, whether it is free or not require constant internet connection to run.

This will bring a problem for countries that do not provide internet connectivity in public places like Malaysia, etc.

Android operating system has got many brands of hardware manufacturers which leaves issues whenever Google releases a new version of Android operating system, Google has to wait for the hardware manufacturer to test and modify code to ensure that their specific hardware configuration works with no errors. This is unlike that of Apple because they have the full control of when and how the iPhone user gets the new hardware and the latest version of the operating system and core functionalities. To overcome this, Google had planned to slow down the release of the operating system, by implementing more core feature of its services into the Android operating system instead of implementing as applications.

2.2.4 Layouts

Android UI monitors are often resized, both on the fly through pinch and zoom as well as at startup when Android adjusts the scale of the UI to fit the screen size of the mobile device on which it's running. In order to make the maximum of the display size and deal with resizing gracefully, Android provides a number of screen format options. First, Android developers must specify whether or not each screen should follow a linear layout which manages controls in a horizontal or vertical style or a relative format which manages controls in terms of one another. Linear layouts are the most common, as in the instance below. At left, the controls best stretch to deal with the text and are located in a horizontal line. In the center image, the equal guidelines follow however in a vertical layout. At right, the vertical layout is maintained however the center button stretches to accommodate the screen in the place of the text (Macadamian, n.d). A relative layout defines the placement

of controls by their relationship to other components on the same screen. In the instance below from the droidcake.com blog, the “OK” button was specified to be set below the radio button group. The “Cancel” button was specified to be set to the right of the OK button with its right edge extended to the edge of the screen. This relative layout positioning guarantees the position of the buttons stays consistent throughout loads of screen sizes.

Android additionally gives unique layout properties to control the way in which screen elements are displayed throughout Android devices and during use:

i) **Weight:** The weight property allows the developer to determine how free space is divided on the screen.

ii) **Gravity:** Gravity is the term used for control alignment (right, bottom, top, or left) on an Android device.

iii) **Density independence:** Your application achieves “density independence” when it preserves the physical size (from the user’s point of view) of user interface elements displayed on screens with different densities. Without density independence, a UI element (such as a button) will appear larger on a low-density screen and smaller on a high-density screen.

2.2.5 Google Maps Android API v2

Google provides comprehensive API for developers working with Android and who want to use google maps in their application. Using this API one can easily add maps to their apps and the API automatically handles access to google maps server, data downloading, map display and response to gestures. Additionally, the API can be used to add the following on to the maps:

i) Markers that are used to show specific position on the map.

ii) Line segments (Polylines)

iii) Enclosed segments (Polygons)

iv) Various images that are shown on the map (overlays) like zoom control, compass etc.

To get started using the google maps Android API v2 one has to first obtain an API key specific to your application by using the apps signing certificate from google's API console. One should also add Google play services SDK as a project on the local disk and add it as an external library in the app (Google, 2014).

2.2.6 Intents

Android applications commonly borrow from other applications already on the device. Using intents, you could simplify both the programming requirements for your app and provide simpler, much less cluttered screens. If your app needs to perform a function past its core abilities such as opening a photo, searching for a contact, or playing a video, the team should investigate whether or not a tool that can perform that function already exists in the OS or in a popular third-party app. If so, you could leverage that functionality with the use of intents. For example, if your app accesses user contacts, you could use intent objects to launch the device's present Contacts application. This will dispose of programming duplication and speed up the user's interaction with the device since the user will not need to re-discover how to add contacts to your particular app. Android offers unique UI controls, activities, interactions, layout and resize options, in addition to unique constructs like fragments and intents. While on the surface these appear to be matters that

the design team needs to work with, we contend that the entire team must be immersed in Android to coordinate design, workflow, and execution right into a single, intuitive application, one which grabs users' interest and attracts them into the real value of your product.

2.2.7 PHP

PHP is one of the most famous server side scripting language. Php scripts can be embedded into the HTML pages. The start and the end tags for a PHP script are respectively. PHP pages can contain text, HTML, CSS, JavaScript and more. They have the .php extension and are executed on the server.

PHP can be help you get the following things done:

i) Create dynamic web pages by generating data based on user requests.

ii) Collect and process data from forms submitted by the user.

iii) Work with files on the server

iv) Manipulate some database

v) Send and receive cookies etc.

Apart from the above reasons PHP can be implemented on various platforms and is compatible with almost all servers in use. It is free and has a big support community (Suryawanshi, 2010).

2.2.8 JSON

JSON stands for JavaScript Object Notation. It is a light weight format for storing and exchanging objects that carry name/value pairs, array and other objects. It is language

independent i.e. although it uses JavaScript syntax, it is still language and platform independent and many different languages support JSON parsers and libraries. The main advantages of JSON over other ways of exchanging data like XML are:

i) JSON is usually smaller than the corresponding XML.

ii) JSON is easier to parse and faster.

For example:

```
{
    "products":[
        {"id": 1,
        "name":"car"},
        { "id":2,
        "name":"truck"}
        ]
}
```

The above JSON has two products (JSONArray) where each product is an Object (JSONObject) with keys id and name and their corresponding values (Suryawanshi, 2010).

2.3 Globally Existing Emergency Systems

In today's market, there are many similar emergency applications available and each of them have one basic objective, which is to enable personal safety. Each one has their own unique features and functionalities. Such applications are as listed below:

2.3.1 Federal Emergency Management Agency(FEMA)

This application receives real-time alerts from the National Weather Service for up to five locations nationwide. It shares real-time notifications with loved ones via text, email and social media. The application enables users to learn emergency safety tips for over 20 types of disasters, including fires, flooding, hurricanes, snowstorms, tornadoes, volcanoes and more. Also helps to access locations of open shelters and disaster recovery centers in your area where you can talk to a FEMA representative in person. It allows users to prepare for disasters with a customizable emergency kit checklist, emergency family plan and reminders as shown in Fig. 2.2. A connection is needed with FEMA to register for disaster assistance online and has both English and Spanish, which can be toggled between. A FEMA representative will not send text messages asking for personal or financial information and will never ask for money. If you suspect disaster fraud, report it immediately. FEMA will never charge you for text messages, but carrier message and data rates may apply. Your carrier may collect additional data elements; please see their privacy policies for further details (FEMA, 2009).

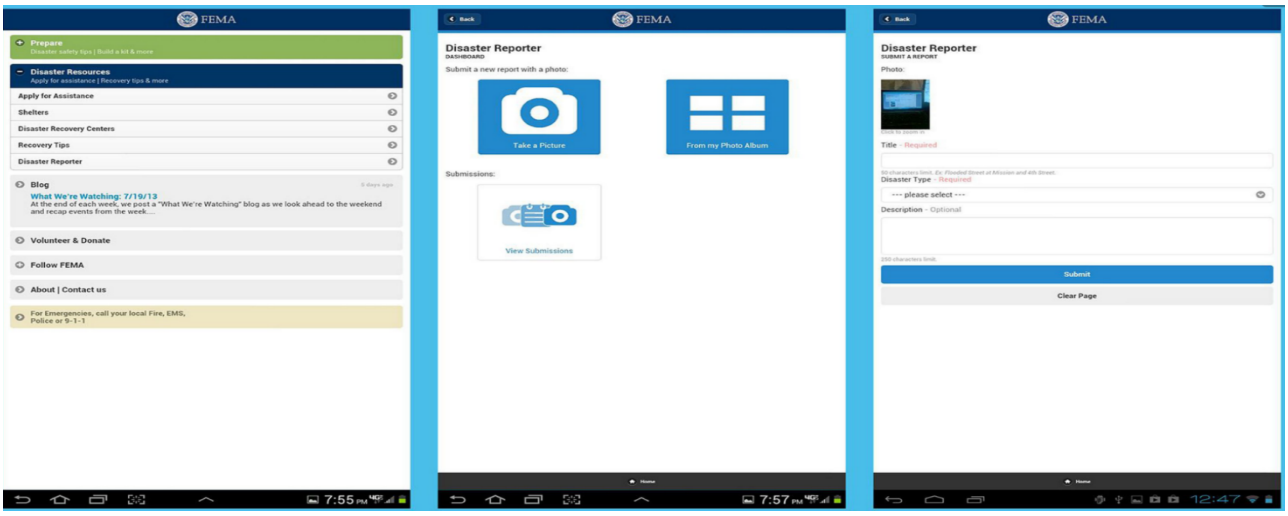


Figure 2.2: Interfaces of FEMA application

2.3.2 bSafe

This is an application that can invite friends to walk with you to prevent loneliness. It enables the setting of automated alarms to notify user so if they fail to check in after the set time which will then notify friends of where you've been and your current location as shown in Fig. 3. The application starts to record a video for 10 seconds when the SOS alarm is on. Besides that, bSafe has another interesting feature that other application doesn't have which is the fake call feature as shown in Fig. 2.3. This fake call features enables users to trigger a fake call which makes it easy for one to get themselves out of an uncomfortable or unsafe situation (Satapathy, 2012). However, this fake call feature can be an interesting feature that can be used to play with kids or to prank friends. The inviting friends to walk with user can be a weakness of this app being that it may bring user to trouble when people hack user's friend account.

Figure 2.3: bSafe main interface and setting of fake call (Richards, 2013).

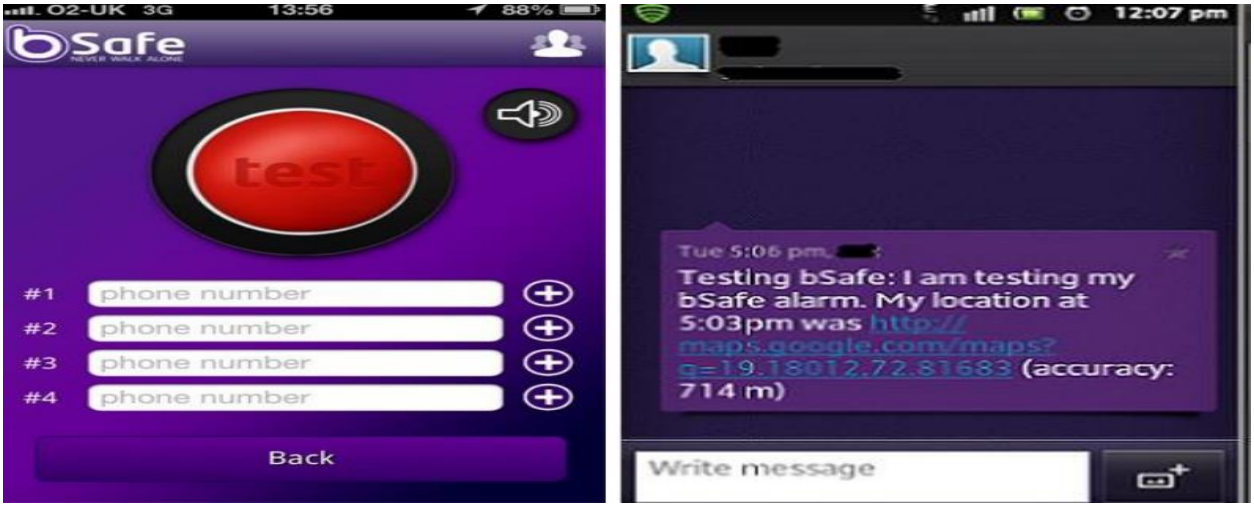


Figure 2.4: Setting emergency contact list and a sample of the SOS message (Richards, 2013).

2.3.3 SOS- Stay Safe!

This is an application that uses shaking of device to trigger the SOS alert to customize contact list. It then sends a message to the already set contact list that contains victim's mobile phone battery life, current location, time triggered and a recorded sound clip of the victim's situation. The application needs to create pin passcode of which it will be used to verify user and stop the SOS as shown in Fig. 2.5. Also, a shaking pattern is recorded and recognized (iXtentia, n.d.) to trigger the SOS as shown in Fig. 4. SOS can also be triggered by pressing the power button 3 times or by pressing the alert button in the dashboard of the application as shown in Fig. 5. A drawback of this system is error in shaking pattern where the user may make a mistake in the shaking pattern which won't match the pre-set pattern.

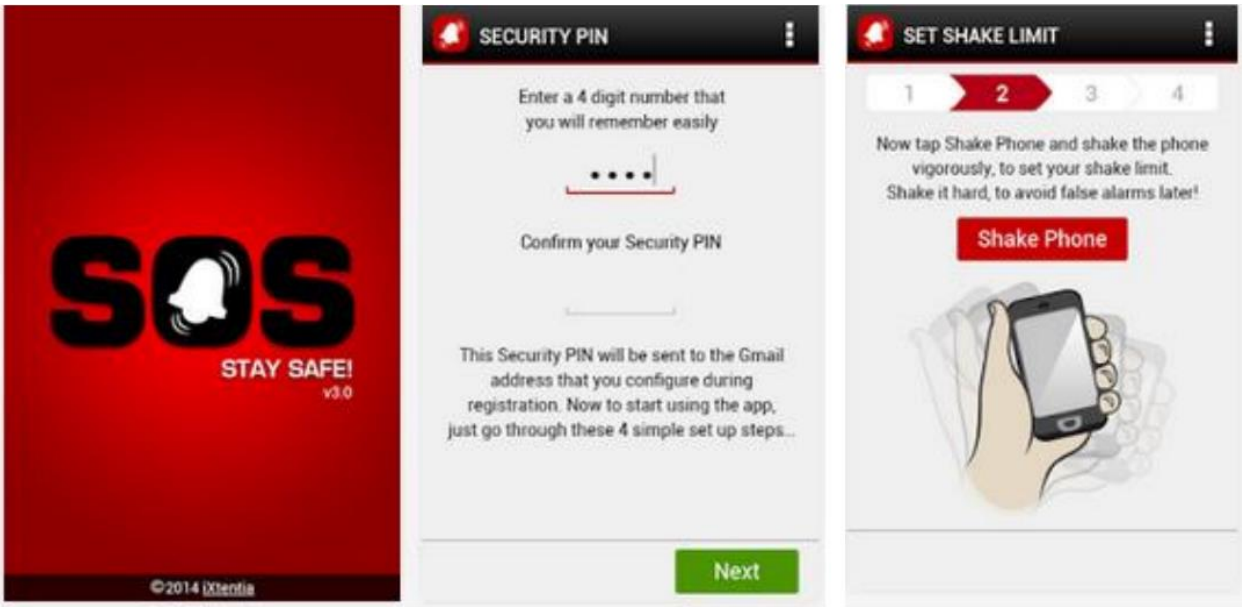


Figure 2.5: Setting of passcode and shaking pattern (Thiek, 2014).

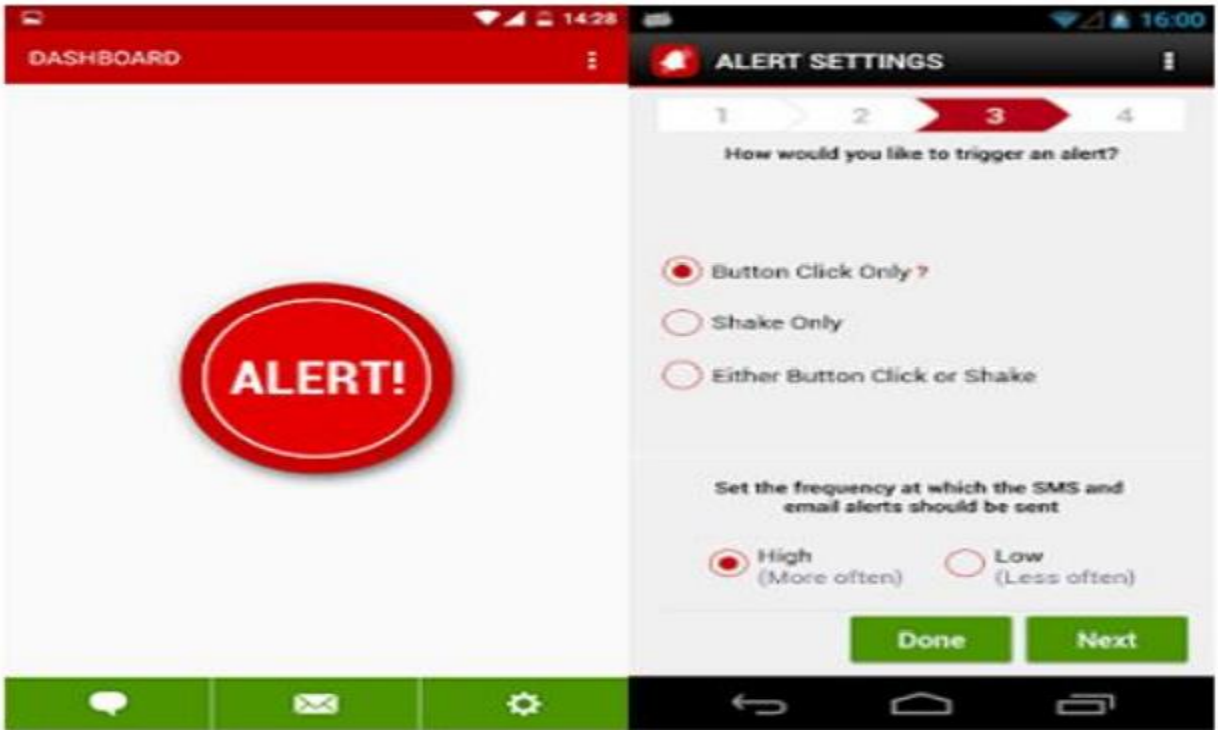


Figure 2.6: The Panic Button and the trigger options (Thiek, 2014).

2.3.4 RapidSOS+

An emergency application that was founded by three Harvard MBA students (TechCocktail, 2015). This application also provides SOS alert with one touch and it can put a call through to 9-1-1 on behalf of family or a loved one and sending their current location and information to the dispatch center closest to the victim of which their request can be dispatched if needed as shown in Fig 2.7 (RapidSOS, n.d.). Besides that, the application allows users to ensure safety of their loved ones by using GPS, and can be monitored to check if the user arrived at their destination successfully. The application is only available for 90 days free trial and the user then needs to purchase after the trial version expires.

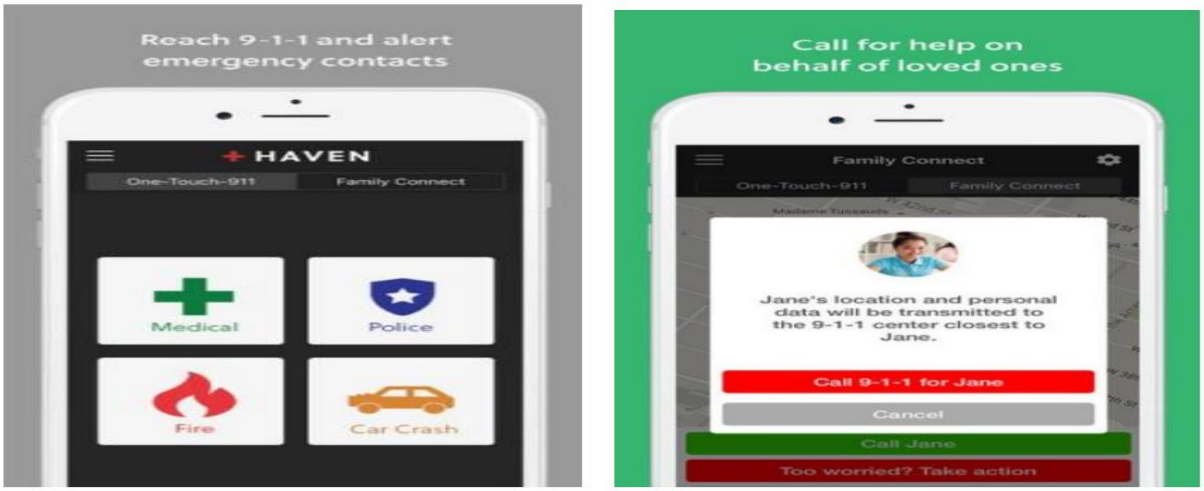


Figure 2.7: Main screen of RapidSOS and helping friend request emergency help (Nikki, 2016).

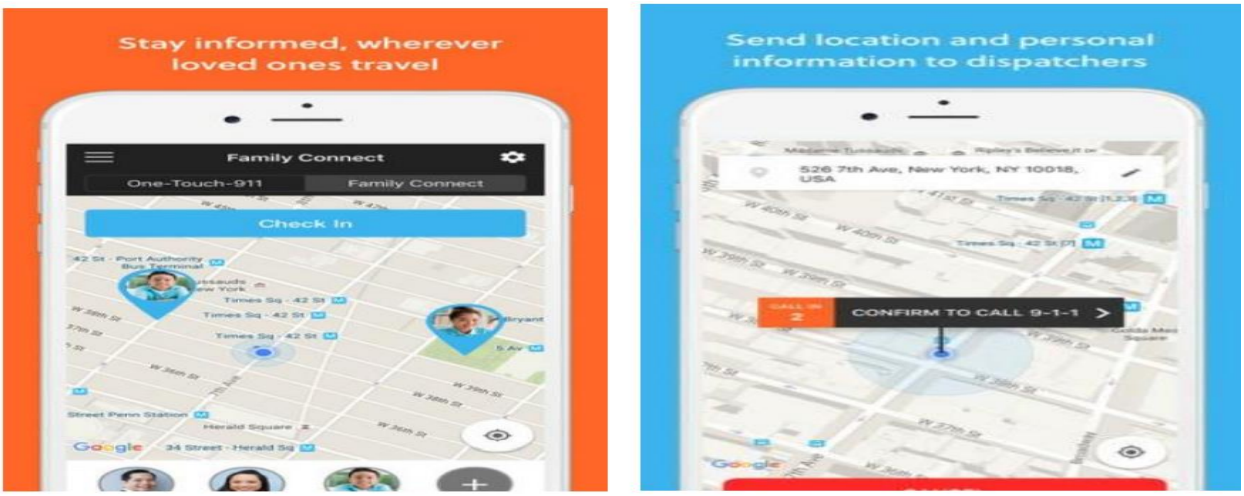


Figure 2.8: Check-in to inform user's family and detect current location to request help (Nikki, 2016).

2.3.5 SaveME 999

This application is as a result of the cooperation between Telekom Malaysia (TM) and the Royal Malaysian Police (PDRM) which is available for Malaysian citizens to allow the victims to seek for help with a panic button. When the panic button is triggered, the current location and victim's information is then sent to MERS 999 Response Centre (MERS 999 RC), if there is any additional information, it can be included together with the emergency message and will be sent to MERS 999 RC (thecinnaboy, 2016). Users can access helpline agencies through locations detected to get information and details of nearby emergencies. The application requires many steps to request for help during emergencies. While in an emergency situation, the user needs to choose the type of emergency before any request for help can be sent.

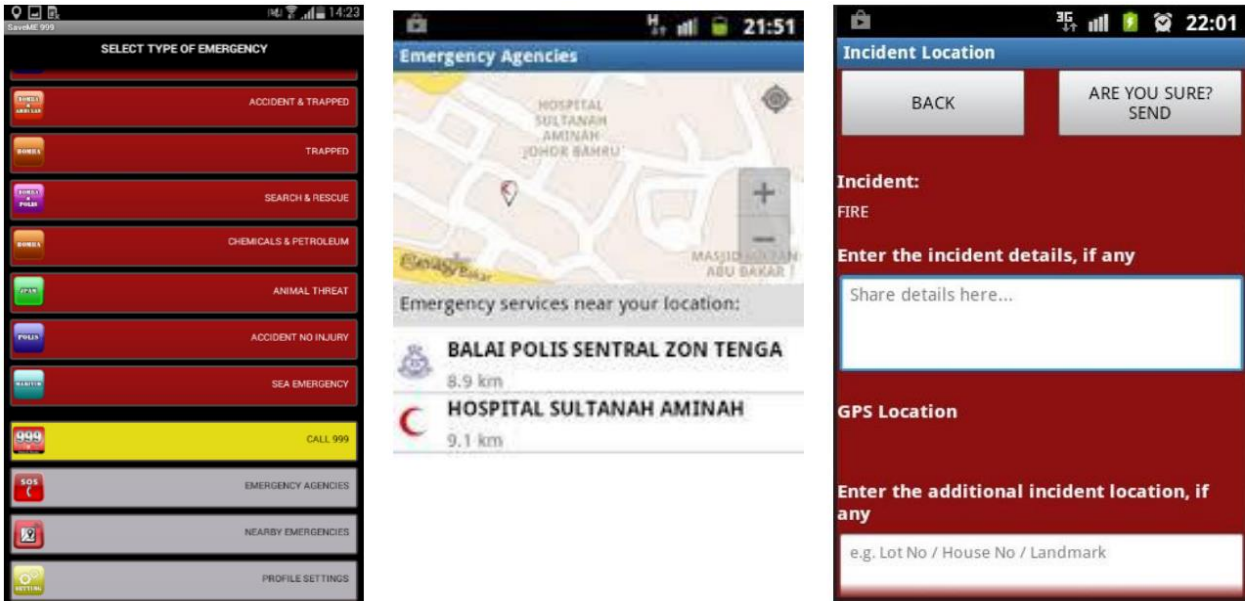


Figure 2.9: Option for type of emergency when requested as shown by the left, nearby agencies as shown in the middle, and request SOS as shown by the right (AndroidApps.Biz, 2016).

2.3.6 SOS Siren/Whistle

This is an application used to trigger an SOS whistle while in an emergency situation. As the application name suggests, the application has a button that when pressed triggers a whistle. As a matter of fact, it can trigger the whistle sound even when the device is on silent or vibration mode. This is a good application to use when in danger and to help signal other passersby of the situation. The user will need to stop the whistle by clicking the screen and there is no other function or setting added.

Figure 2.10: Interface of SOS Siren/Whistle, the whistle is triggered or stopped by tapping the green area (APKMonk, n.d.).

2.3.7 ELERTS

This is a widely used mobile application in America. The system is centralized and available in both application and web format. When using this application, the person at the emergency site can take a photo which is automatically uploaded to the website and other application users will get the alert along with the location and photo (Official ELERTS website, 2013). The application has certain features like two-way communication which enables users to reply alert messages with any other detail and information regarding

the emergency. With this feature, rescue teams or helpers can suggest the directions to get out of the situation at hand.

2.3.8 HelpMe

This is an interesting mobile application that deals with disastrous environment (Osnat, Dror, Akiva & Tomer, 2012). It brings forth the approach of building the ad-hoc network using Wi-Fi to allow smart phones to communicate during disastrous times. It doesn't require the help of any telephone providers network, but forwards the message on hop-to-hop basis. It also makes decision to forward the message by using routing algorithms. The application has a centralized HelpMe server to record everything happening about an emergency. It may not be appropriate for personal emergencies like when a person is alone or when a person is unconscious.

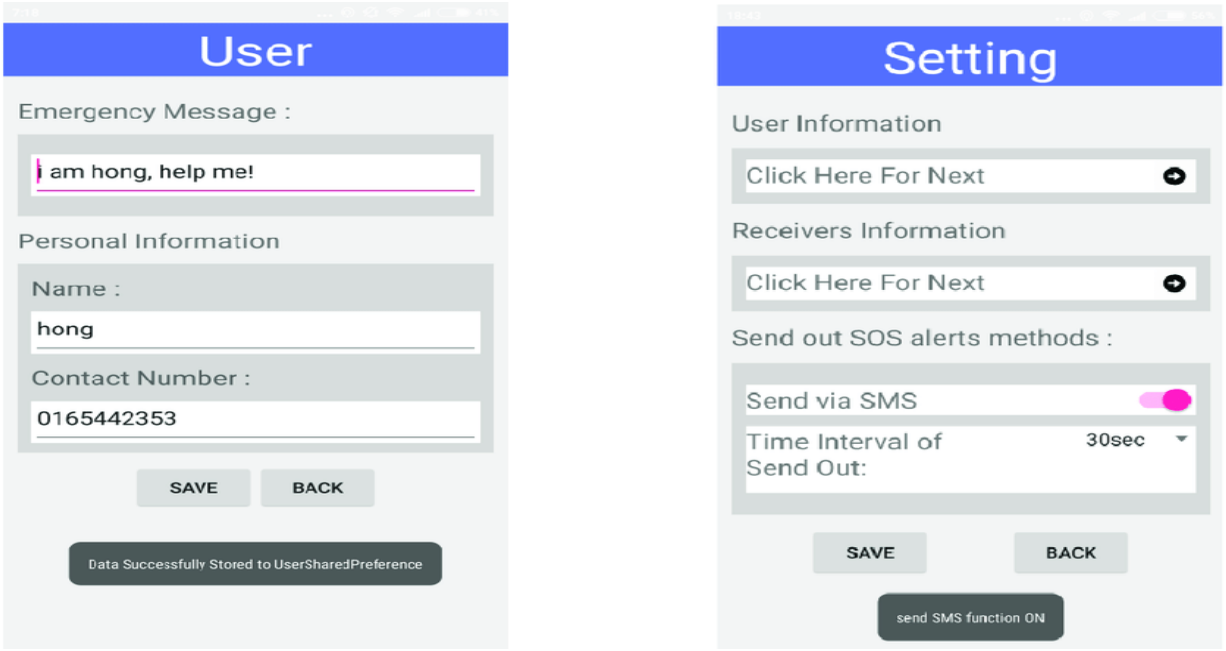


Figure 2.11: Main Interfaces for HelpMe application (Ng, Hanayanti, Norfaradilla, & Shahreen , 2017).

2.3.9 GreatCall

GreatCall is an application that gives reliable and easy-to-use wireless services (Official GreatCall website, 2013). This wireless system is very similar to a normal cell phone but is a one button device. 5star is the most used service in which an individual in emergency can talk with service representation directly upon pressing the help button. The main feature of such services is the quick response with trained agents on call and this is mostly used as a personal device advisable in the US as most citizens stay alone at home and getting noticed by someone is important during emergencies.

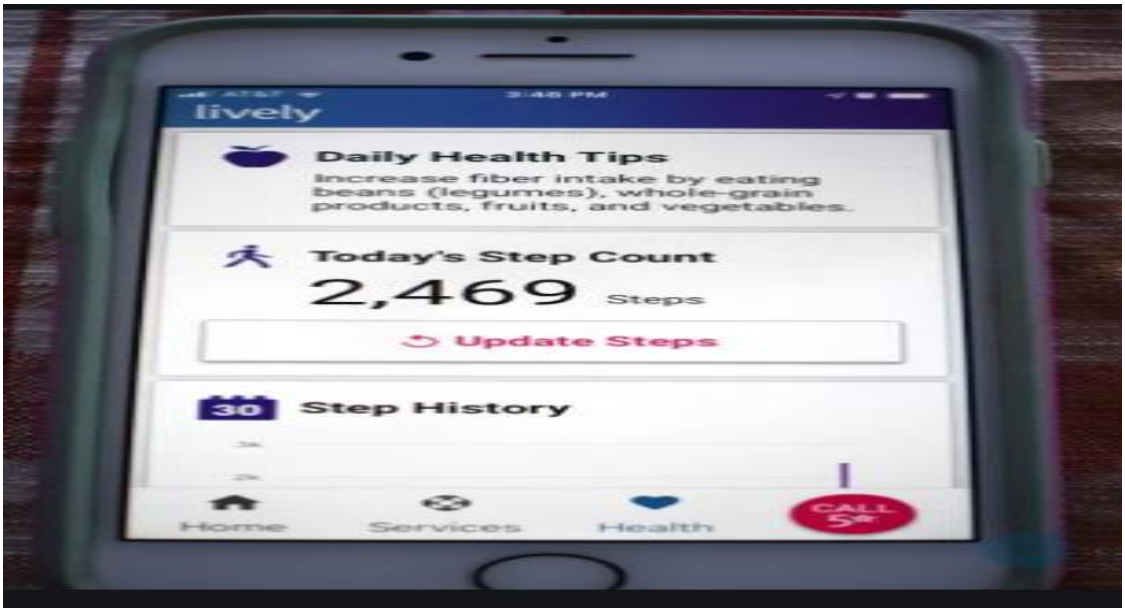


Figure 2.12: Interface for GreatCall application (Jenny, 2021)

2.3.10 CFA FireReady

The CFA released an extremely useful app called “CFA FireReady”, which is free to download for Blackberry, Android or Apple smart phones. The majority of places that hikers visit would be considered high bushfire risk areas, so it would be ignorant to go

hiking in the drier months without checking bushfire information first. With this installed on a device, staying informed barely even adds to my preparation time. It only takes one touch to open the app and within seconds a map is displayed, showing my location according to the phone's in-built GPS, the location and status of any fires. Aside from the fire map, the app also provides information on current warnings, total fire bans and fire danger ratings. Push notifications can even be set up to automatically alert the user of any fire activity in specified areas. I haven't heard of any other applications providing the same information but even if they do exist, there would be more confidence in this one since it is the only official CFA app (CFA FireReady, 2014).

Figure 2.13: Main Interface for CFA FireReady application (Kim, 2013).

2.3.11 GoSuraksheit

GoSuraksheit is a Security Application that enables you to seek for help from the contacts you trust the most from almost anywhere. This application provides you with 4 extremely strong e-weapons which you can use at just a click of a button. When you launch the application and click on the Help button it automatically sends an SMS to a list of pre-provisioned contacts (Maximum of 10). This SMS includes a customized message, the details of your current location and a link to the exact location on the map. After the help button is clicked, it updates your Facebook status with the same context as the SMS and sends a call to a pre-provisioned emergency number then triggers an alarm to seek attention for nearby help. The most differentiating part about this application is that all the above can also be done by just pressing your power button more than 4 times in case the application is installed on your phone (GoSuraksheit Android app, 2018).

Figure 2.14: Main Interface for GoSuraksheit application (Hughes, n.d)

2.3.12 Emergency Medical Service (EMS)

Emergency medical service (EMS) is a service which is responsible for leading the department in providing proper planned and organized emergency management resources which is capable of responding to public emergencies whenever it is need. (Jadhav et al., 2014). Ambulance can be categorized as a limited resources in EMS and since the congested and rapid development of urbanization and concrete jungle in each and every part of the world, the route to search and rescue for human shelter is very complex. This reflects the performance of the ambulance driver to reach the emergency spot on time (Kumar & Benedict, 2011). These issues helps to identify that the need and responsibility of an EMS is crutial and equally important to save lives. Utilizing and improvising this system helps the community to be safe and worry less on the service provided by the authority.

2.3.13 Remote Panic Button

According to (Lee et al., 2013), the remote panic button works in such way that, if the user presses the "Help Button" once, it sends an emergency triggering signal to the monitoring system and displays a message to notify the user that the remote panic button has being triggered. The controller will then read the status of the unit and sends message (SMS) to the emergency service immediately. From the article we can conclude that panic button can be used to transmit the notification or signal to ambulance when the users are alone or unable to help themselves in certain circumstances. This will help them request for help which required the use of the service of ambulance. It saves time from calling and request for help which slows down the rescue process by any means.

2.3.14 GPS and GIS for Ambulance

GPS stands for Global Positioning System is a satellite-based navigation system, the GPS Receiver retrieves the location information from satellite in the form of latitude and longitude real time reading (Pethakar, 2013). This system can be used to track real-time positions of a vehicle and smartphones which whenever the GPS setting is switched on GIS stands for Geographic Information System which is a software used for linking spatial and non-spatial data. Ambulance service in a congested area like Shah Alam, Kuala Lumpur and many major cities in Malaysia facing much constraints such as traffic lights, tolls, and devastating heavy trafficked roundabouts. If an inexperienced ambulance driver has taken a wrong route, the driver will result on late arrival on the emergency scene (Nordin et al., 2012). GPS and GIS can be implemented on the ambulance to track their location by the system archive and utilize the service of track the user and find the best route to the user as fast as possible. Integrated GPS component in the vehicles and smartphones will be useful for this project which we will be utilizing the main function in our application.

2.3.15 SOSPhone System

SOSPhone1 is an application for smartphones that enables users to contact emergency services without a voice call, which is especially helpful for deaf and earing impaired persons, but also for elderly people and in panic situations. The interaction is achieved by selecting icons (symbolizing emergency situations) and buttons (providing some associated information) in a touchscreen, according to the emergency request protocol. The result of the request is a message that is sent to the emergency services, containing a coded

description of the occurrence (enhancing communication; useful in low network conditions) along with the coordinates of the location. After sending the occurrence information to the PSAP, a communication channel can be established in order to gather further details and filter possible false alarms. A prototype of the SOSPhone system as shown in Fig. 14 was demonstrated and subject to evaluation by a large number of users, including people without any disability, emergency services professionals and deaf people (Paredes, Fonseca, Cabo & Pereira, 2013).

Figure 2.15: SOSPhone mobile application screenshots (Hugo, Benjamim, Mariam & Tania, 2013).

The development process was summarized into three iterative evaluation phases:

i) Generic user's evaluation phase

ii) Users with special need evaluation phase

iii) Operational evaluation phase

CHAPTER THREE

METHODOLOGY

Panic Management system is applying the Incremental Methodology process for the development of the application. As it is a process of software development where all the requirements are broken into different standalone modules of software development cycle. Likewise, each process development of the application is based on each repeated authorization starting from the requirements, to the design, coding and testing phases. Each of these subsequent releases of the mobile application add function in to the previous publication till all of the designed has been implemented. The mobile application is put into production after the first incremental is delivered. Usually, the first incremental release is often a core product where all user basic requirements are attended. And some of the supplementary features of the mobile applications will be added into the next incremental. Once the first core product has been done and analysed by the user. If the user is not satisfied on the core product, the next incremental is needed to fulfil user final product.

3.1.1 Phases in Incremental Methodology

The model of incremental methodology is shown in Figure 3-1-1-F1 below:

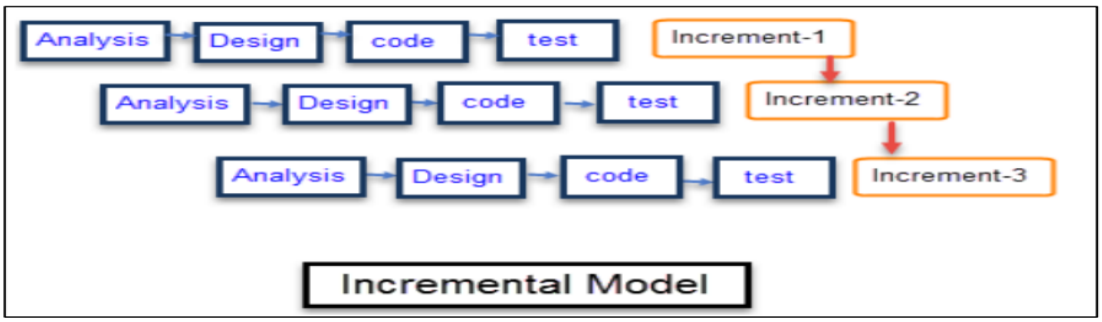


Figure 3.1: Phases in incremental methodology (Wekeat, 2018).

In each of the incremental phases, all the information about the related field is gathered, user's feedback regarding to the current existing emergency application is important to conclude the problem statement. After the review and study the existing weakness and problem, some solution will be explored to overcome and improve the existing feature to further proceed to next phases. During design phases, the data gathered and solution from requirement phases would be carried out for better understanding the problem and clearly define the objective. Each module of the application is clearly defined and design. There will be one or more prototypes will be created in order to choose the best and appropriate prototype to suit the application. The development phases will begin after the design phases. At this phase, each module porotype will be used to create and develop actual functional application. A complete version of application will proceed to final phases which is testing. Lastly, the final phases which will perform a final test about the functionality of the application, the purpose of this testing is to test there is no bugs and error occur. This is to make sure that all the features and function perform in the best situation before publish to market or user. Each iteration status will repeat this life cycle continuously until complete application is build and fully develop (Wekeat, 2018).

3.2 Requirement Analysis of Panic Management System

Requirements gathering is one of the most important phase of a software development life cycle. It is the phase that tells us what is the system supposed to do and drives the other phases in the life cycle. Requirement gathering for the PANIC app started with brain storming and discussion with other students as to what features are the most essential in a panic situation. This led to the most basic and initial draft of requirements for the application. Requirements were also collected by looking at other devices like personal locator beacons and satellite messengers that are

commercially available. A brief study of the functionality of the devices helped me to refine and narrow down the requirements even further. One important thing to learn for these devices was the simplicity of their design. This helped me to design an effective and simple UI design for the application. The next step for requirements understanding was to look for existing solutions and similar applications in the Android market. A careful study of these applications, adding other important features and removing unnecessary features was done.

3.2.1 Major functional requirements for Panic Management System

i. The user of the application should be asked to log in only the first time he uses the application on his device. The user must see the main page of the app (with the buttons to send notifications) for every other time he opens the app.

ii. The user shall be able to send notifications with the tap of a single button. Separate functions should be available for sending panic messages.

iii. An option must be provided to enable/disable these buttons to avoid pressing them by accident.

iv. The user shall be able to see their current location.

v. The user shall be able to set the contacts to send text message from within the application. Also the user may select these contacts from the predefined contacts or enter them manually.

3.2.2 Non-functional requirements for Panic Management System

i. Providing a simple and elegant UI for the main screen. This is necessary as the user would usually come on to this screen in case of a panic or emergency and hence each button should be clearly visible and easily pressed.

ii. In case the option to track location is selected and there is no internet connectivity on the device (both wireless and Cellular data), the application should be able to store the locations offline and send them to be stored in the database once the internet connectivity is up again.

iii. Providing a tab based view to display the different setting for the application and location history for the user.

iv. Displaying user friendly dialogs for picking the date, time, entering the contacts to send text messages to.

3.3 Requirement Specifications

3.3.1 Software Requirements

These requirements are separated based on the development and running of the application on a device.

For development:

- Operating System: Windows XP or higher
- Platform: Android SDK Framework 10 or higher
- Tools: Eclipse SDK 3.5, ADP plug-in for eclipse
- Technologies used: Java, Android, Google maps v2 API
- Debugger: Android Dalvik Debug Monitor Service (DDMS)
- Android Emulator: API level 14 or higher

For running on a device:

- Operating System: Android 3.0 or higher
- Cellular capabilities for SMS messages

3.3.2 Hardware Requirements

For development:

- Processor: Intel Pentium IV or higher
- RAM: 256 MB
- Space on disk: 250 MB (at the least)

For running on a device:

- Device: Phone or tablet running Android 3.0 or higher
- Disk space: 6 MB (at the least)

3.4 System Design

3.4.1 System Architecture

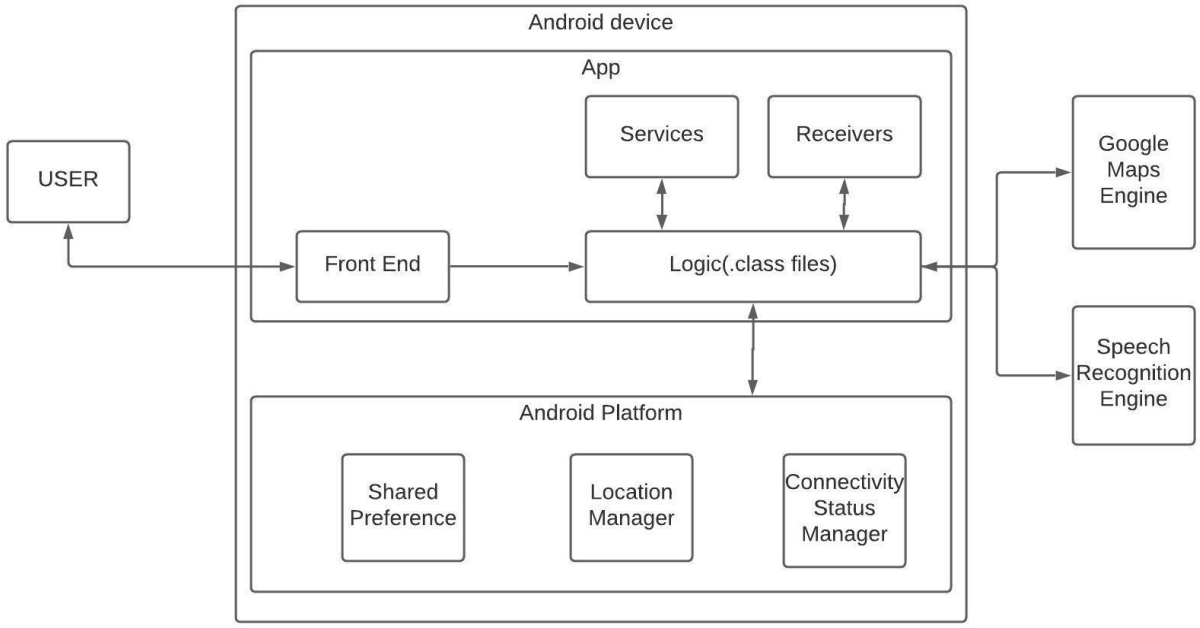


Figure 3.2: Panic Management System Architecture diagram

The different components in the architecture are:

i. **User** – This is the person who installs the application on his/her Android device. The user provides various inputs like username, password, and contact numbers etc. and triggers various events on the application (Suryawanshi, 2010).

ii. **Front End** – This is the part of the application that is visible to the user. A screen presented to the user is usually an Activity, Fragment or a Dialog Box. They contain various elements like text box or buttons to take inputs from and provide outputs to the user (Suryawanshi, 2010).

iii. **Logic:** These are the java files that contain the logic of the application. They contain various methods and classes that meet the functional requirements of the application (Suryawanshi, 2010). These files also contain code to communicate with other components in the application. For example, a file called Map.java will make use of Google maps Android API v2 to connect the Android app with Google Maps Engine to render map and markers of them.

iv. **Services**: This is the component of the application that is typically used to perform long background tasks that do not have a user interface (Suryawanshi, 2010). For example – a service is used to track the location of the device at every fixed interval of time.

v. **Receivers**: This is the component of the application that typically listens for some events or responses from other services (Suryawanshi, 2010). For example – A receiver is used to fetch the location co-ordinates from the location service and then add this location to the database for future references.

vi. **Shared Preference**: Shared preferences allow you to store small amounts of primitive data as key/value pairs in a file on the device. To get a handle to a preference file, and to read, write, and manage and control preference data, use the SharedPreferences class. The Android framework manages the shared preferences file itself.

vii. **Location Manager**: It is used to fetch the location of the Android device. The app uses both the GPS provider and the network provider to find the location for the device. GPS provides more accurate data about the location but usually takes sufficient time to start up after the connection is relinquished. Network provider on the other hand are quicker but the accuracy is lesser than GPS (Suryawanshi, 2010).

viii. **Connectivity status manager:** This system service tells the SOS application about changes in the connectivity status for the device. The application uses it to make sure that if there is no active internet connection on the device at the time of sending the fetched location to the database, it needs to store the location in the SQLite database. The application should also listen to the connectivity manager to make sure that once the internet is up again all the locations that are stored in the SQLite database are sent over to the database on the server and then the SQLite database is cleared (Suryawanshi, 2010).

ix. **Google Maps Engine**: Panic app uses google maps Android API v2 to work with maps. When this API is used, calls are made to the google maps engine to fetch the map or place various markers on it (Suryawanshi, 2010).

x. **Speech Recognition Engine**: Panic app uses speech recognizer function to perform speech recognition. When the function is called, the speech recognition engine is to process the utterance and provide result.

3.4.2 System Use Case Diagrams

The use case diagrams are used to specify the functionalities of the panic management system from the point of view of a user. Each use case describes a logical task that may be performed by the user. It mainly shows the interaction between the system and the outside world.

i. Use case for location tracking and fetching location address

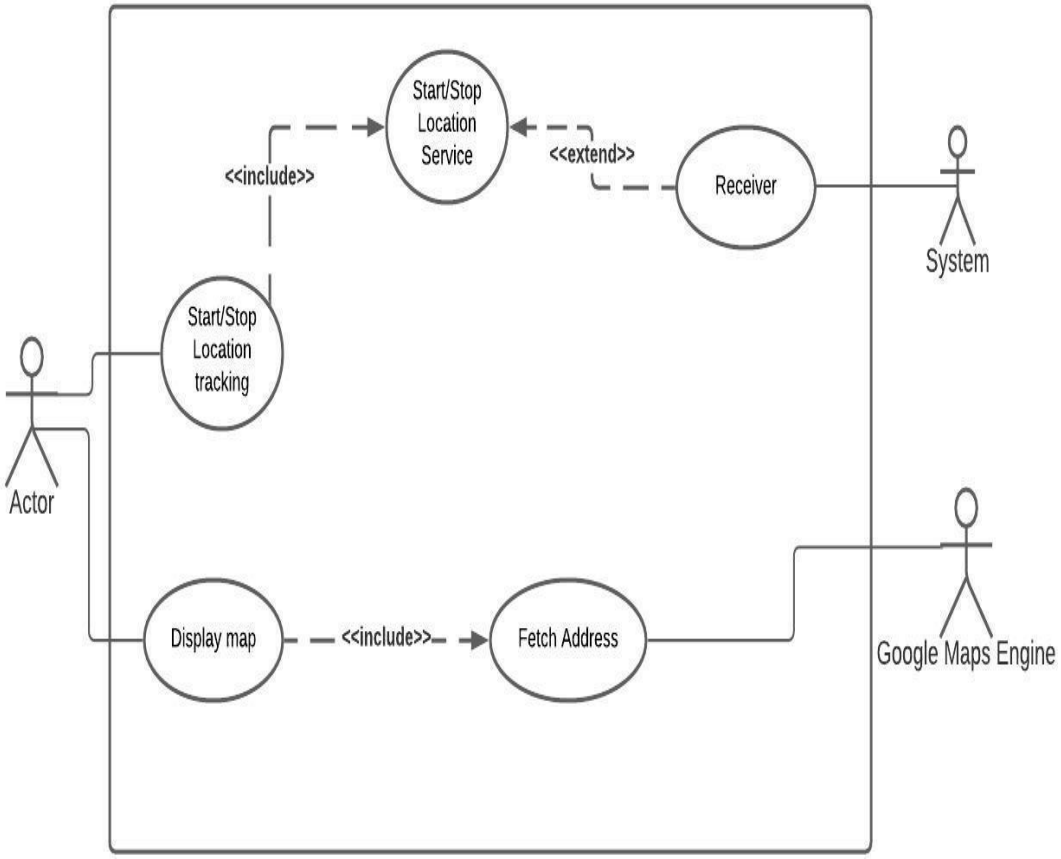


Figure 3.3: Location Tracking Use Case Diagram of Panic Management System

ii. Use case for sending emergency message

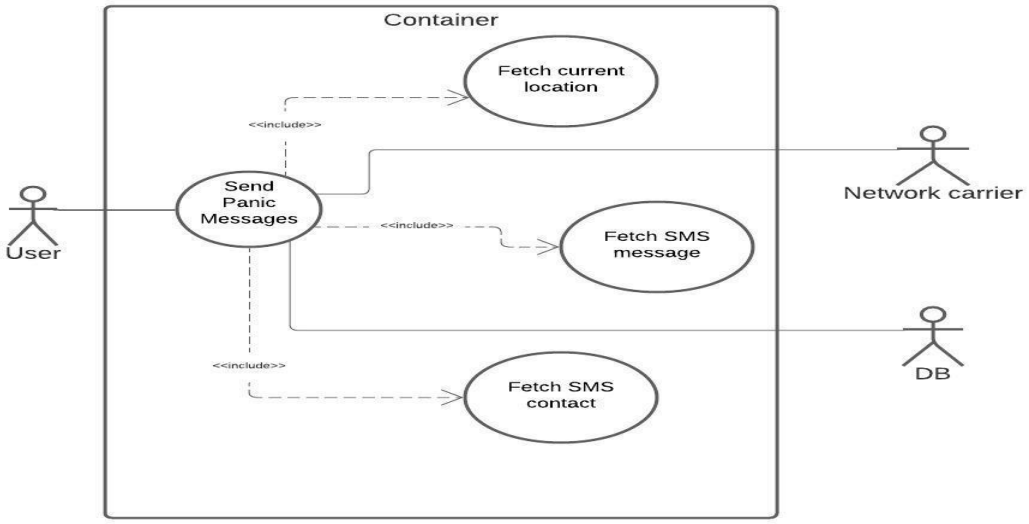


Figure 3.4: Emergency Message Sending Use Case Diagram of Panic Management System

iii. Use case for setting contacts

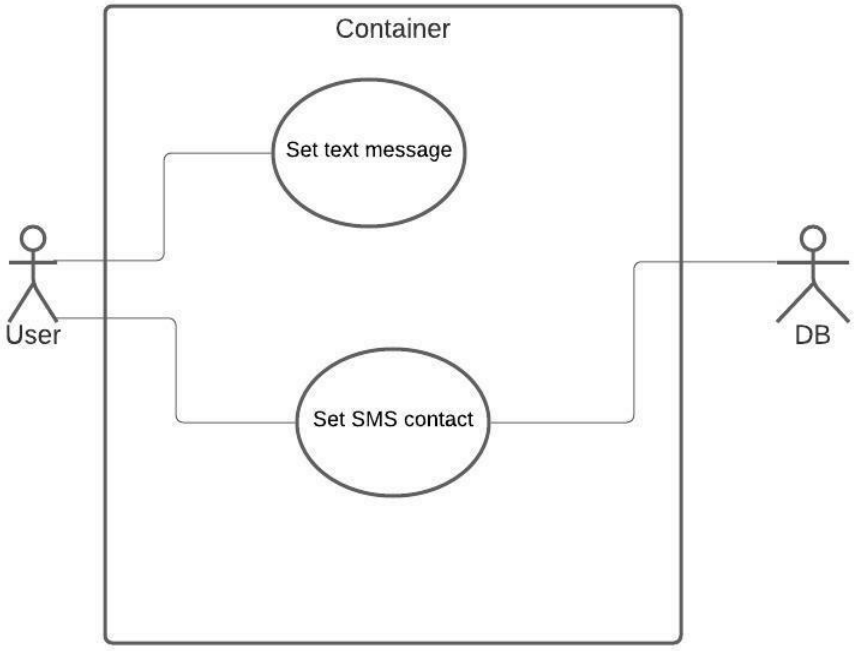


Figure 3.5: Contact Setting Use Case Diagram of Panic Management System

3.4.3 System Activity Diagrams

i. Register Activity Diagram

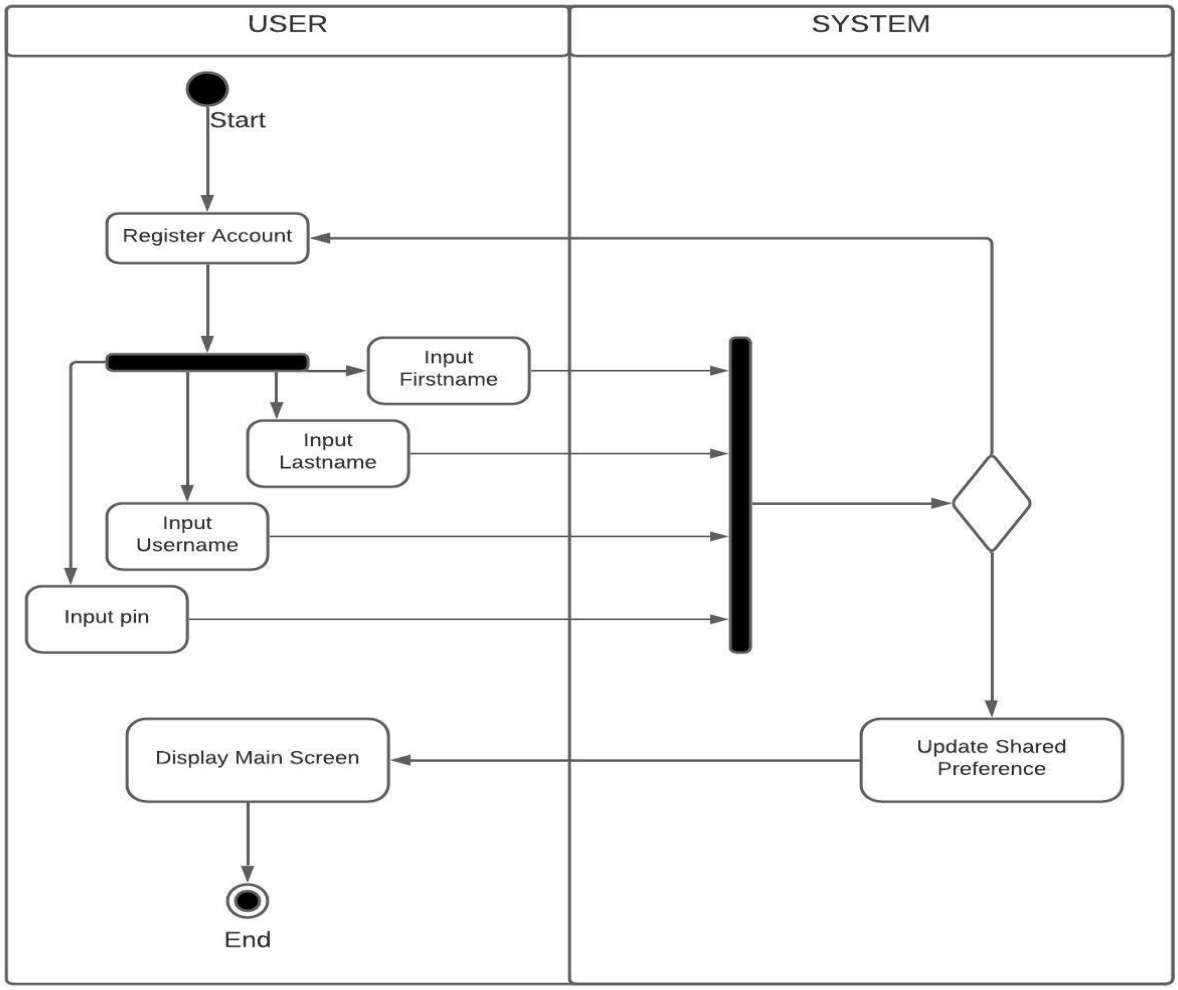


Figure 3.6: Register account activity diagram of Panic Management System

In register activity diagram, if a user is not existing member, it requires to sign up an account. User need to key their name, number, and password for registration. System will

validate if user already exists, then it will redirect to register account activity again, else, it will update firebase and redirect user to main screen.

ii. Login Activity Diagram

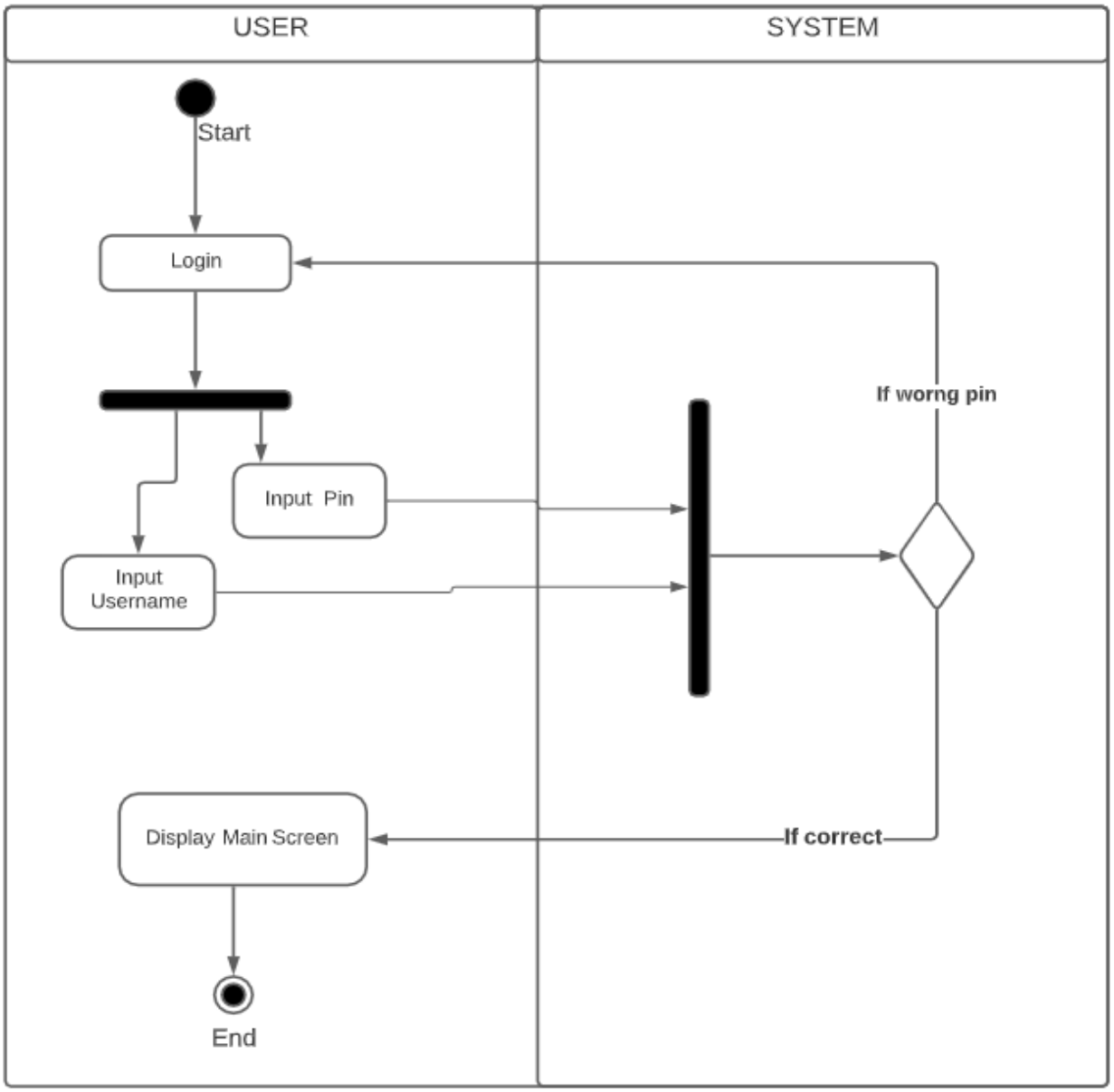


Figure 3.7: Login activity diagram of Panic Management System

In login activity diagram, after clicking login, user must enter their number and password and further for the system to validate. If there is a wrong number or password input, it will redirect to login screen again, else, it will redirect to main screen of the application.

iii. Request SOS Activity Diagram

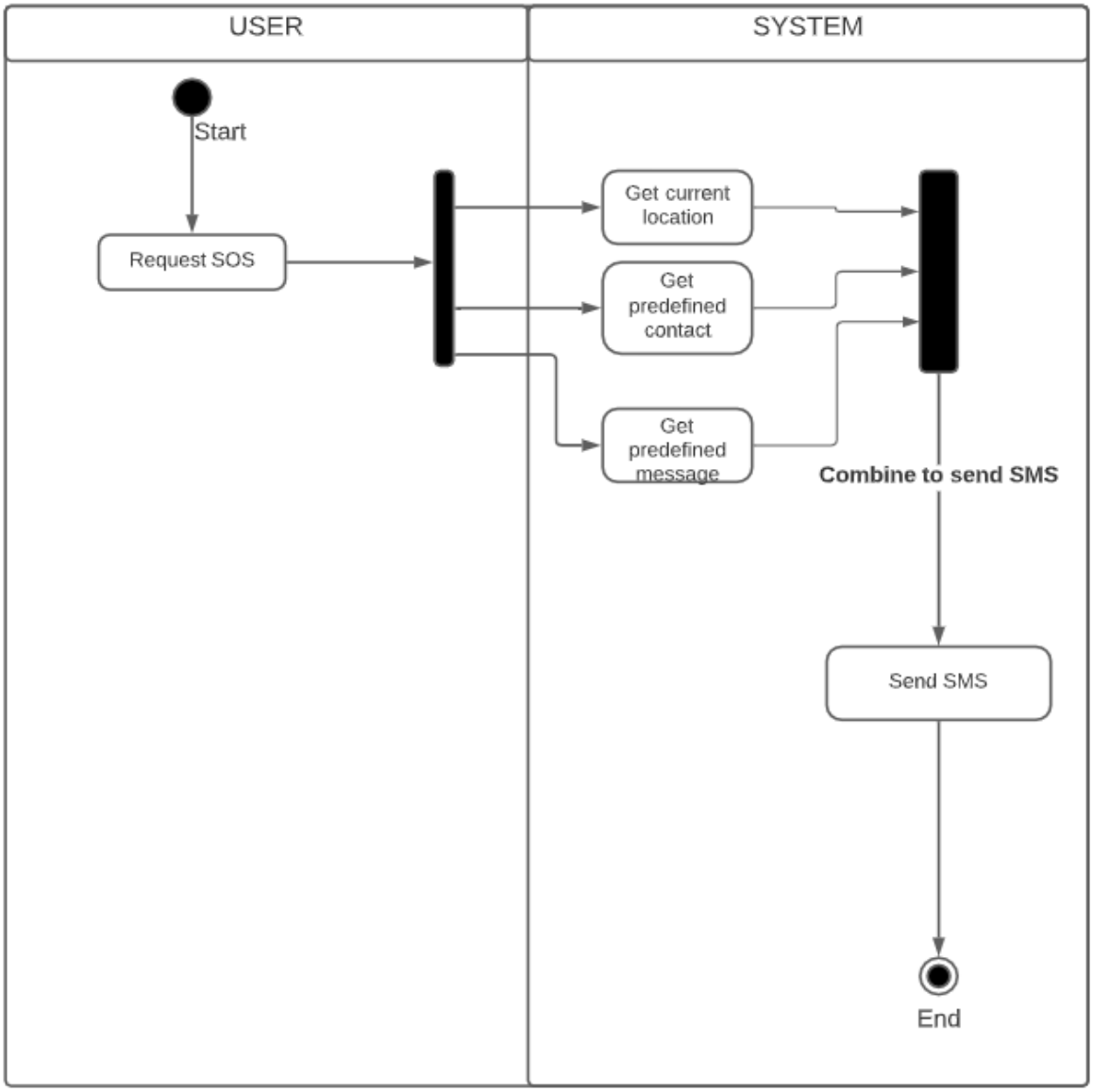


Figure 3.8: Request SOS activity diagram of Panic Management System

In request SOS activity diagram, when user requests SOS, the system will get current location and user predefine emergency contact list. With all this information, system will broadcast a SMS to their predefine contact list.

iv. Settings Activity Diagram

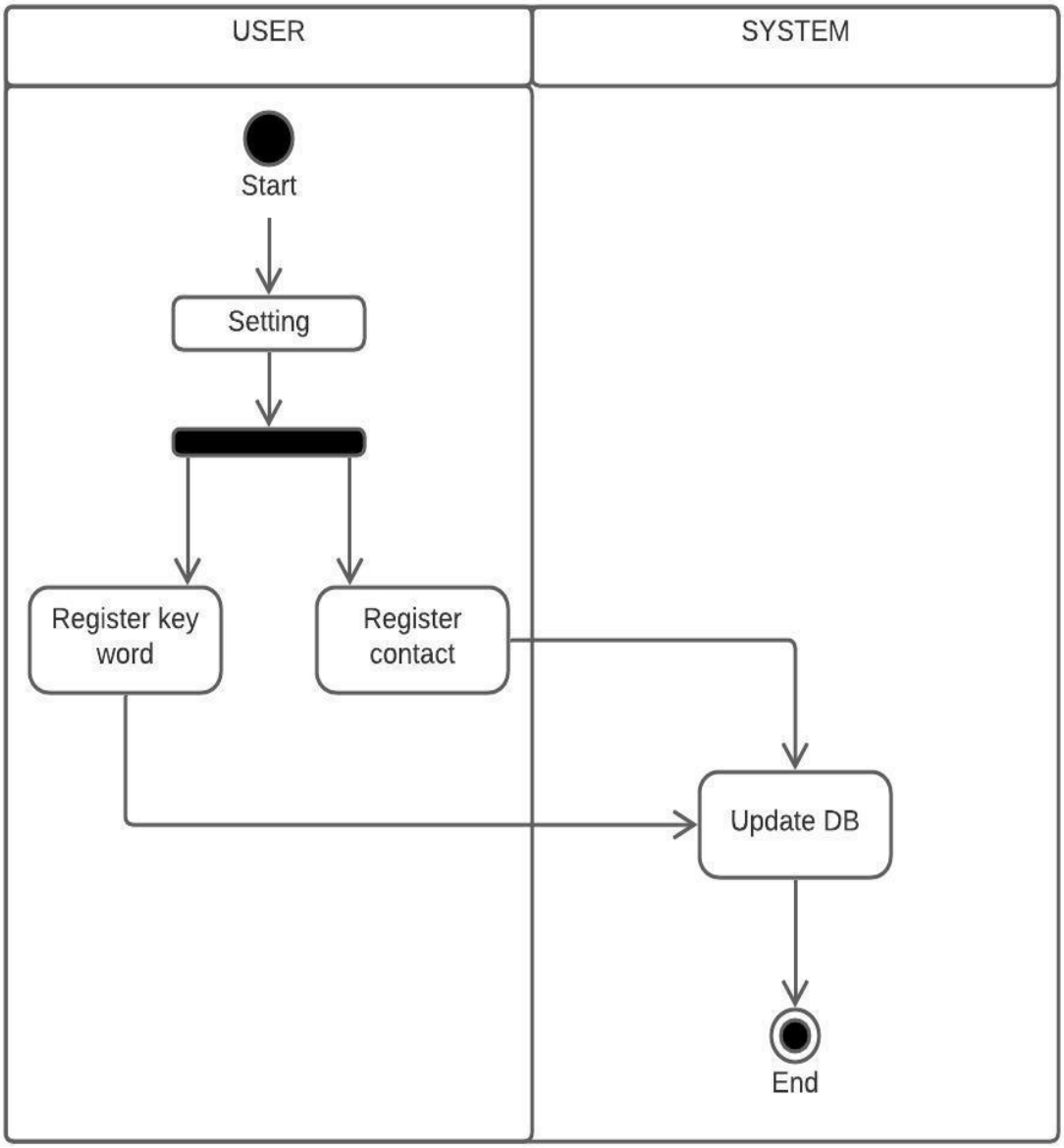


Figure 3.9: Settings activity diagram of Panic Management System

In settings activity diagram, when a new user is registered to the system, they need to set their special keywords and predefine contact(s) which the emergency message will be sent to. System will then update the database.

3.4.4 System Block Diagram

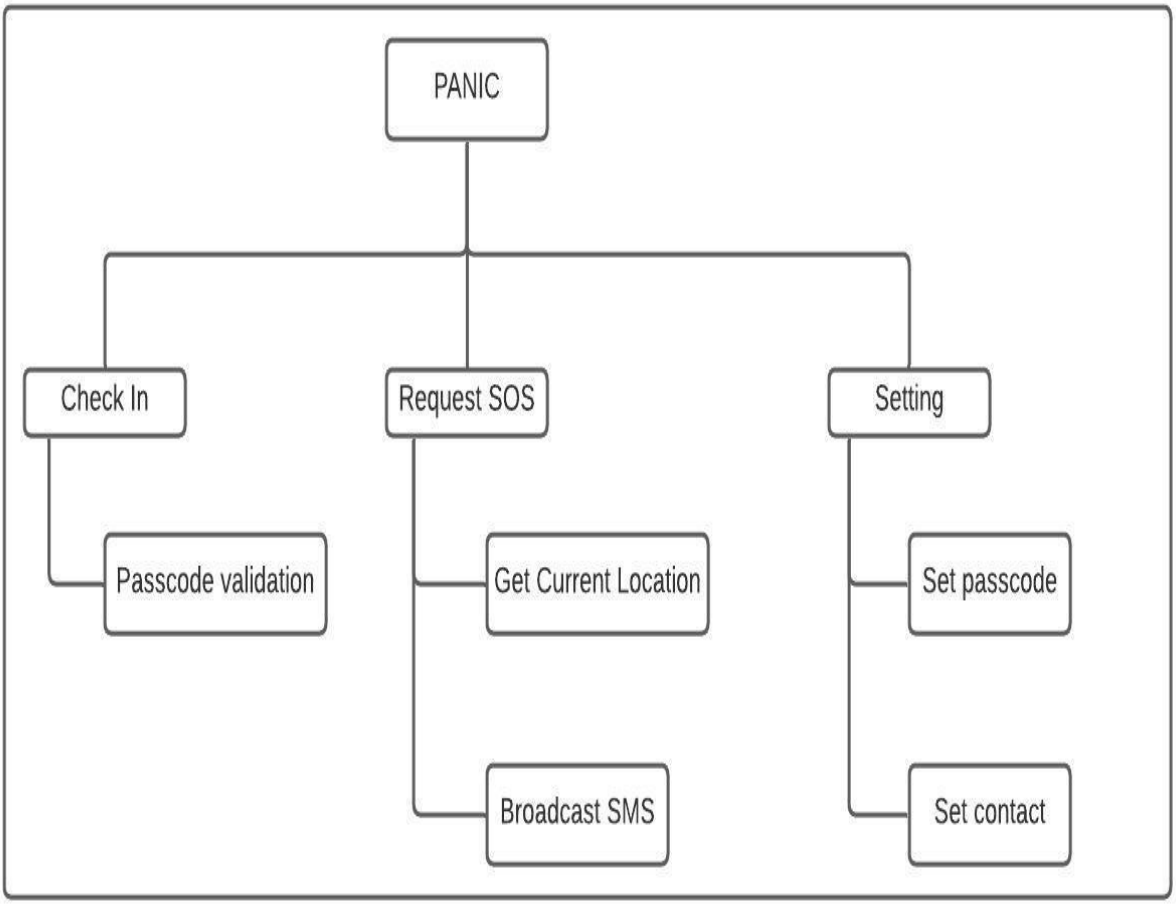


Figure 3.10: Panic management system block diagram

Based on the block diagram shown in Fig. 3.10, the entire system is defined in top down approach. Each module is clearly stated and will be developed through life cycle in incremental methodology. In this block diagram, one can have a fresh idea of what to start with and where to start from to

better design for coding. Interaction between some of the modules of which they have shared functions and class which can be reused in another module. Furthermore, efficiency and effectiveness of the program can be improved.

3.4.5 System Flow Chart Diagram

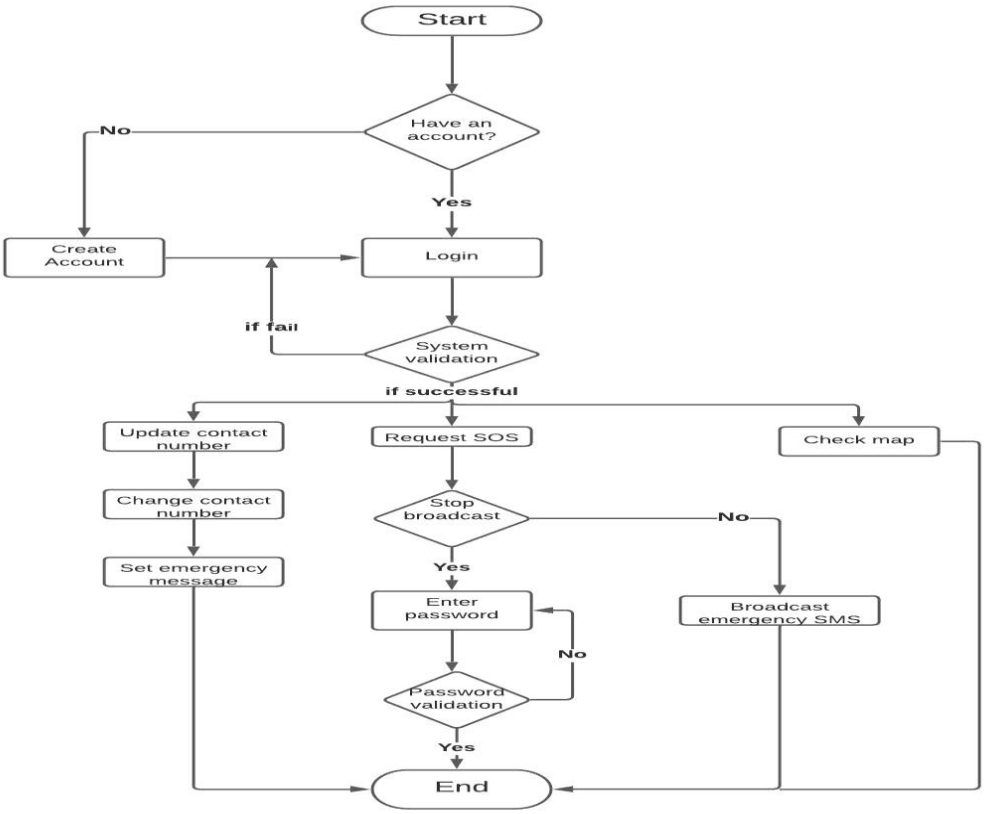


Figure 3.11: Panic management system flow chart diagram

In the first state, user need to create an account if they are new to it, otherwise user need to login with their valid number and password. After successful login, user can choose their action such as update their contact number, request SOS and check map. In request SOS, it will directly retrieve predefined emergency contact list and broadcast to them. In check map, the user can access the google map with their location pin pointed on the map.

CHAPTER 4

SYSTEM IMPLEMENTATION

This chapter explains the implementation of the Panic Management System in details. The Panic app is a collection of Activities and Fragments that are presented to the user. These Activities and Fragment have associate XML files (Layouts) declared in the layout folder, which determine the graphical interface for these components. The Panic app also contains other Service and Broadcast Receivers along with the declarations and necessary permissions in the Manifest file in the root directory of the project.

4.1 Software Used

4.1.1 Android Studio

Android studio is an official IDE for Android application development based on IntelliJ IDEA (Anon., n.d.). It is also an open source software for developers used as a platform for developing and debugging. It also features with intelliSense that can help developers in coding faster in development and correct their syntax error. Besides that, it provides drag and drop function to enable developers create layout for their GUI (Graphical User Interface) easily.

Although it provides convenient tools and environment for developers to develop their apps, but the problem encountered is that when develop make small change(s) to their code, it needs time to rebuild the apps and display the results to the developer. Sometimes, it spends few minutes to rebuild the application in order to see the result.

4.1.2 Android SDK

Android SDK is a comprehensive set of development that supports all versions of Android platforms (Anon., n.d.). It is used to develop Android application through Android Studio, Android

IDE, Eclipse IDE and others. Android SDK allows developers to develop applications in specified Virtual Android devices. This virtual emulator enables developers to view the result in a faster and convenient way. The problem found in Android SDK is that when a developer switches their work environment or multiple developers work on one project, they have to manually download required platform in order to run their apps successfully.

4.1.3 Android Emulator

This is a virtual android device that allow developers to run their apps on the android emulator. This emulator supports tablet, android wear, android phone, and android TV devices. Developers can launch this emulator to test their prototypes and develop applications without using hardware device. The problems encountered when using the emulator is that although it is convenient and best for developers that do not have a physical device, but creating a virtual environment in pc will slow down the performance of the apps. For example, the emulator requires large size of RAM and memory space to run on a pc, it will extremely slow down the overall pc performance that is if the RAM is not up to requirement. Other than that, this virtual environment may not perform as well as physical device.

4.1.4 Lucid Chart

It is a platform that provides UML design tools in design diagram which is available free for non-commercial use. It offers a complete tool-set software that need for requirements collection, software planning, test planning, class modeling, data modeling and others.

4.2 Graphical User Interface (GUI)

4.2.1 Home Screen

Every time the user opens the Panic app, this is the first screen presented to the user as shown in Fig. 4.1. The screen contains a switch at the top-left of the screen, which makes the panic button active if turned on and inactive if turned off as displayed in Fig. 4.2. A settings button at the top-right of the screen, it uses an intent function that redirects the user to the login screen in Fig. 4.3 of which a sign-in is required to access the Settings screen in Fig. 4.9. Then the Panic button at the center of the screen is the button that calls the speech recognizer and SMS manager function, which sends the emergency SMS. Then lastly is the Maps button, this button uses an intent function that redirects the user to a map fragment screen, which displays a google map with the location of the user being marked on the map.

Figure 4.1: Home screen of the Panic Management System.

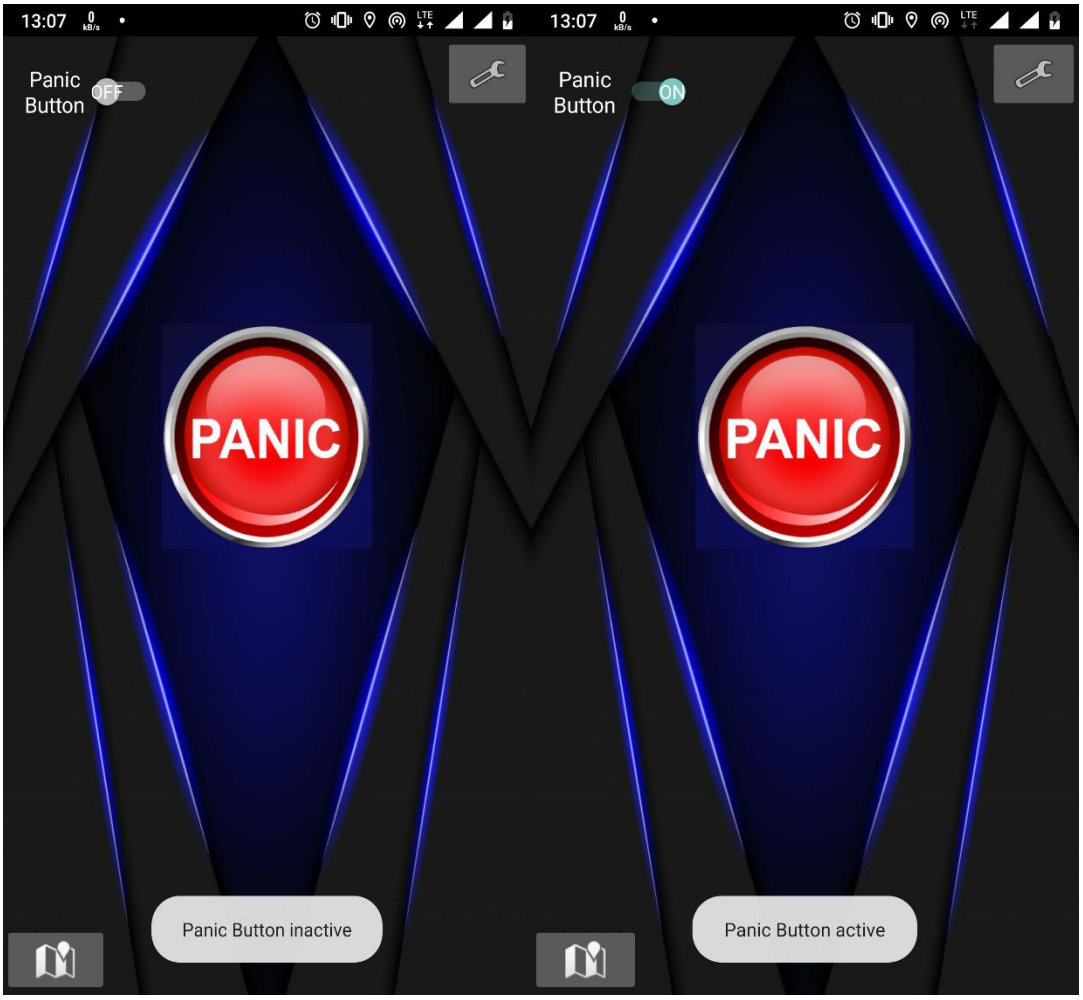


Figure 4.2: Switch activity and the respective feedbacks.

4.2.2 Login Screen

This screen is what comes on when the settings button on the home screen as displayed in Fig. 4.1 is clicked. This activity is responsible for accepting string inputs and authentication of the user's inputted data. This page consists of two editable text boxes for collecting inputs for the username and password of the user. The Login button as shown in Fig. 4.3, which uses an intent function to redirect the user to the settings screen as displayed in Fig. 4.9. The "or Register" text box which has a clickable function that uses intent function to redirect the user to the registration screen as

shown in Fig. 4.5. The “Forgot password” text box which has a clickable function that uses intent function to redirect the user to the forgot password screen as presented in Fig. 4.7.

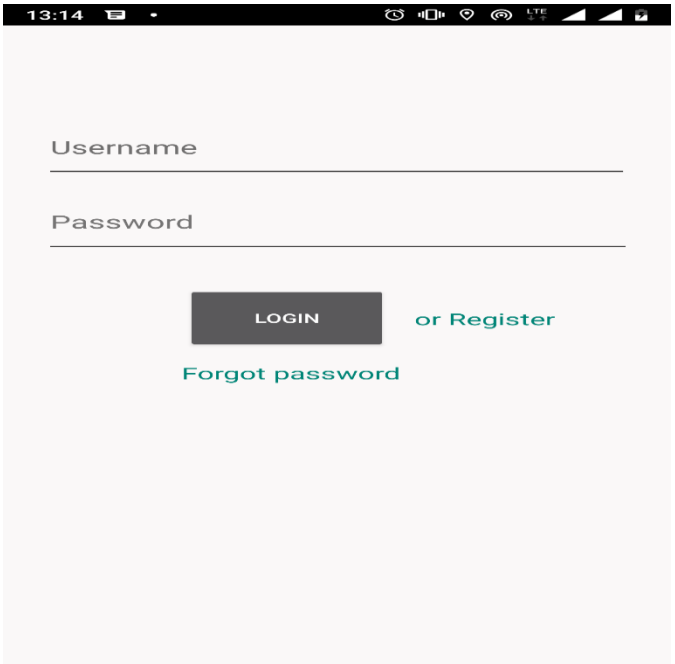


Figure 4.3:Login screen of the Panic Management System.

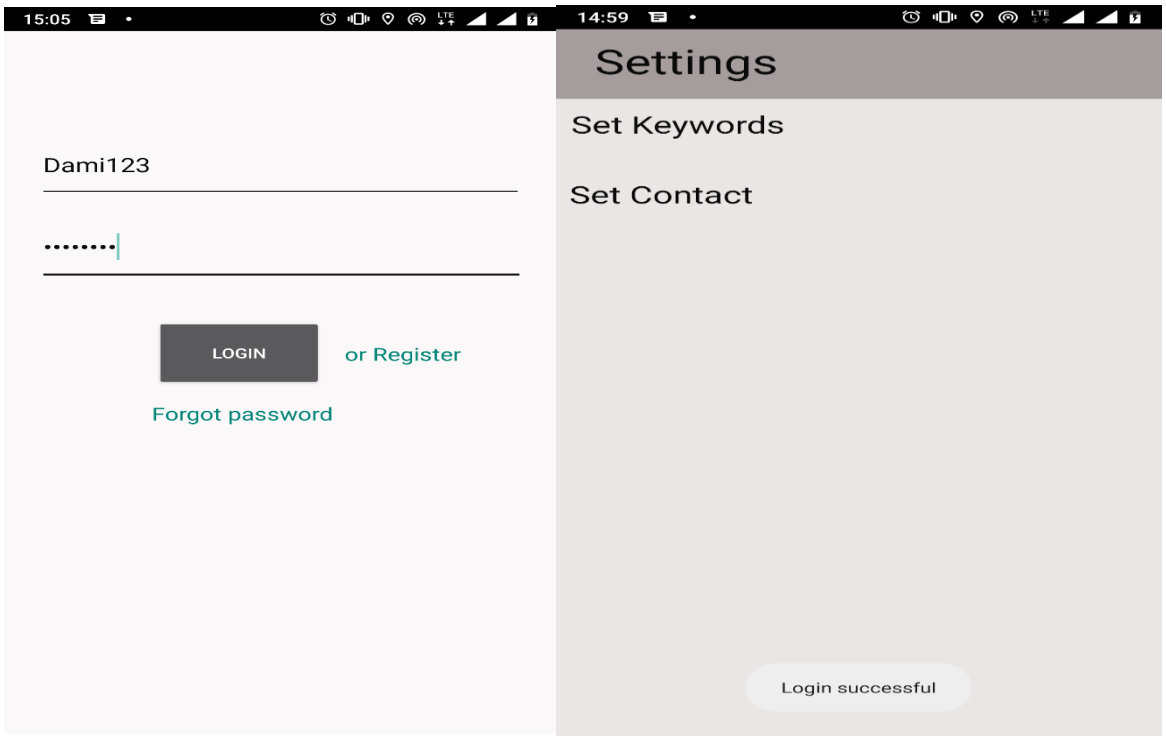


Figure 4.4: Login process and the feedback afterwards on the settings screen.

4.2.3 Registration Screen

The registration screen is called as a result of the intent function of the login screen activity. This screen consist of five editable text boxes for first name, last name, username, answer and password which all accept inputs that is store on the device using shared preference. There is a spinner which enables the user have a drop down list for private questions that serve as a form of authentication in the forgot password screen as shown in Fig. 4.5. The Register button is used to save all the inputs and redirect back to the login screen using the intent function.

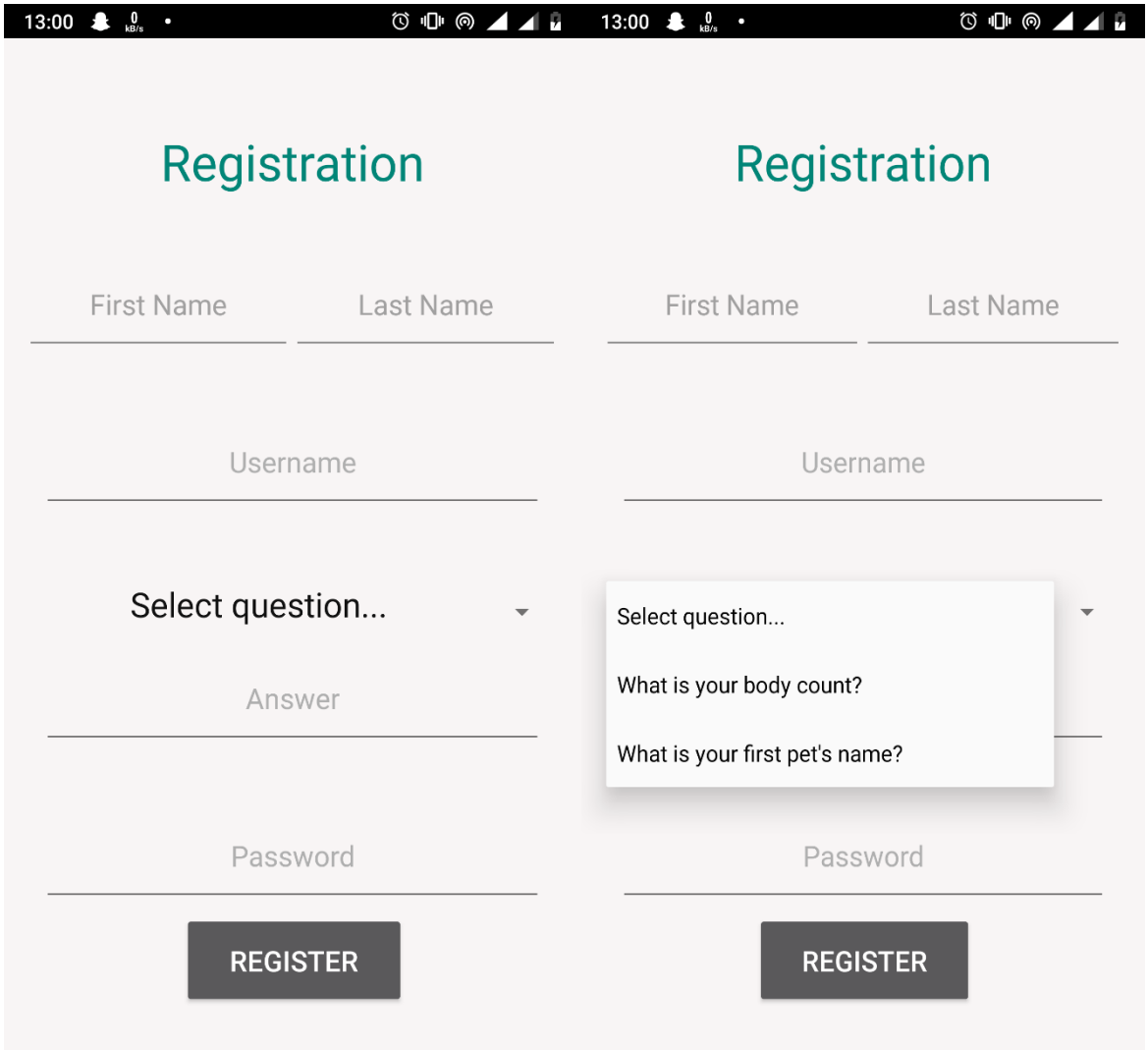


Figure 4.5: Registration screen of the Panic Management System.

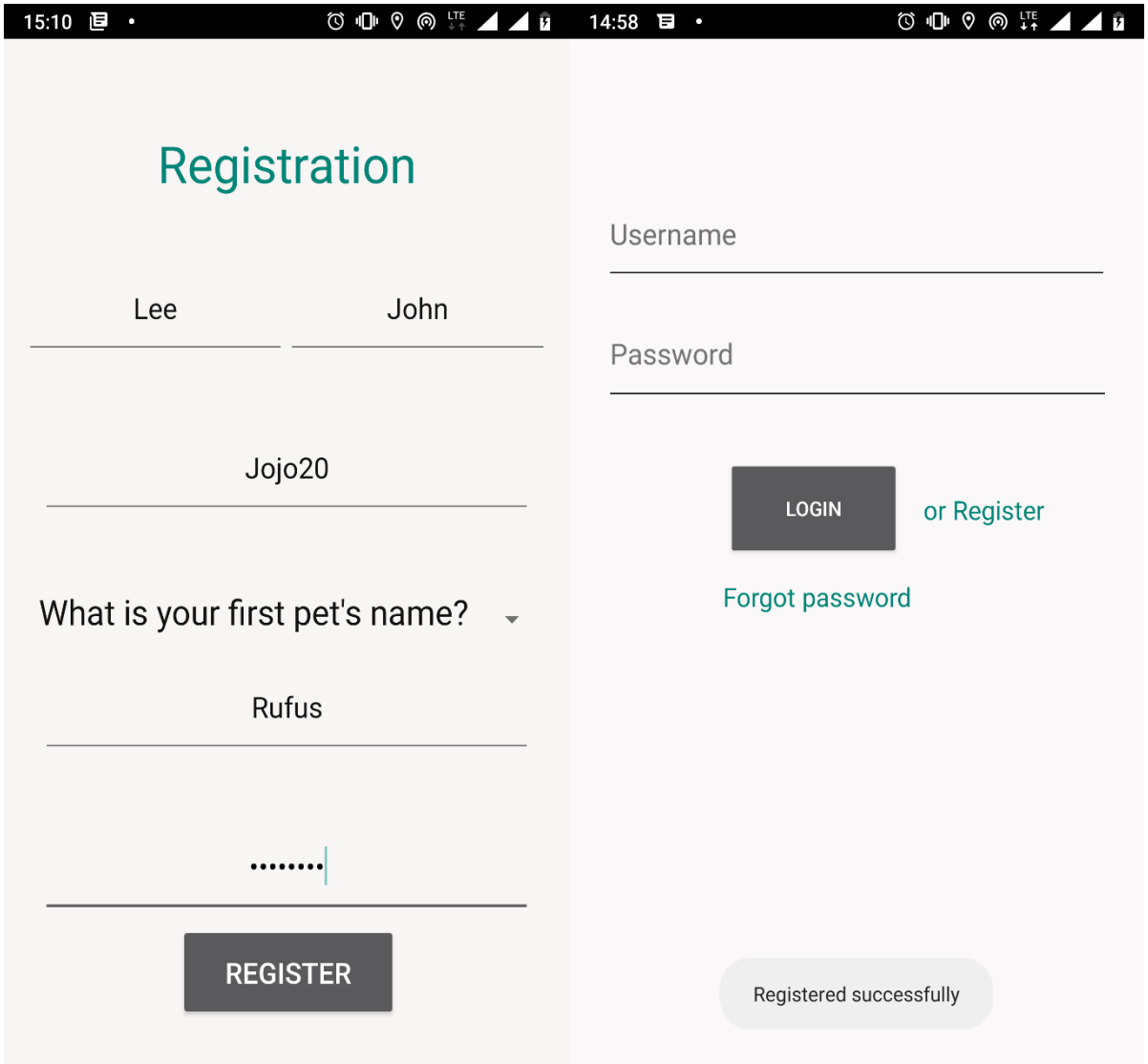


Figure 4.6: Registration process and feedback on the login screen after registering.

4.2.4 Forgot Password Screen

This is the page used to reset the password. It consists of five(5) editable text boxes to accept user inputs, a text boxes with an already fixed string containing the authentication question chosen during the registration phase and a reset button for saving the new password as shown in Fig. 4.8.

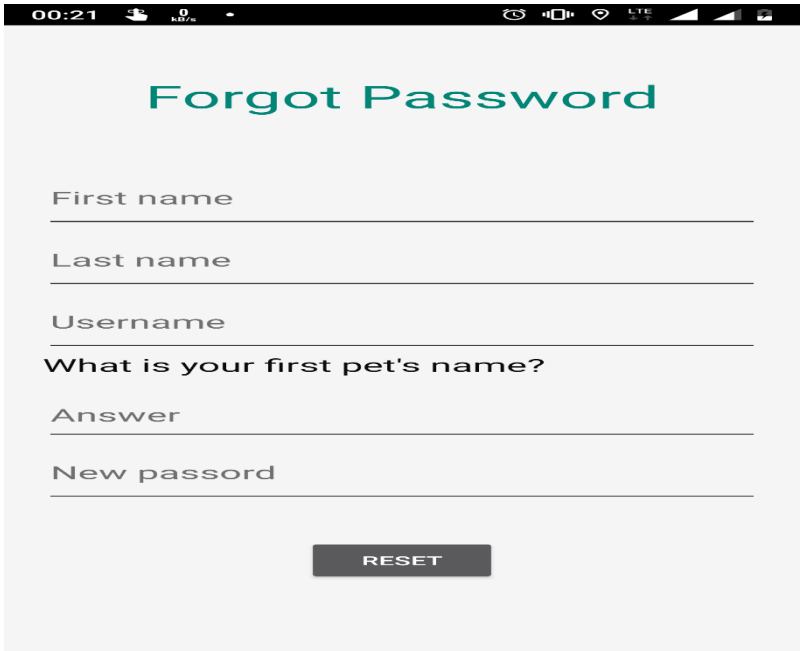


Figure 4.7: Forgot Password screen of the Panic Management System.

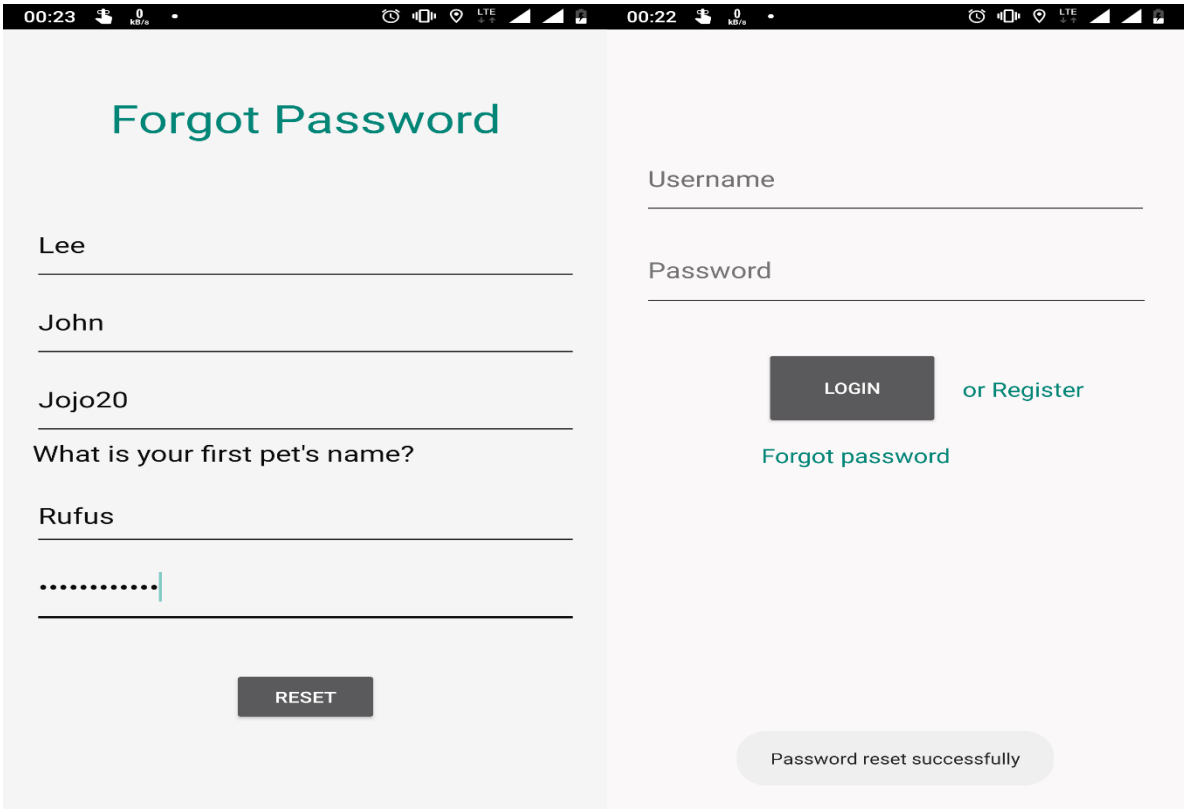


Figure 4.8: Resetting password process and the feedback on the login screen.

4.2.5 Settings Screen

This screen contains this set keywords and set contact intent class in it. It redirects to the set keywords and set contact screens respectively depending on the selected choice as shown in Fig. 4.9.

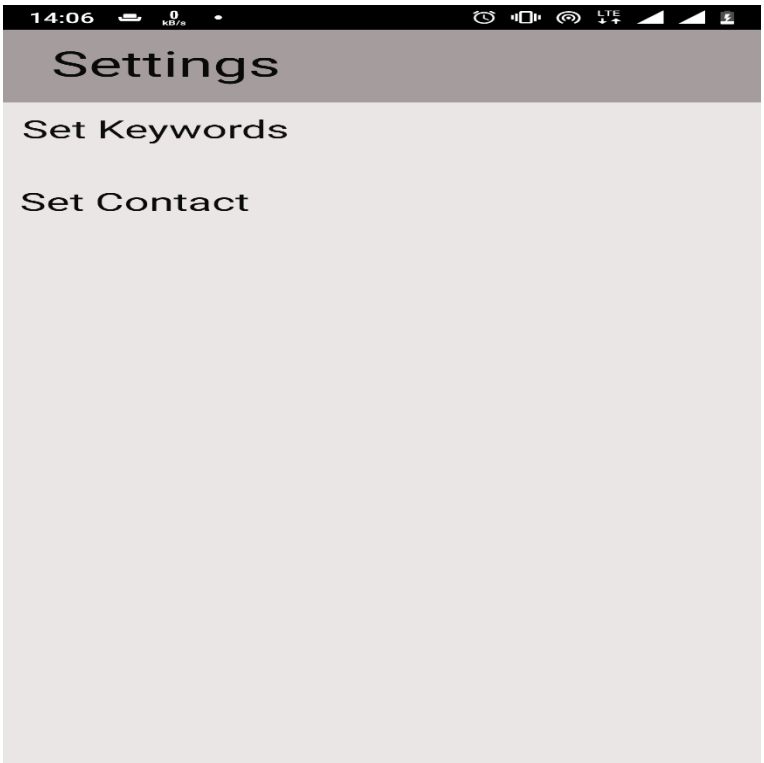


Figure 4.9: Settings screen of the Panic Management System.

4.2.6 Keywords Settings Screen

This scrollable interface enables the user to predefine keywords to be used for recognition of situation. It contains a total of nine(9) text boxes containing various common emergency situations and nine(9) editable text boxes for inputting of respective words or phrases. Then the Save button is what is used to save the string inputs into the repository using shared preference as displayed in Fig. 4.10.

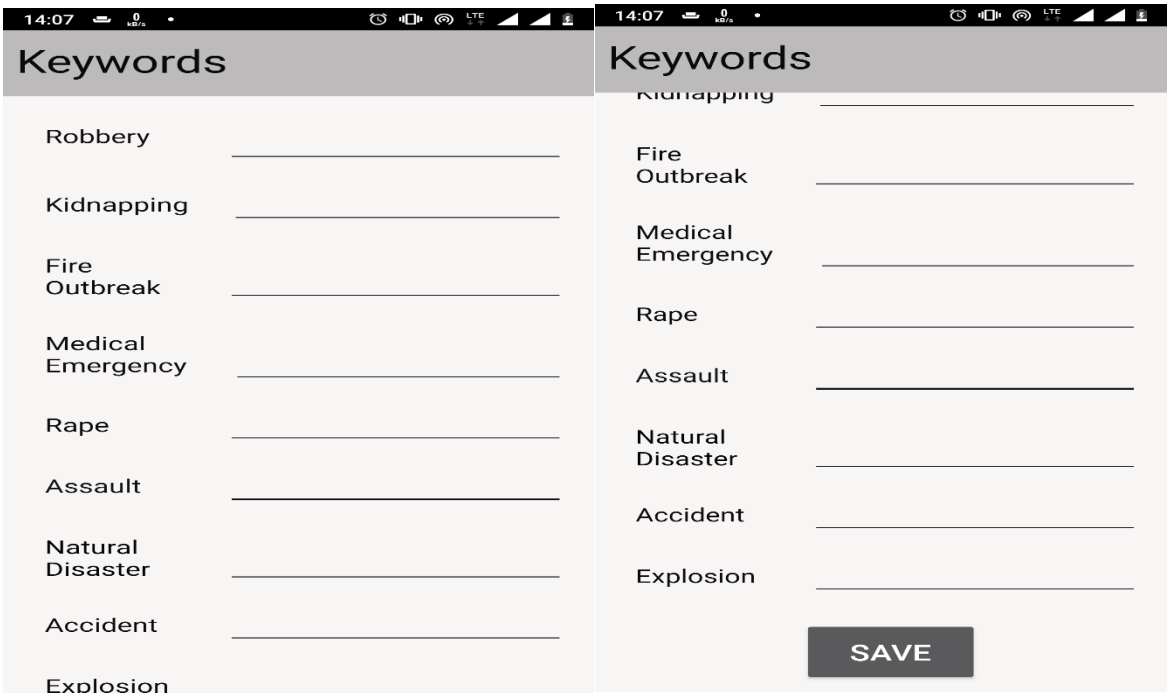


Figure 4.10: Keywords setting screen of the Panic Management System.

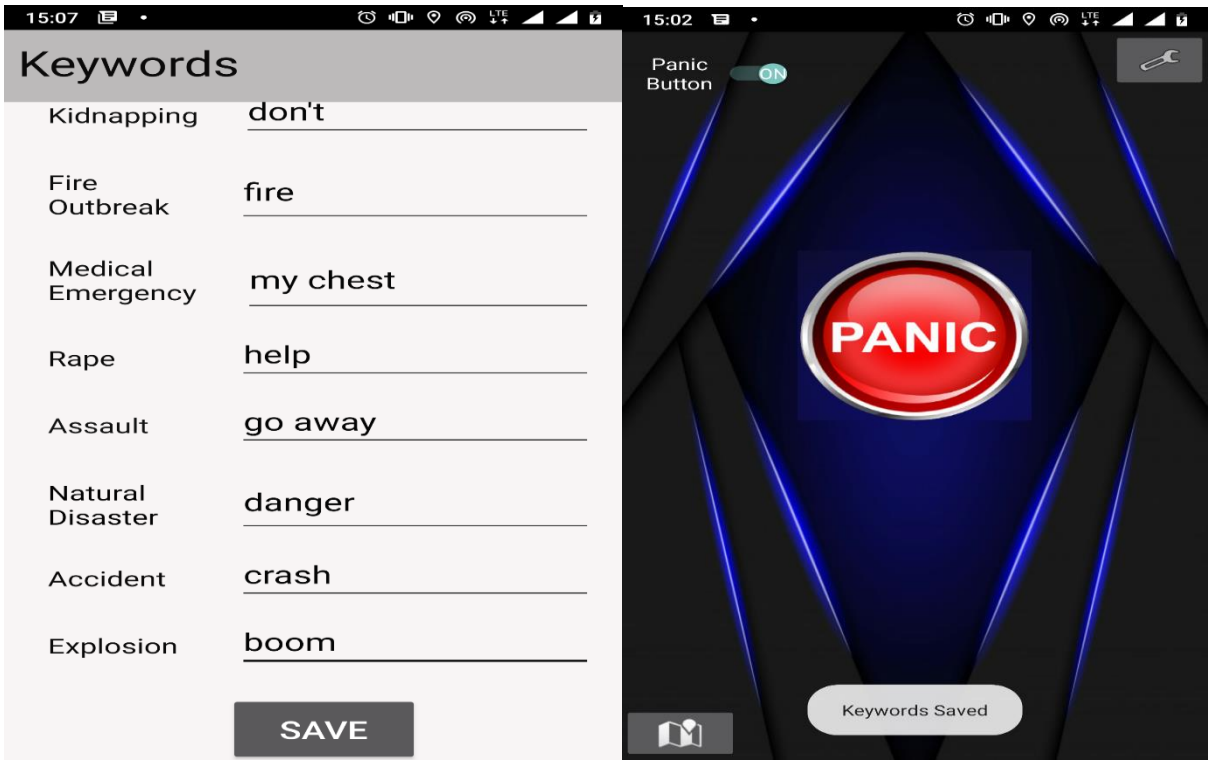


Figure 4.11: Keywords setting process and the feedback afterwards on the home screen.

4.2.7 Contact Setting Screen

The user can set a contact that will receive the emergency SMS using the Set Contact interface. The user can also specify the message to be sent to the predefined contact. This interface consists of three(3) editable text boxes to accept inputs for the name of the contact, the contact's number and the message to be sent to the contact as shown in Fig. 4.12. Then there is the Save button that enables the user to store the inputs in the repository.

Figure 4.12: Contact-setting screen of the Panic Management System.

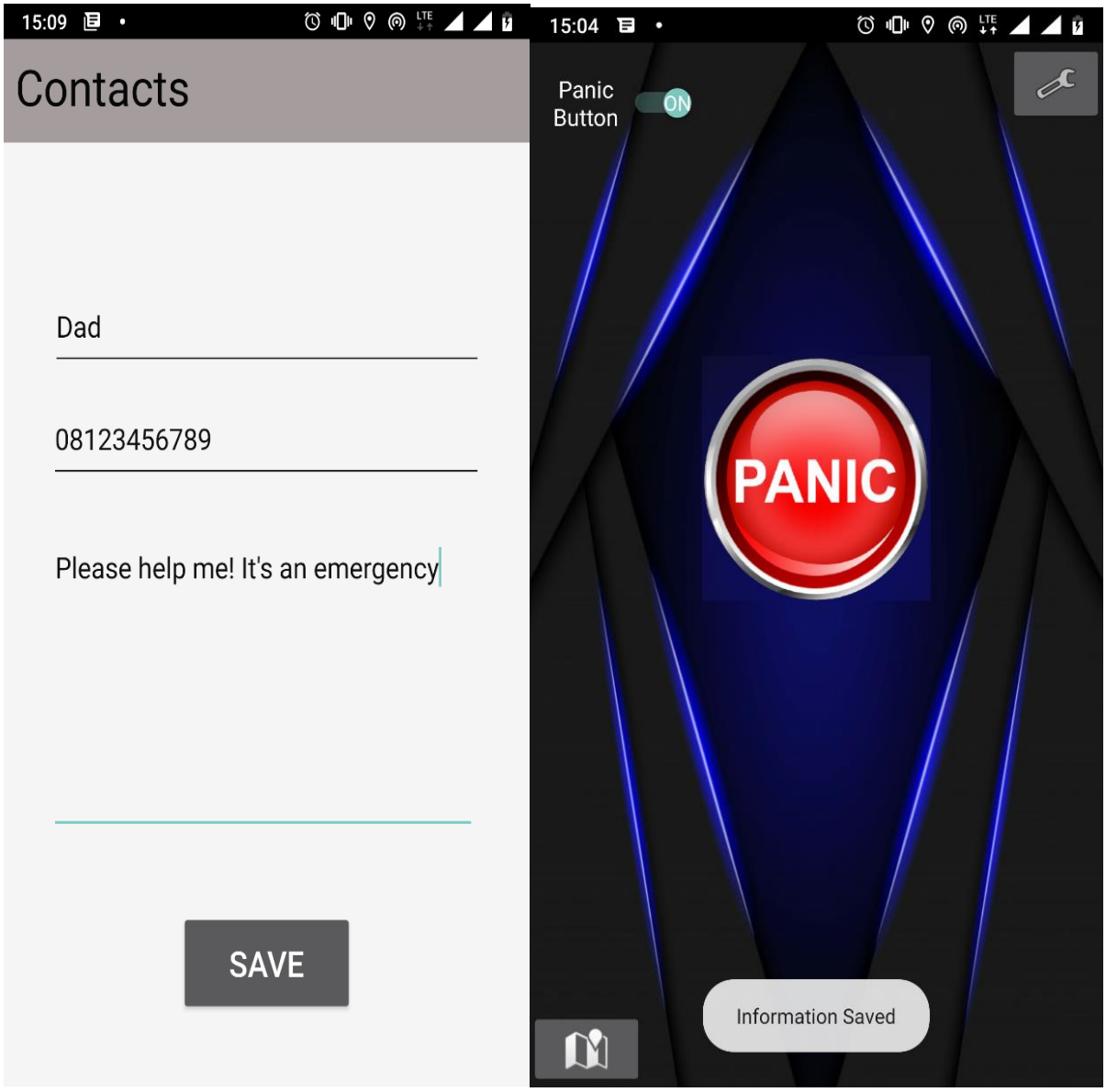


Figure 4.13: Contact and message setting alongside the feedback afterwards on the home screen.

4.2.8 Maps Fragment Screen

This maps fragment screen is the interface that displays the google map using Google maps engine as presented in Fig. 4.14. This interface also displays a marker which is used to identify the user's current location of which the location of the device has to be turned on to work as shown in Fig. 4.8.

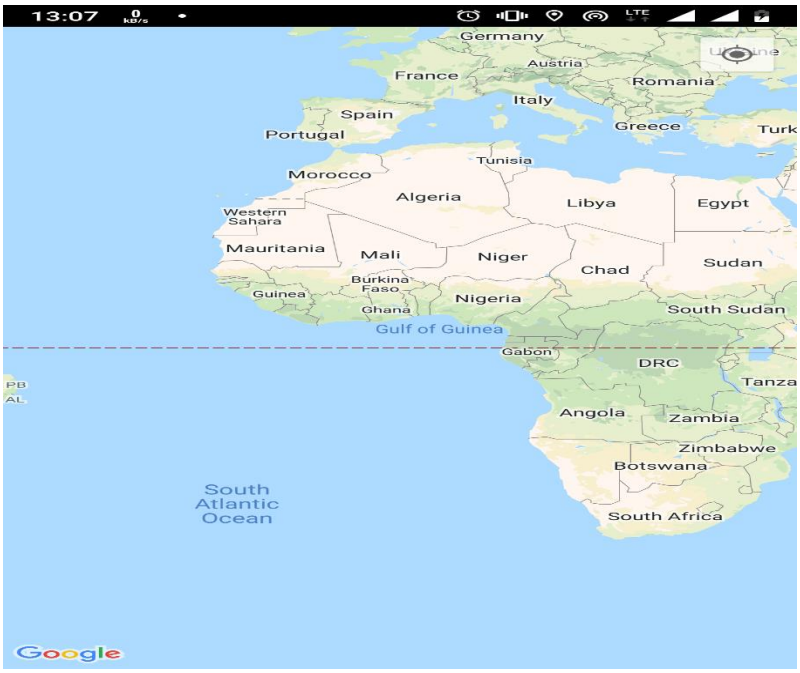


Figure 4.14: Maps fragment for the Panic Management System

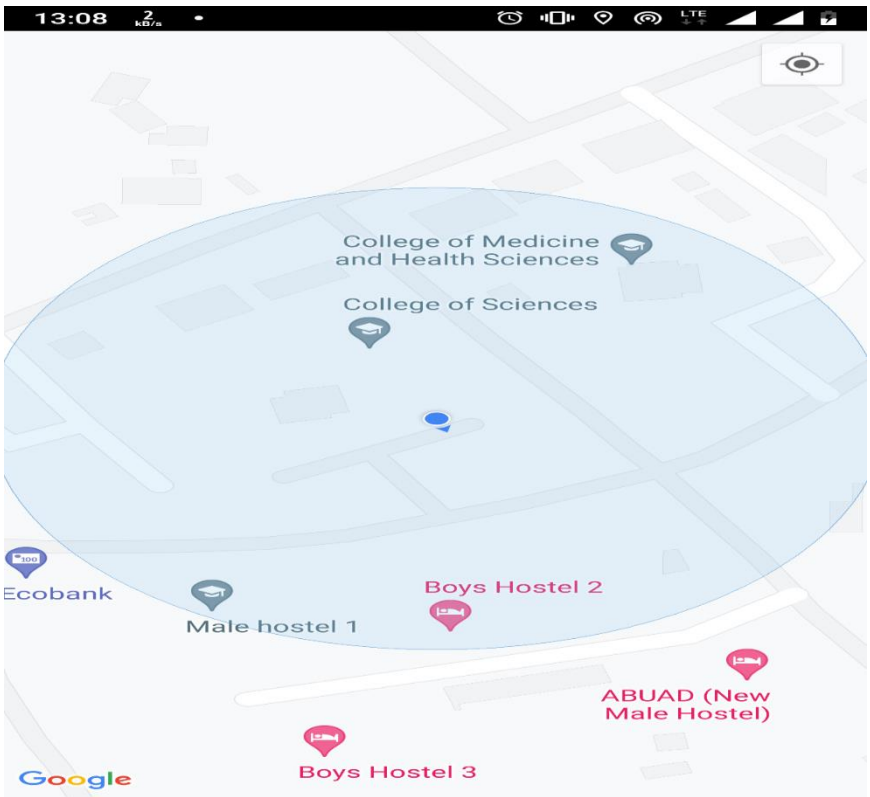


Figure 4.15: Marked location on the maps fragment.

4.3 Testing Methods

Testing is one of the important steps in development. It is used to make sure the application is well functioning and meets the requirement. It is also one of the phases in the life cycle of development method. Each of every module is developed and testing is conducted at this session. Functional Testing methodology is applied in this entire project. They are few testing components, which are indexed below:

i) Unit Testing

ii) Integration Testing

iii) Acceptance Testing

4.3.1 Unit Testing

Unit testing is an approach in software testing where individual components in a software are examined for correctness. In the PANIC apps, these components are the Activities that are presented to the user as screens on the Android device, Fragments, Services and Receivers. It takes on every module when each module function is done test to make sure they are well functioning. Each module in this will be taken for testing at this stage.

4.3.2 Integration Testing

Integration testing is testing in different module that had been successful in unit testing and combine together to perform specific function. With combined modules above, integration test is to apply in this stage. Combined modules will be test consequently until there is no error to perform specific function. It is carried out when the components are done with unit testing and then the main objective is to test the interfaces between different components.

4.3.3 Acceptance Testing

In accepted testing, it tests with respect to user requirements, and business processes conducted to determine whether a system satisfies the acceptance criteria or not and to enable the user, customers or other authorized entity to determine whether to accept the system or not. There are few test cases to be tested.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

Chapter 1 provides a brief idea of background, problem statement, project scope, objective, impact and contribution, and lastly definition of related terms. In in background, it provides general introduction for reader to have some idea on it. In problem statement, it explained the existing problem people face and its reason. In project scope, it outlines the things and area that will be covered in this proposed application. While there are several points listed in the objective section. It helps proposer to have clear objective in order to complete the project. Lastly, impact and contribution mentions how this proposed application would impact to the society.

Chapter 2 covers the literature review of existing system. There are 15 existing applications chosen for review. It helps to screen out the weaknesses and strengths of existing applications. It also include some existing product screenshot that shows how they look like.

Chapter 3 describes the methodology use in this application. It shows different stage of development phases and explain in details of each process. Fig. 3.1 shows the phases and how it moves at each cycle. Other than that, analysis of the requirements and requirement specification for the system was specified. Lastly, the system design were identified to show the system architecture, system use case diagrams, system activity diagrams, system block diagram and system flow chart diagram. It provides and explains the proposed application flow. Readers can understand easily through the system design diagrams instead on reading everything in document.

Chapter 4 covers the implementation and testing. It involves the tools and software used to develop this application. It lists out the details of software and hardware needed. In Graphical User Interface, it shows the proposed application layout and the navigation between the pages. Lastly is testing method. All the test method and test cases are taken to be tested. It used to test the actual and expected outcome of the system.

5.2 Conclusion

In conclusion, the crime index in Nigeria is increasing; the rate of public fear of crime has an equivalent of it. Nowadays, many people like to hang out at night, as the result of increase in public fear; they tend to worry about their personal safety including their friends and family. A personal security app lets you send notifications to certain people via text messages in case of emergencies. The app also keeps a track of your current location so that you always known the address of where you are. The text messages also has this location information. The app can also track your location periodically and store it permanently enabling you to see your location history. PANIC app was my first attempt at an Android application. It gave me very good exposure to the Android platform and mobile development in general. The app enabled me in understanding the basic of Android development and learning about Shared Preference, Google Maps API for Android and performance testing the app.

5.3 Recommendations

The current work on the PANIC app has a lot of essential features that would be used in case of an emergency situation like sending text messages from within the app on tap of a single button. An app for such a purpose has a lot of scope for enhancement. In the future the app may include features like:

i. A home screen widget that can be used as a triggering point to send panic notifications. A user would then not have to open the app to send these panic notifications.

ii. Initiating a call to a number set from within the application when the user presses the panic button.

iii. The app can also listen to incoming messages from the set contacts. If these message have a pre-defined text like “Update Location” the app can reply with a text message containing the current location or for some other text like “Audio” in which case the app can record a short audio and send it as an email to the person. This is very helpful as you may have already pressed the panic button and may be in some trouble where you cannot reply. This way the person can track you constantly and also understand something about the nature of the emergency from the audio clip.

iv. The app should also be able to listen while the app is minimized just like the “Shazam” app.

REFERENCES

Amit Kushwaha. (2011). “Location Based Services using Android Mobile Operating System”, International Journal of Advances in Engineering and Technology.

Andreas Geyer-Schulz, Michael Ovelgonne, Andreas C. Sonnenbichler. (n.d). “Getting Help In A Crowd A Social Emergency Alert Service," IEEE Transaction.

Android Apps for Sms: http://www.fileguru.com/Android-Apps-For-SMS/info.

AndroidApps.Biz. (2016). saveme 999 1.2.6 APK for Android. [Online] Available at: http://www.androidapps.biz/app/com.tm.saveme999.android [Accessed 24 April, 2021].

Andtek. (2012). Just Expenses: Expense Tracker. Retrieved October 20, 2012, from Google Play.

Anind D, Jeffrey H, Eyal dL, Nigel D. (2010). “Location-based services,”IEEE Pervasive Computing, pp. 11-12.

Anon. (2016). Lee: Crime index down, but not public fear. [Online] Available at: http://www.thestar.com.my/news/nation/2016/04/28/lee-crime-index-down-butnot-public-fear/ [Reviewed April 2021].

APKMonk. (n.d). SOS Siren/Whistle. [Online] Available at: http://www.apkmonk.com/app/com.aiss.siren/ [Accessed 20 April, 2021].

Bhaskar Kamal Baishya. (2014). “Mobile Phone Embedded With Mobile and Security and Applications,” Journal of Computer Engineering (IOSR-JCE).

Bin Yu, Lisheng Xu and Yongxu Li. (2012). “Bluetooth Low Energy (BLE) Based Mobile Electrocardiogram Monitoring System”, Proceeding of the IEEE, International Conference on Information and Automation, Shenyang, China.

Building apps with Graphics and animation. (n.d). Retrieved from: http://developer.android.com/training /building-graphics.html.

C. K. Harnett, Member, IEEE. (2010). “Open Wireless Sensor Network Telemetry Platform for Mobile Phones” IEEE SENSORS JOURNAL, VOL. 10, NO. 6.

Corp., G. (2005). Android. Reviewed May 2021.

Cutter, S.L. (2003). GI Science, Disasters, and Emergency Management. Transactions in GIS 7: 439-445. DOI: 10.1111/1467-9671.00157.

Developers, Android. (2014). Activity. Online: http://developer.Android.com/training/basics/Activity-lifecycle/starting.html. [Reviewed June 2021].

Developers, Android. (2014). Intents. Online: http://developer.Android.com/guide/components/Intents-filters.html. [Reviewed June 2021].

Developers,Android. (2014). Traceview tool. Online: http://developer.Android.com/tools/debugging/debugging-tracing.html. [Reviewed June 2021].

Developers,Android Services. (2014). Online: http://developer.Android.com/guide/components/services.html. [Reviewed June 2021].

Developers,Android Broadcast Receivers. (2014). Online: http://developer.Android.com/reference/Android/content/BroadcastReceiver.html. [Reviewed June 2021].

Developers,Android Android basics. (2014). Online: http://developer.Android.com/training/index.html. [Reviewed June 2021].

Developers, Google. (2014). Google Maps Android API V2. Online: https://developers.google.com/maps/documentation/Android/. [Reviewed June 2021].

Dhrubajyoti Gogoi, Rupam Kumar Sharma. (2013). “Android Based Emergency Alert Button," IJITEE Volume-2, Issue-4.

Durairaj Ponraj, LuoZhiqiang, AngCandy. (2012). “Developing a Location-based Mobile App to Save Heart Patients in Emergency,” The Asian Conference on Education.

Earnest Paul, Swathi Pusuluri. (2012). “Hospital Information Service On Android Mobile Devices,"IJERT Vol. 1 Issue 3.

Favell, A. (2012). Global mobile statistics 2012. Reviewed May 2012, from Mobi Thinking.

Grell, G. (2010). “Introduction to Android Development”.

Hugo Paredes, Bejamim Fonseca, Mariam Cabo, Tania Pereira. (2013). “SOSPhone: A mobile application for emergency calls”, Universal Access in the Information Society 13(3):1-14 DOI:10.1007/s10209-013-0318-z.

Hughes Systique. (n.d.). “Go Suraksheit”. Retrived from: https://appadvice.com/app/go-suraksheit/724254561.

IDC. (2014). Online: http://www.idc.com/getdoc.jsp?containerId=prUS24676414.

James Steele, N. T. (2010). The Android Developer’s Cookbook. Rights and Contracts Department 501 Boylston Street, Suite 900 Boston, MA 02116: RR Donnelley, Crawfordsville, Indiana. Pearson Education, In.

Jenny Wisniewski. (2021). Medical Alert Systems Expert. Retrieved from: https://www.safehome.org/medical-alert-systems/greatcall/reviews/.

Kim, A. (2012). Coin Keeper. Retrieved October 25, 2012, from http://coinkeeper.tumblr.com/.

Kim Stephens. (2013). “Smartphone Apps, the Next Step for Social Media and Emergency Management?”. Retrieved from: https://idisaster.wordpress.com/2013/01/.

Lee, L.K. Zachariah, M. Everett, P. (1995). “CCTV camera site selection: a field experience”, 6th International VNIS. “A Ride into the Future“, pp.21-27.

L.Hariprasath, R.Dhivya, S.Adithya. (2013). “Emergency Alert System Using Android”, International Journal of Research in Engineering & Advance Technology, Volume 1,Issue 1.

Mansour Ali Almadani. (2018). “AI Speech Recognition System”, Computer Science And Information Technology affiliated to Dr. Babasaheb Ambedkar Marathwada University, Aurangabad.

Manuel Fogue, Piedad Garrido, Francisco J. Martinez, Juan-Carlos Cano, Carlos T. Calafate, and Pietro Manzoni. (2012). “Automatic accident detection”, IEEE vehicular technology magazine.

Mattias Spangmyr. (2014). Development of an Open-Source Mobile Application for Emergency Data Collection in Department of Physical Geography and Ecosystem Science, Lund University.

M.F.L. Abdullah and L. M. Poh. (2011). “Mobile Robot Temperature Sensing Application via Bluetooth”, International Journal of Smart Home, vol. 5(3), pp. 39-48.

Michael Ovelgonne, Andreas C. Sonnenbichler, Andreas Geyer-Schulz, “Social Emergency Alert Service - A Location-Based Privacy-Aware Personal Safety Service,” IEEE 2010 Fourth International Conference on Next Generation Mobile Applications, Services and Technologies.

M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda. (2009). “Peir, the Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research,” Proc. 7th ACM MobiSys, New York, USA, pp.5568.

NA. (2010). Disadvantages of File Processing System. Reviewed May 2021, from JKiNFOLINE.com.

NA. (2012). Expense Manager. Reviewed May 2021, from Google Play.

NA. (n.d.). Android, the world's most popular mobile platform. Reviewed May 2012, from developer.android.com.

NA. (n.d.). Daily Exense Tracker. Reviewed May, 2012, from freelancer.com.

Ng Ken Hong, Hanayanti Hafit, Norfaradilla Wahid, Shahreen Kasim. (2017). "Help Me Please!: Designing and Developing Application for Emergencies", IOP Conference Series Materials Science and Engineering 226(1):012116 DOI:10.1088/1757-899X/226/1/012116.

Nikki. (2016). "RapidSOS Haven Review". [Online] Available at: http://www.theiphonemom.com/haven-review/ [Accessed 20 April, 2021].

Nokia. (2005). "Workshop on Large-Scale Sensor Networks and Applications," Kuusamo, Finland.

N. Eagle and A. Pentland. (n.d). "Reality Mining: Sensing Complex Social Systems," Personal Ubiquitous Comp., vol. 10, no. 4, 2006, pp. 255268.

N.D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A.T.Campbell. (2010). "A Survey of Mobile Phone Sensing," in IEEE Commun. Mag.,vol. 48(9), pp. 140-150.

OC Index. (2019). Organized Crime Index. Retrieved from: https://ocindex.net/assets/downloads/ocindex_profile_nigeria.pdf. [Reviewed June 2021].

Official Bluetooth Website. (2013). [Online]. Available: http://www.bluetooth.com

Official ELERTS Website. (2013). [Online]. Available: http://elerts.com

Official GreatCall Website. (2013). [Online]. Available: http://www.greatcall.com

Official Linear corp"s Website. (2013). [Online]. Available: http://www.linearcorp.com/product_detail.php?productId=1538

OGC. (2013). OpenGeoSpatial.org website. Retrieved 2013-07-25, from http://www.opengeospatial.org/.

Osnat Mokryn, Dror Karmi, Akiva Elkayam, Tomer Teller. (2012). “Help Me: Opportunistic Smart Rescue Application and System” The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net).

Pratyush Poddar, Karanveer Singh Dehal, Praveen Kumar. (2013). “Sos Android Application," IJERT Vol. 2 Issue 3.

P. Belchior, M. R. Tomita, B. J. Kemp. (2005). “Use of personal emergency response systems by older individuals with disabilities," Assistive technology, vol. 17, no. 1, pp. 8288.

P. Kalyanchakravarthy, T. Lakshmi, R. Rupavathi, S. Krishnadilip, P. Lakshmankumar. (2014). “Android Based Safety Triggering Application," International Journal of Computer Science and Information Technologies, Vol. 5 (1), 646-647.

R. Grzeszczyk, J. Merkisz, P. Bogus, and T. Kaminski. (2009). “Methods and Procedures for testing the E-call in-vehicle unit for the purpose of its performance assessment and certification, in Proc. 21st Int. Technical Conf. Enhanced Safety of Vehicles (ESV), Stuttgart, Germany.

Richards, H. (2013). “bSafe Could Save Your Life”. [Online] Available at: https://iphone.appstorm.net/reviews/lifestyle/bsafe-could-save-your-life/ [Accessed 23 April, 2021].

Satapathy, A. (2012). Top 5 SOS apps for women. [Online] Available at: http://www.rediff.com/getahead/slide-show/slide-show-1-gadgets-and-gaming-top5-sos-apps-for-women/20121221.htm#4 [Accessed April 2021].

Satarkar, P. (2004). J2EE Design Patterns. Reviewed April 2021, from All App Labs.com.

Shahrudin, H. S. (2016). Malaysia's crime index sees 4.6 per cent increase. [Online] Available at: http://www.nst.com.my/news/2016/05/144196/malaysias-crime-index-sees-46- cent-increase [Accessed March 2021].

Shri Krishna, G., Lokesh, M.P. (2014). “Advanced SOS App in Smartphone," JBAER Volume 1.

Suryawanshi Akash. (2014). “SOS – An Android Application for Emergencies”, Department of Computing and Information Sciences College of Engineering, B.E., Shri Govindram Seksaria Institute of Tech & Science, India.

Sutcliffe, Geoff. (2014). Activity Life Cycle. Online: http://www.cs.miami.edu/~geoff/Courses/CSC300-13S/Content/ActivityLifeCycle.html.

S. Aram, A. Troiano, and E. Pasero. (2010). “Environment Sensing using Smartphone”, ©2012 IEEE 13. H. Sugiharto , “Current and Future Mobile Platforms” SNET Master Project. Available:

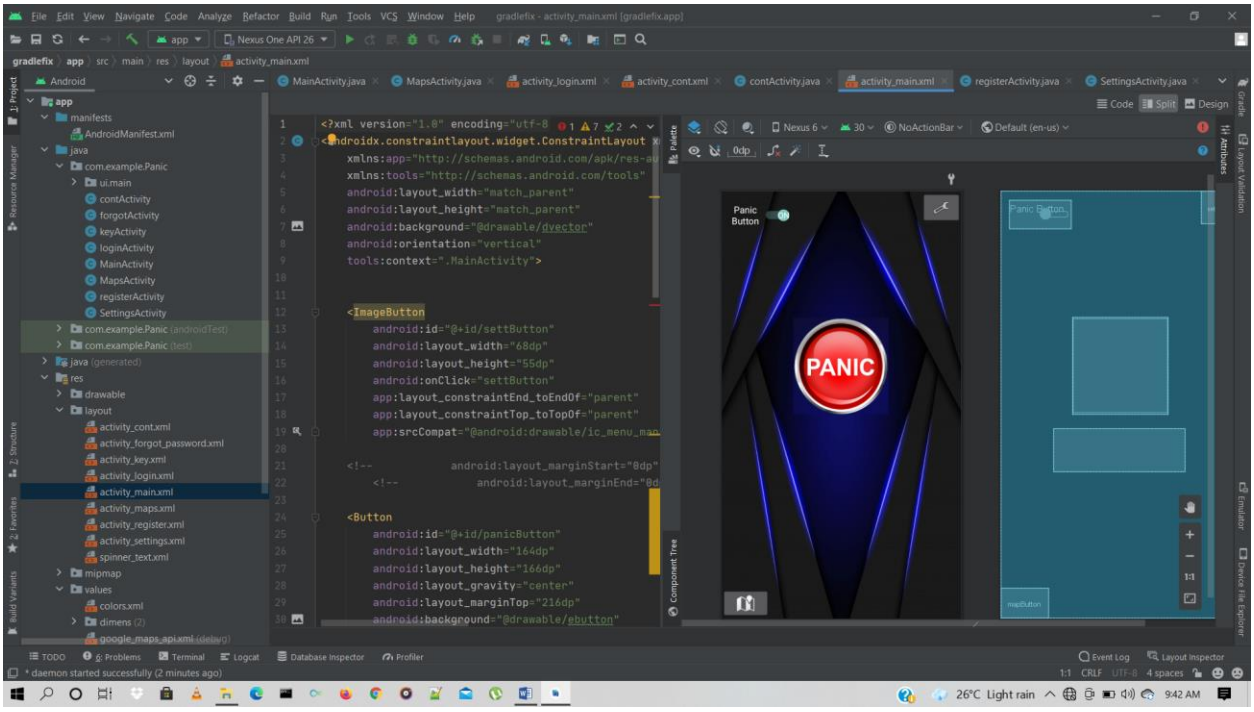
http://www.snet.tuberlin.de/fileadmin/fg220/courses/SS10/snetproject/mobileplatforms_sugiharto.pdf

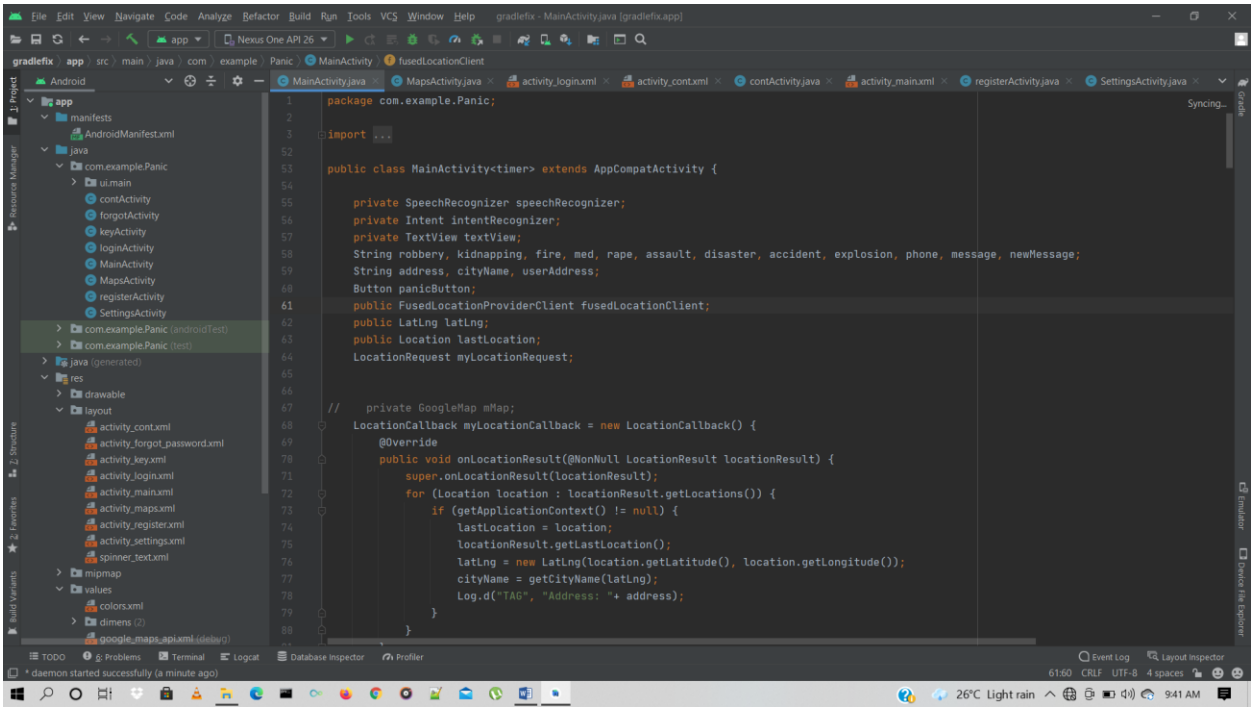
Thiek, J. (2014). “10 Personal Safety Apps To Keep You Protected”. [Online] Available at: http://www.snapptips.com/10-personal-safety-apps/ [Accessed 20 April, 2021].

Vogel, Lars. (2014). Android System Architecture. Online: http://www.vogella.com/tutorials/Android/article.html.

Wekeat Heng. (2018). Mobile Application for Emergency Services By Using Location Tracking in Faculty of Information and Communication Technology, Universiti Tunku Abdul Rahman.

APPENDIX


gradlefix - activity_main.xml [gradlefix.app]
<?xml version="1.0" encoding="utf-8
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:app="http://schemas.android.com/apk/res-au
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@drawable/dvector"
android:orientation="vertical"
tools:context=".MainActivity">
<ImageButton
android:id="@+id/settButton"
android:layout_width="68dp"
android:layout_height="55dp"
android:onClick="settButton"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:srcCompat="@android:drawable/ic_menu_man
<!-- android:layout_marginStart="0dp"
<!-- android:layout_marginEnd="0d
<Button
android:id="@+id/panicButton"
android:layout_width="164dp"
android:layout_height="166dp"
android:layout_gravity="center"
android:layout_marginTop="216dp"
android:background="@drawable/ebutton"
Panic Button
PANIC

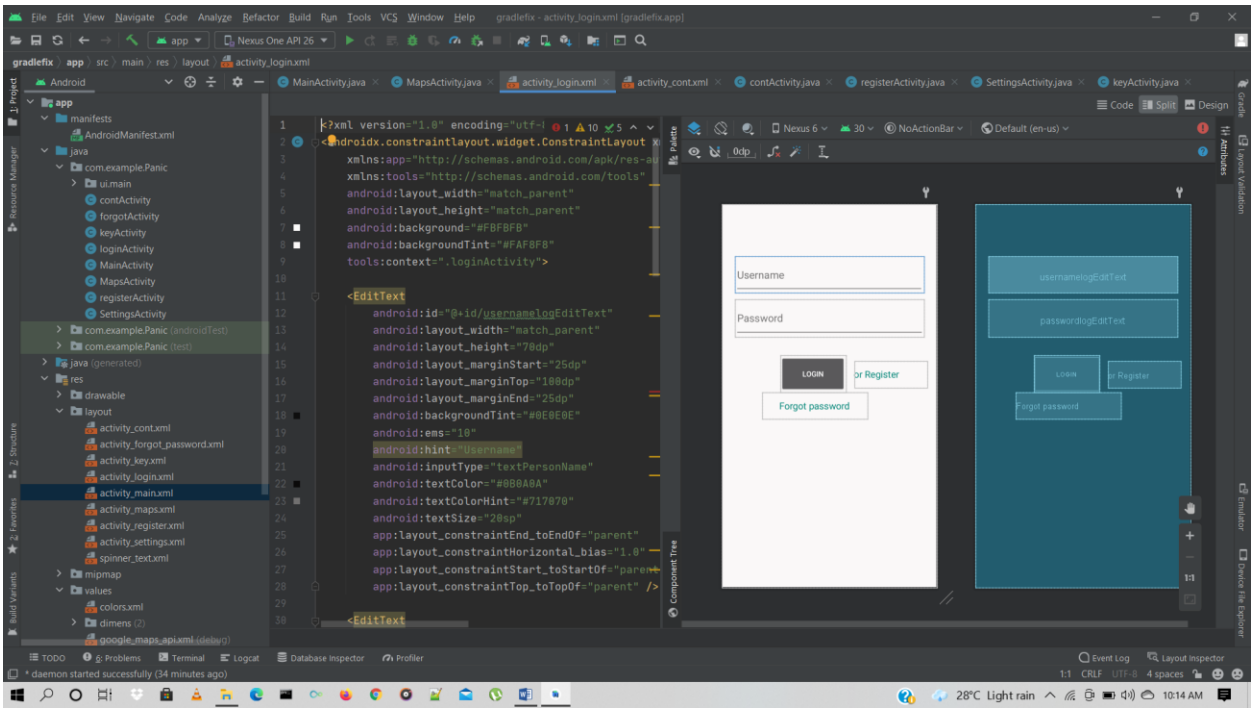

gradlefix - MainActivity.java [gradlefix.app]
package com.example.Panic;
import ...
public class MainActivity<timer> extends AppCompatActivity {
private SpeechRecognizer speechRecognizer;
private Intent intentRecognizer;
private TextView textView;
String robbery, kidnapping, fire, med, rape, assault, disaster, accident, explosion, phone, message, newMessage;
String address, cityName, userAddress;
Button panicButton;
public FusedLocationProviderClient fusedLocationClient;
public LatLng latLng;
public Location lastLocation;
LocationRequest myLocationRequest;
// private GoogleMap mMap;
LocationCallback myLocationCallback = new LocationCallback() {
@Override
public void onLocationResult(@NonNull LocationResult locationResult) {
super.onLocationResult(locationResult);
for (Location location : locationResult.getLocations()) {
if (getApplicationContext() != null) {
lastLocation = location;
locationResult.getLastLocation();
latLng = new LatLng(location.getLatitude(), location.getLongitude());
cityName = getCityName(latLng);
Log.d("TAG", "Address: "+ address);

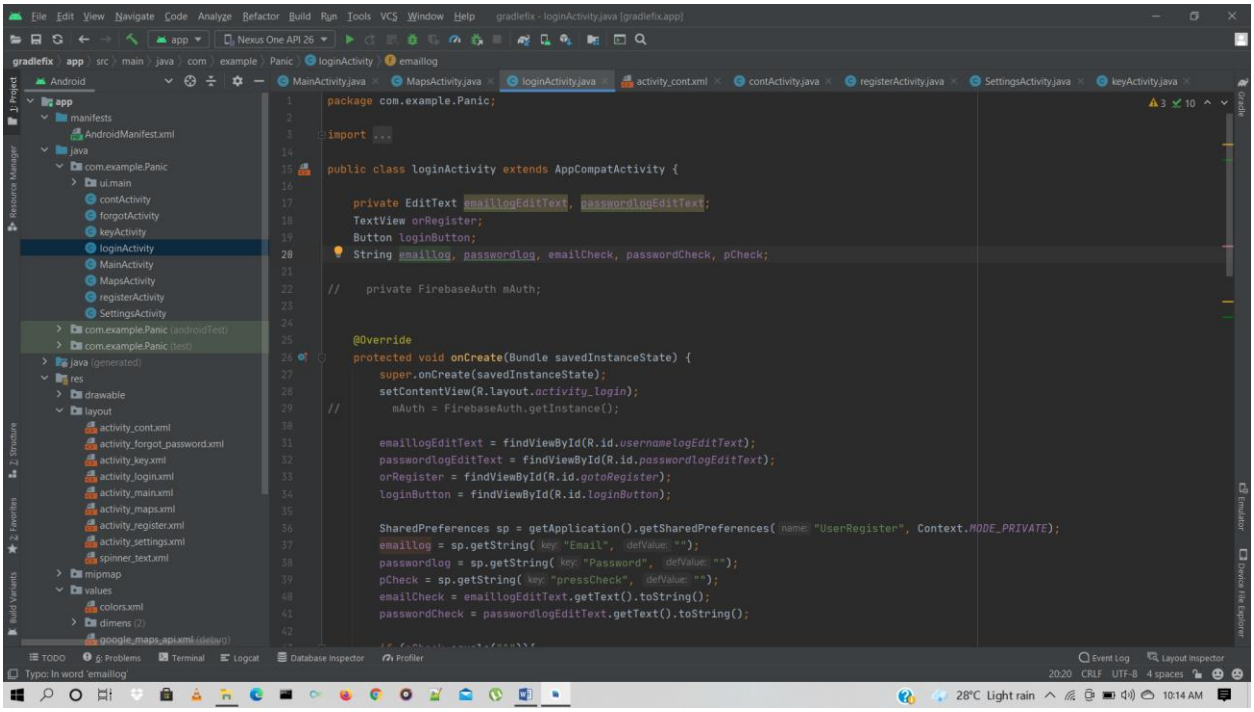
```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ActivityCompat.requestPermissions(this, new String[]{RECORD_AUDIO}, PackageManager.PERMISSION_GRANTED);
    ActivityCompat.requestPermissions(MainActivity.this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);

    fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);

    myLocationRequest = LocationRequest.create();
    myLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

    if(ContextCompat.checkSelfPermission(MainActivity.this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRAN
        fusedLocationClient.requestLocationUpdates(myLocationRequest,myLocationCallback,Looper.myLooper());
//          mMap.setMyLocationEnabled(true);
    }else{
        checkLocationPermission();
    }

    textView = findViewById(R.id.textView);
    intentRecognizer = new Intent(ACTION_RECOGNIZE_SPEECH);
    intentRecognizer.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);

    speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this);
    speechRecognizer.setRecognitionListener(new RecognitionListener() {
        @Override
        public void onReadyForSpeech(Bundle params) {
```

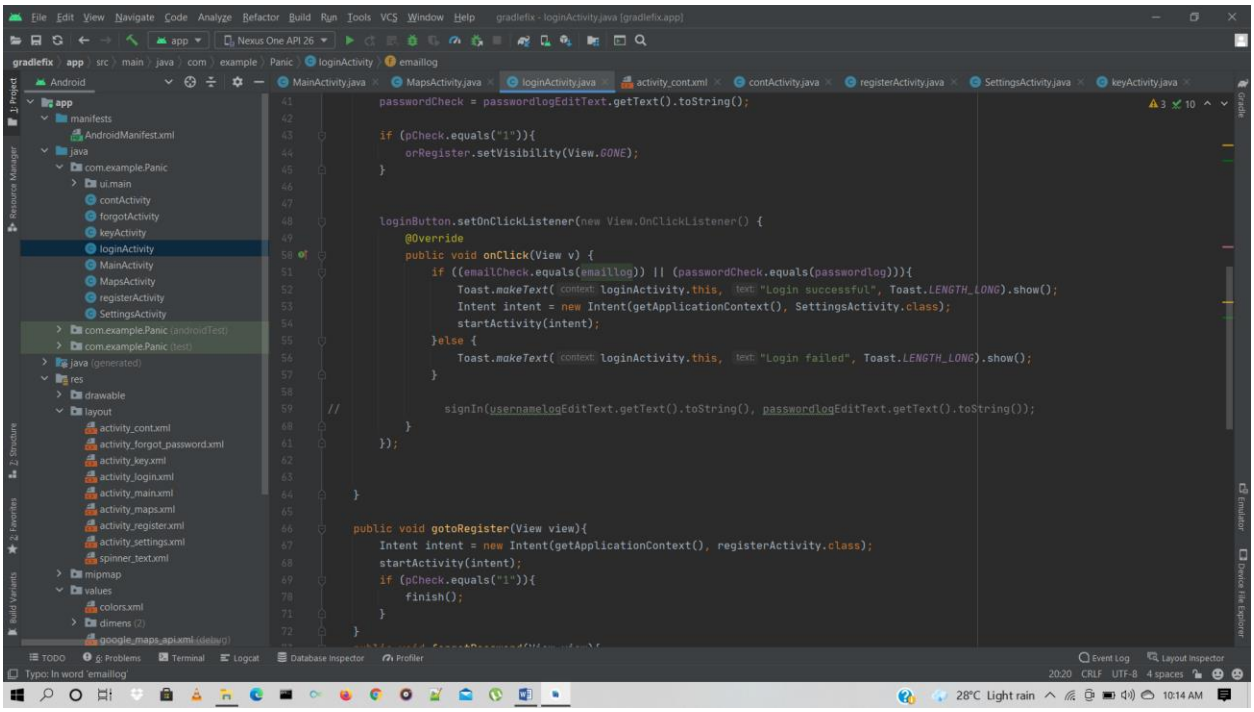
```java
@Override
public void onResults(Bundle bundle) {

    SharedPreferences sp = getApplication().getSharedPreferences("UserKeywords", Context.MODE_PRIVATE);
    robbery = sp.getString("Robbery", "");
    kidnapping = sp.getString("Kidnapping", "");
    fire = sp.getString("Fire", "");
    med = sp.getString("Med", "");
    rape = sp.getString("Rape", "");
    assault = sp.getString("Assault", "");
    disaster = sp.getString("Disaster", "");
    accident = sp.getString("Accident", "");
    explosion = sp.getString("Explosion", "");

    ArrayList<String> matches = bundle.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
    String string = "";
    if (matches != null) {
        string = matches.get(0);
        textView.setText(string);
        if (string.contains(robbery)) {
            textView.setText("Robbery");
        } else if (string.contains(kidnapping)) {
            textView.setText("Kidnap");
        } else if (string.contains(fire)) {
            textView.setText("Fire");
        } else if (string.contains(med)) {
            textView.setText("Medical");
        } else if (string.contains(rape)) {
            textView.setText("Rape");
        } else if (string.contains(assault)) {
            textView.setText("Assault");
        } else if (string.contains(disaster)) {
```

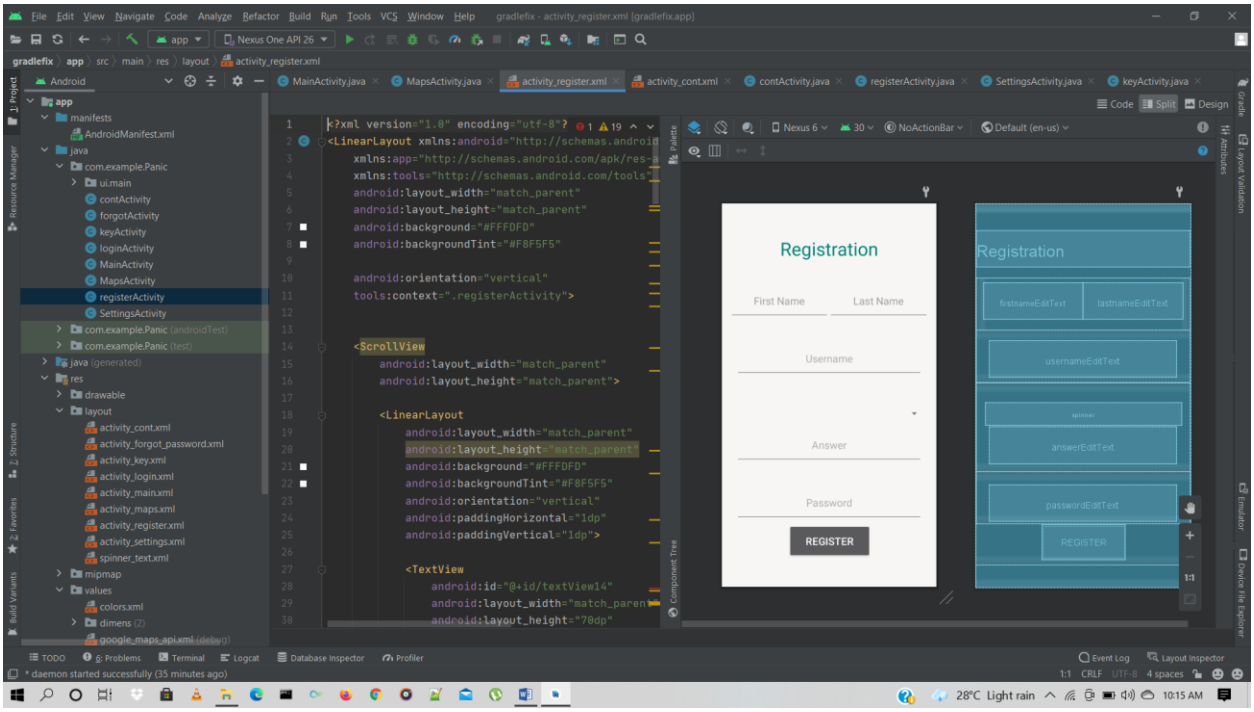
```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    ActivityCompat.requestPermissions( activity: MapsActivity.this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, requestCode: 1);
    // Obtain the SupportMapFragment and get notified when the map is ready to be used.
    fusedLocationClient = LocationServices.getFusedLocationProviderClient( activity: this);
    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync( callback: this);
}



LocationCallback myLocationCallback = onLocationResult(locationResult) -> {
        super.onLocationResult(locationResult);
        for(Location location : locationResult.getLocations()){
            if(getApplicationContext() != null){
                lastLocation = location;

                locationResult.getLastLocation();
                latLng = new LatLng(location.getLatitude(),location.getLongitude());

                String cityName = getCityName(latLng);

                mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
                mMap.animateCamera(CameraUpdateFactory.zoomTo(16));
            }

        }
};
```

```java
private String getCityName(LatLng latLng) {
    String myCity = "";
    Geocoder geocoder = new Geocoder( context: MapsActivity.this, Locale.getDefault());
    try {
        List<Address> addresses = geocoder.getFromLocation(latLng.latitude, latLng.longitude, maxResults: 1);
        String address = addresses.get(0).getAddressLine( index: 0);
        myCity = addresses.get(0).getLocality();
        Log.d( tag: "mylog", msg: "Complete Address: "+ addresses.toString());
        Log.d( tag: "mylog", msg: "Address: "+ address);
    } catch (IOException e) {
        e.printStackTrace();
    }
    return myCity;
}


/**
 * Manipulates the map once available.
 * This callback is triggered when the map is ready to be used.
 * This is where we can add markers or lines, add listeners or move the camera. In this case,
 * we just add a marker near Sydney, Australia.
 * If Google Play services is not installed on the device, the user will be prompted to install
 * it inside the SupportMapFragment. This method will only be triggered once the user has
 * installed Google Play services and returned to the app.
 */
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;


    myLocationRequest = LocationRequest.create();
```


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
gradlefix - activity_login.xml [gradlefix.app]
gradlefix app src main res layout activity_login.xml
MainActivity.java MapsActivity.java activity_login.xml activity_cont.xml contActivity.java registerActivity.java SettingsActivity.java keyActivity.java
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#FBFBFB"
android:backgroundTint="#FAF8F8"
tools:context=".loginActivity">
<EditText
android:id="@+id/usernamelogEditText"
android:layout_width="match_parent"
android:layout_height="70dp"
android:layout_marginStart="25dp"
android:layout_marginTop="100dp"
android:layout_marginEnd="25dp"
android:backgroundTint="#0E0E0E"
android:ems="10"
android:hint="Username"
android:inputType="textPersonName"
android:textColor="#0B0A0A"
android:textColorHint="#717070"
android:textSize="20sp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="1.0"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
<EditText
Username
Password
LOGIN
or Register
Forgot password
usernamelogEditText
passwordlogEditText
LOGIN
or Register
Forgot password
* daemon started successfully (34 minutes ago)
28°C Light rain
10:14 AM

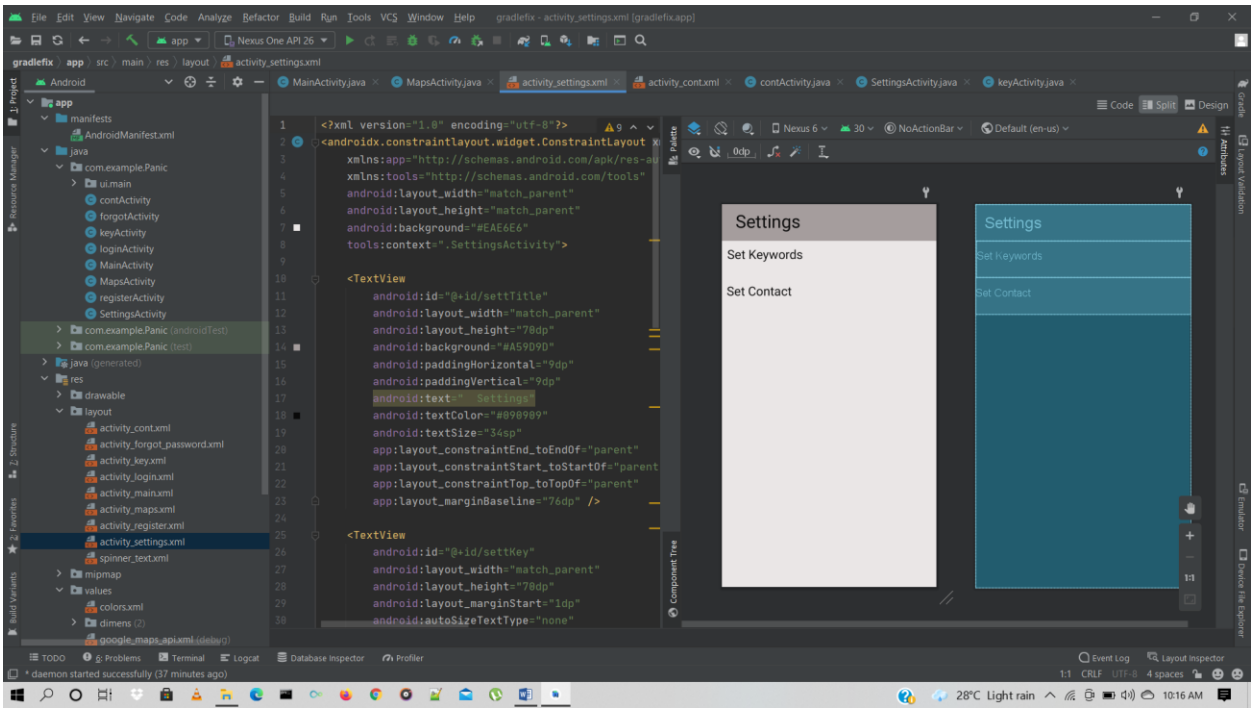

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
gradlefix - loginActivity.java [gradlefix.app]
gradlefix app src main java com example Panic loginActivity emaillog
MainActivity.java MapsActivity.java loginActivity.java activity_cont.xml contActivity.java registerActivity.java SettingsActivity.java keyActivity.java
package com.example.Panic;
import ...
public class loginActivity extends AppCompatActivity {
private EditText emaillogEditText, passwordlogEditText;
TextView orRegister;
Button loginButton;
String emaillog, passwordlog, emailCheck, passwordCheck, pCheck;
// private FirebaseAuth mAuth;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_login);
// mAuth = FirebaseAuth.getInstance();
emaillogEditText = findViewById(R.id.usernamelogEditText);
passwordlogEditText = findViewById(R.id.passwordlogEditText);
orRegister = findViewById(R.id.gotoRegister);
loginButton = findViewById(R.id.loginButton);
SharedPreferences sp = getApplication().getSharedPreferences(name: "UserRegister", Context.MODE_PRIVATE);
emaillog = sp.getString(key: "Email", defValue: "");
passwordlog = sp.getString(key: "Password", defValue: "");
pCheck = sp.getString(key: "pressCheck", defValue: "");
emailCheck = emaillogEditText.getText().toString();
passwordCheck = passwordlogEditText.getText().toString();
Typo: In word 'emaillog'
28°C Light rain
10:14 AM

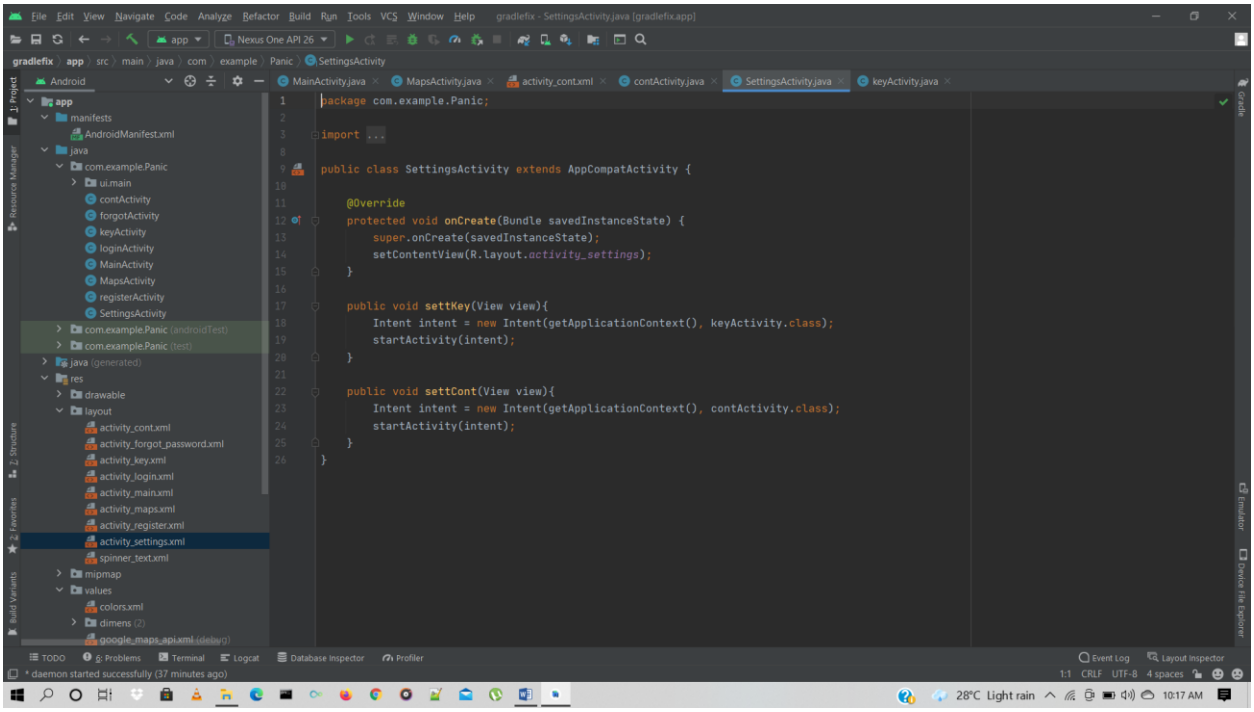

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
gradlefix - loginActivity.java [gradlefix.app]
gradlefix > app > src > main > java > com > example > Panic > loginActivity > emaillog
MainActivity.java
MapsActivity.java
loginActivity.java
activity_cont.xml
contActivity.java
registerActivity.java
SettingsActivity.java
keyActivity.java
passwordCheck = passwordlogEditText.getText().toString();
if (pCheck.equals("1")){
orRegister.setVisibility(View.GONE);
}
loginButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
if ((emailCheck.equals(emaillog)) || (passwordCheck.equals(passwordlog))){
Toast.makeText(context: loginActivity.this, text: "Login successful", Toast.LENGTH_LONG).show();
Intent intent = new Intent(getApplicationContext(), SettingsActivity.class);
startActivity(intent);
}else {
Toast.makeText(context: loginActivity.this, text: "Login failed", Toast.LENGTH_LONG).show();
}
// signIn(usernamelogEditText.getText().toString(), passwordlogEditText.getText().toString());
}
});
}
public void gotoRegister(View view){
Intent intent = new Intent(getApplicationContext(), registerActivity.class);
startActivity(intent);
if (pCheck.equals("1")){
finish();
}
}
TODO
6: Problems
Terminal
Logcat
Database Inspector
Profiler
Event Log
Layout Inspector
Typo: In word 'emaillog'
20:20 CRLF UTF-8 4 spaces
28°C Light rain
10:14 AM

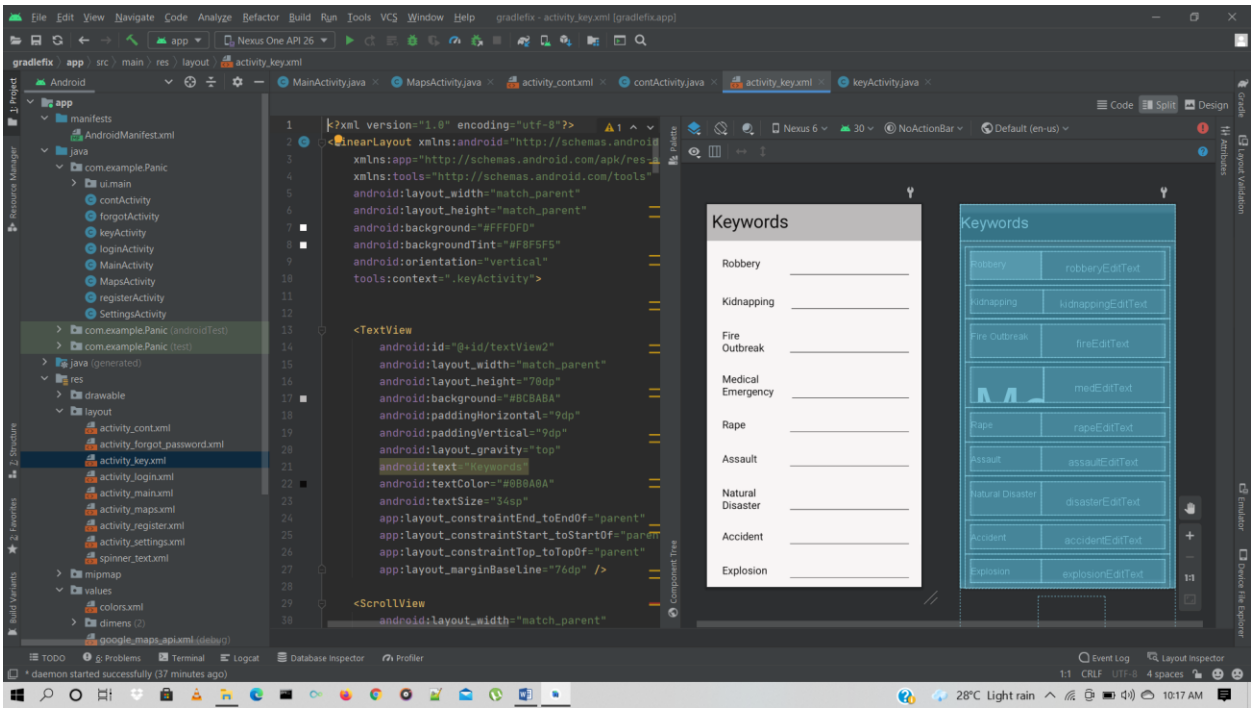

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
gradlefix - activity_register.xml [gradlefix.app]
gradlefix > app > src > main > res > layout > activity_register.xml
MainActivity.java
MapsActivity.java
activity_register.xml
activity_cont.xml
contActivity.java
registerActivity.java
SettingsActivity.java
keyActivity.java
<?xml version="1.0" encoding="utf-8"?
<LinearLayout xmlns:android="http://schemas.androi
xmlns:app="http://schemas.android.com/apk/res-
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#FFFDFD"
android:backgroundTint="#F8F5F5"
android:orientation="vertical"
tools:context=".registerActivity">
<ScrollView
android:layout_width="match_parent"
android:layout_height="match_parent">
<LinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#FFFDFD"
android:backgroundTint="#F8F5F5"
android:orientation="vertical"
android:paddingHorizontal="1dp"
android:paddingVertical="1dp">
<TextView
android:id="@+id/textView14"
Registration
First Name
Last Name
Username
Answer
Password
REGISTER
firstnameEditText
lastnameEditText
usernameEditText
answerEditText
passwordEditText
TODO
6: Problems
Terminal
Logcat
Database Inspector
Profiler
Event Log
Layout Inspector
* daemon started successfully (35 minutes ago)
1:1 CRLF UTF-8 4 spaces
28°C Light rain
10:15 AM

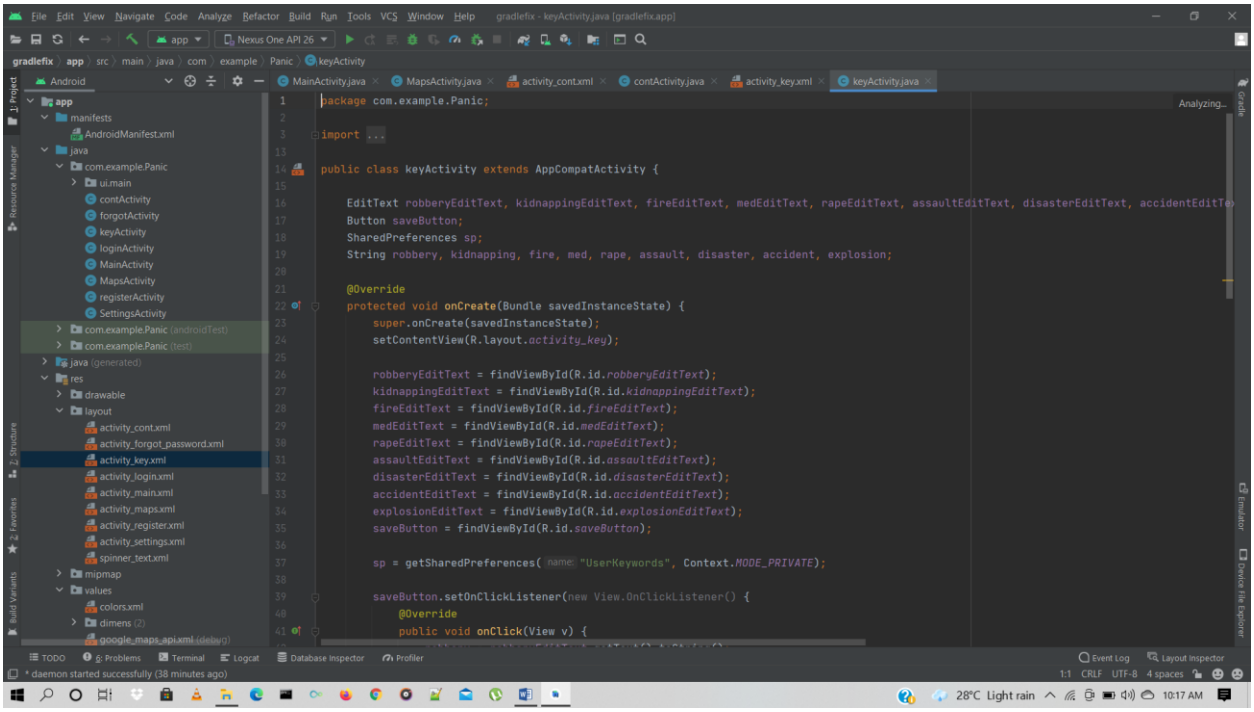
```java
package com.example.Panic;

import ...

public class registerActivity extends AppCompatActivity implements
        AdapterView.OnItemSelectedListener{

    private EditText firstnameEditText, lastnameEditText, usernameEditText, answerEditText, passwordEditText;
    private String firstName, lastName, email, password, answer, authQues, fName, lName, eMail, pWord;
    Button registerButton;
    SharedPreferences sp;
    String pCheck = "0";



        String[] backUpQuestion = {"Select question...", "What is the last name of your favorite president?", "What is your first pet's nam



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        Spinner spinner = findViewById(R.id.spinner);
        spinner.setOnItemSelectedListener(this);
        ArrayAdapter adapter = new ArrayAdapter( context: this, R.layout.spinner_text, backUpQuestion);
//        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this, R.array.Backup_question, android.R.layout.simple_spinne
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner.setAdapter(adapter);

        firstnameEditText = findViewById(R.id.firstnameEditText);
        lastnameEditText = findViewById(R.id.lastnameEditText);
```

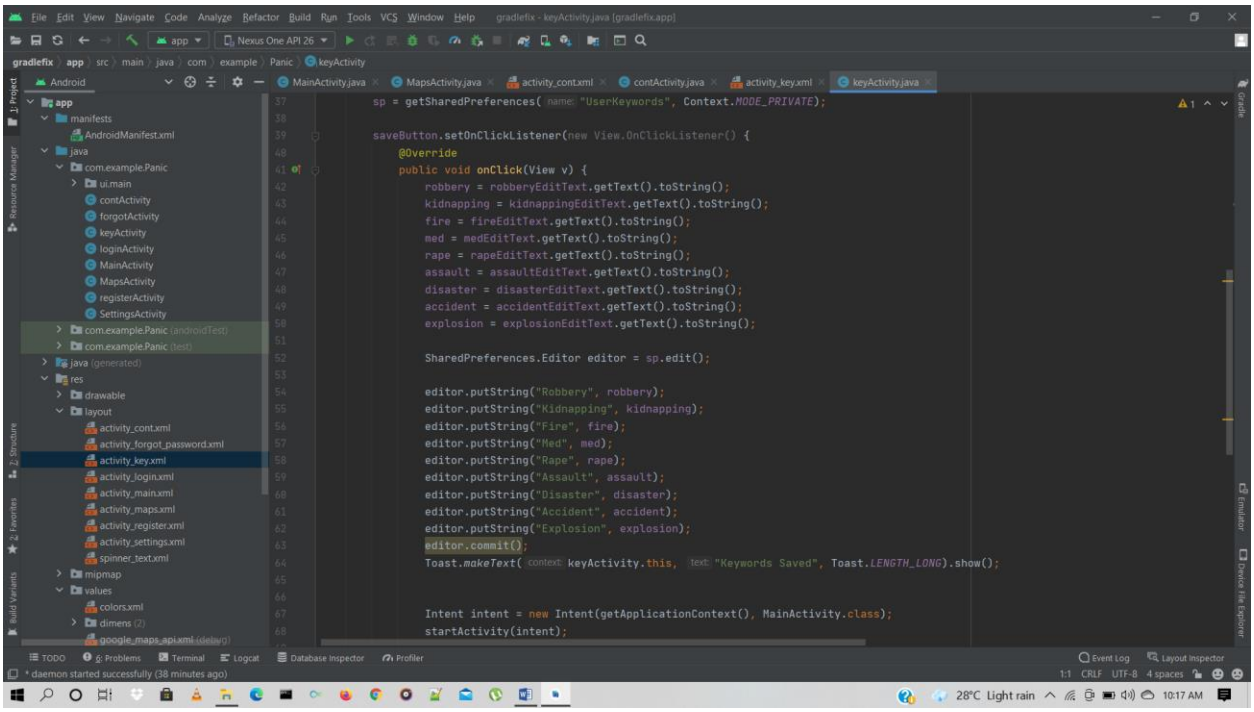
```java
        registerButton = findViewById(R.id.registerButton);


        sp = getSharedPreferences( name: "UserRegister", Context.MODE_PRIVATE);



        registerButton.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                pCheck = "1";
                firstName = firstnameEditText.getText().toString();
                lastName = lastnameEditText.getText().toString();
                answer = answerEditText.getText().toString();
                email = usernameEditText.getText().toString();
                password = passwordEditText.getText().toString();
                authQues = spinner.getSelectedItem().toString();
                Log.d( tag: "MyLog",  msg: "HERE: "+authQues);


                SharedPreferences.Editor editor = sp.edit();

                editor.putString("Firstname", firstName);
                editor.putString("Lastname", lastName);
                editor.putString("Email", email);
                editor.putString("Auth_Question", authQues);
                editor.putString("Auth_Answer", answer);
                editor.putString("Password", password);
                editor.putString("pressCheck", pCheck);
                editor.commit();
                Toast.makeText( context: registerActivity.this,  text: "Registered successfully", Toast.LENGTH_LONG).show();
```

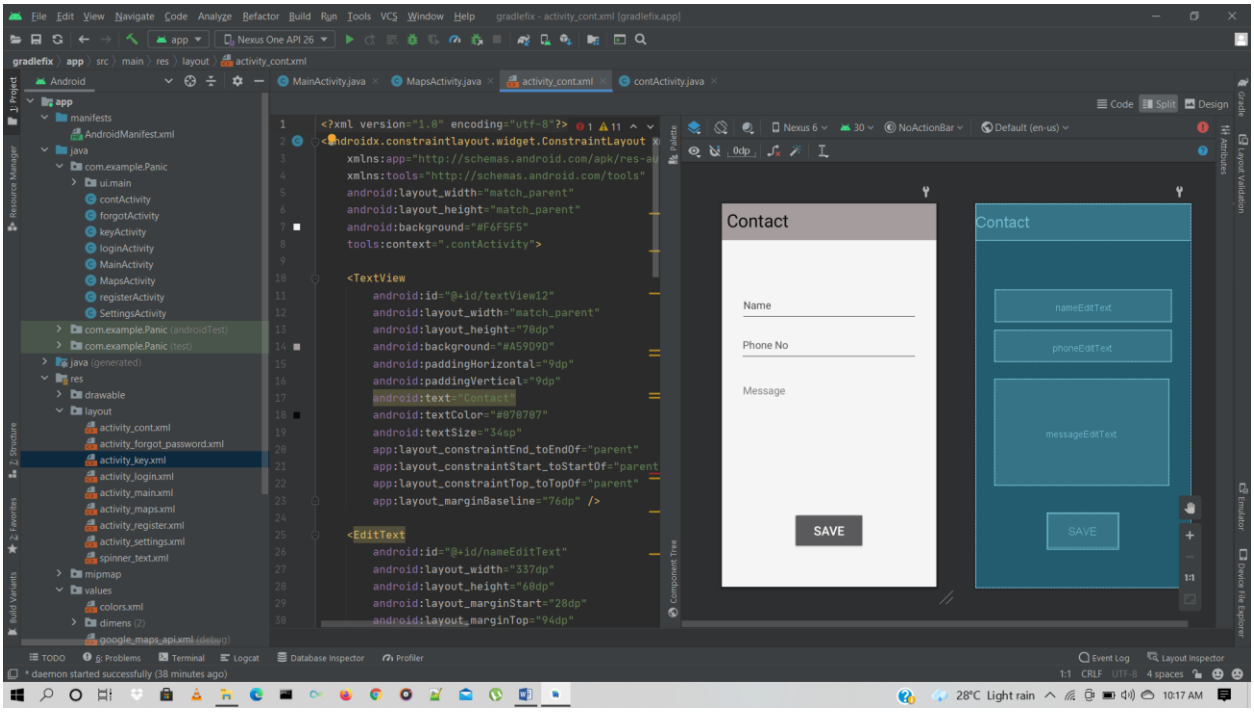

gradlefix - activity_settings.xml [gradlefix.app]
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:app="http://schemas.android.com/apk/res-au
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#EAE6E6"
tools:context=".SettingsActivity">
<TextView
android:id="@+id/settTitle"
android:layout_width="match_parent"
android:layout_height="70dp"
android:background="#A59D9D"
android:paddingHorizontal="9dp"
android:paddingVertical="9dp"
android:text=" Settings"
android:textColor="#090909"
android:textSize="34sp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent
app:layout_constraintTop_toTopOf="parent"
app:layout_marginBaseline="76dp" />
<TextView
android:id="@+id/settKey"
android:layout_width="match_parent"
android:layout_height="70dp"
android:layout_marginStart="1dp"
android:autoSizeTextType="none"
Settings
Set Keywords
Set Contact
Settings
Set Keywords
Set Contact


gradlefix - SettingsActivity.java [gradlefix.app]
package com.example.Panic;
import ...
public class SettingsActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_settings);
}
public void settKey(View view){
Intent intent = new Intent(getApplicationContext(), keyActivity.class);
startActivity(intent);
}
public void settCont(View view){
Intent intent = new Intent(getApplicationContext(), contActivity.class);
startActivity(intent);
}
}


gradlefix - activity_key.xml [gradlefix.app]
activity_key.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android
xmlns:app="http://schemas.android.com/apk/res-
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#FFFDFD"
android:backgroundTint="#F8F5F5"
android:orientation="vertical"
tools:context=".keyActivity">
<TextView
android:id="@+id/textView2"
android:layout_width="match_parent"
android:layout_height="70dp"
android:background="#BCBABA"
android:paddingHorizontal="9dp"
android:paddingVertical="9dp"
android:layout_gravity="top"
android:text="Keywords"
android:textColor="#0B0A0A"
android:textSize="34sp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_marginBaseline="76dp" />
<ScrollView
android:layout_width="match_parent"
Keywords
Robbery
Kidnapping
Fire Outbreak
Medical Emergency
Rape
Assault
Natural Disaster
Accident
Explosion
Keywords
robberyEditText
kidnappingEditText
fireEditText
medEditText
rapeEditText
assaultEditText
disasterEditText
accidentEditText
explosionEditText

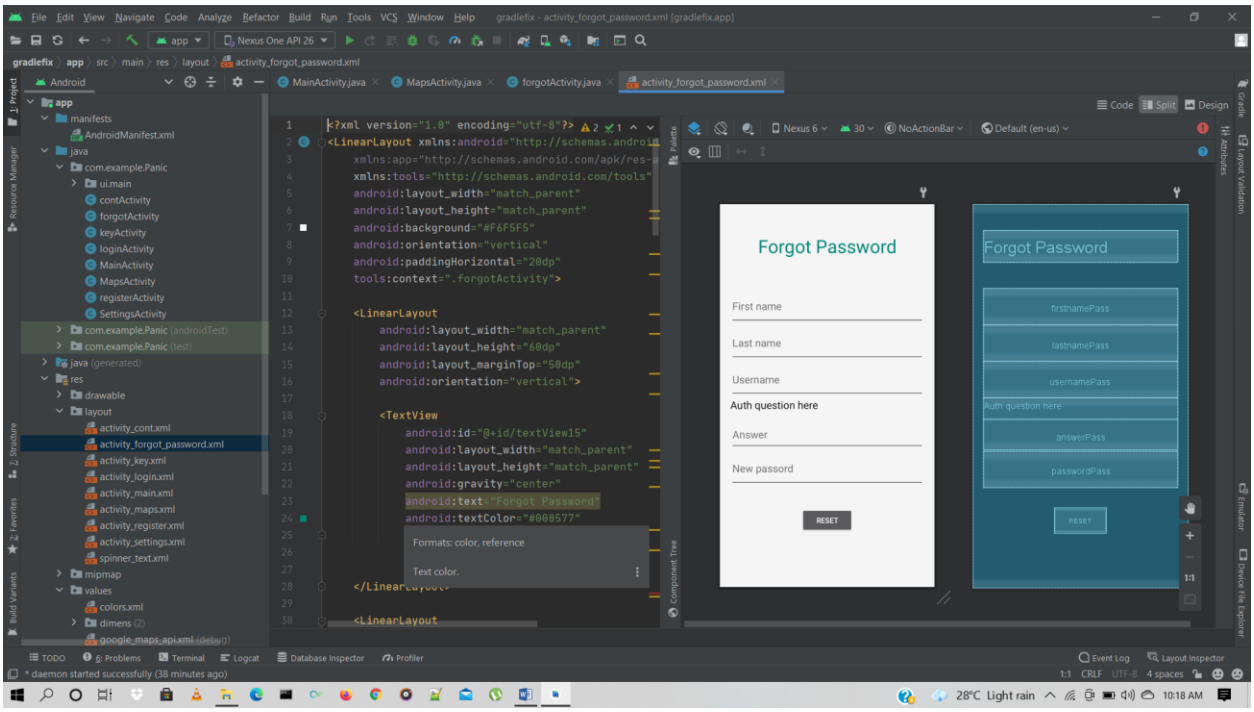

gradlefix - keyActivity.java [gradlefix.app]
package com.example.Panic;
import ...
public class keyActivity extends AppCompatActivity {
EditText robberyEditText, kidnappingEditText, fireEditText, medEditText, rapeEditText, assaultEditText, disasterEditText, accidentEditTe
Button saveButton;
SharedPreferences sp;
String robbery, kidnapping, fire, med, rape, assault, disaster, accident, explosion;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_key);
robberyEditText = findViewById(R.id.robberyEditText);
kidnappingEditText = findViewById(R.id.kidnappingEditText);
fireEditText = findViewById(R.id.fireEditText);
medEditText = findViewById(R.id.medEditText);
rapeEditText = findViewById(R.id.rapeEditText);
assaultEditText = findViewById(R.id.assaultEditText);
disasterEditText = findViewById(R.id.disasterEditText);
accidentEditText = findViewById(R.id.accidentEditText);
explosionEditText = findViewById(R.id.explosionEditText);
saveButton = findViewById(R.id.saveButton);
sp = getSharedPreferences(name: "UserKeywords", Context.MODE_PRIVATE);
saveButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {

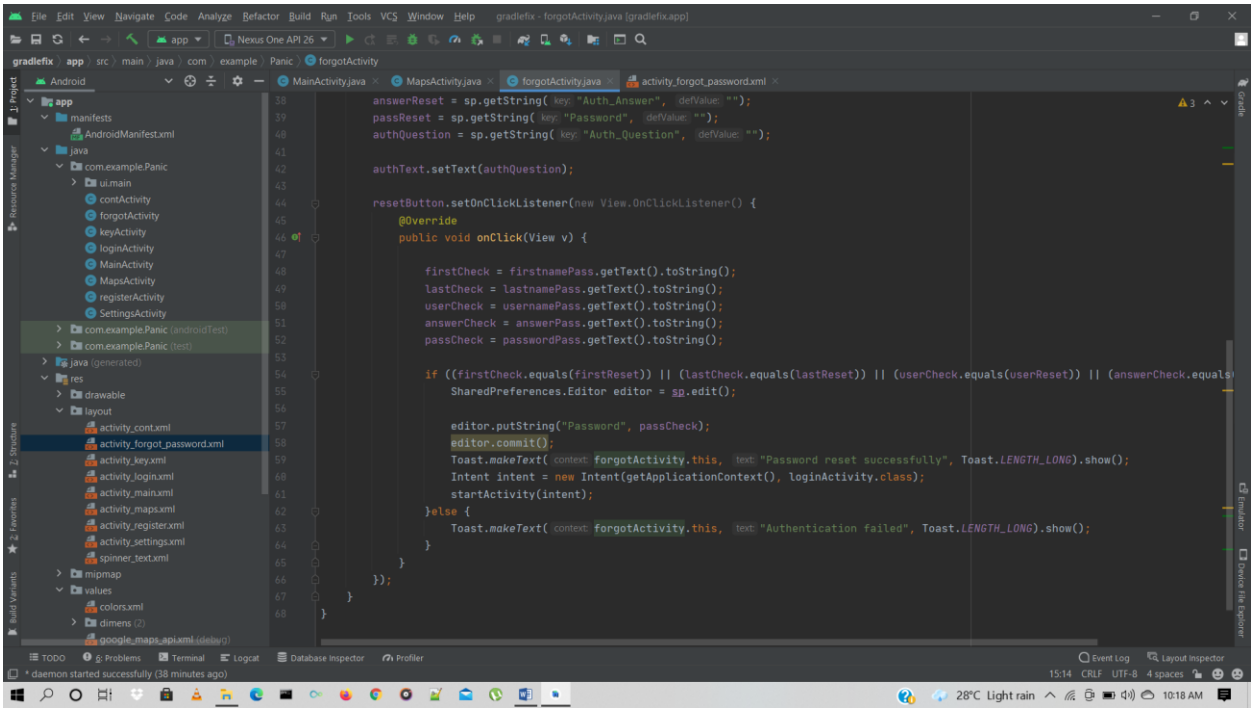

gradlefix - keyActivity.java [gradlefix.app]
sp = getSharedPreferences(name: "UserKeywords", Context.MODE_PRIVATE);
saveButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
robbery = robberyEditText.getText().toString();
kidnapping = kidnappingEditText.getText().toString();
fire = fireEditText.getText().toString();
med = medEditText.getText().toString();
rape = rapeEditText.getText().toString();
assault = assaultEditText.getText().toString();
disaster = disasterEditText.getText().toString();
accident = accidentEditText.getText().toString();
explosion = explosionEditText.getText().toString();
SharedPreferences.Editor editor = sp.edit();
editor.putString("Robbery", robbery);
editor.putString("Kidnapping", kidnapping);
editor.putString("Fire", fire);
editor.putString("Med", med);
editor.putString("Rape", rape);
editor.putString("Assault", assault);
editor.putString("Disaster", disaster);
editor.putString("Accident", accident);
editor.putString("Explosion", explosion);
editor.commit();
Toast.makeText(context: keyActivity.this, text: "Keywords Saved", Toast.LENGTH_LONG).show();
Intent intent = new Intent(getApplicationContext(), MainActivity.class);
startActivity(intent);

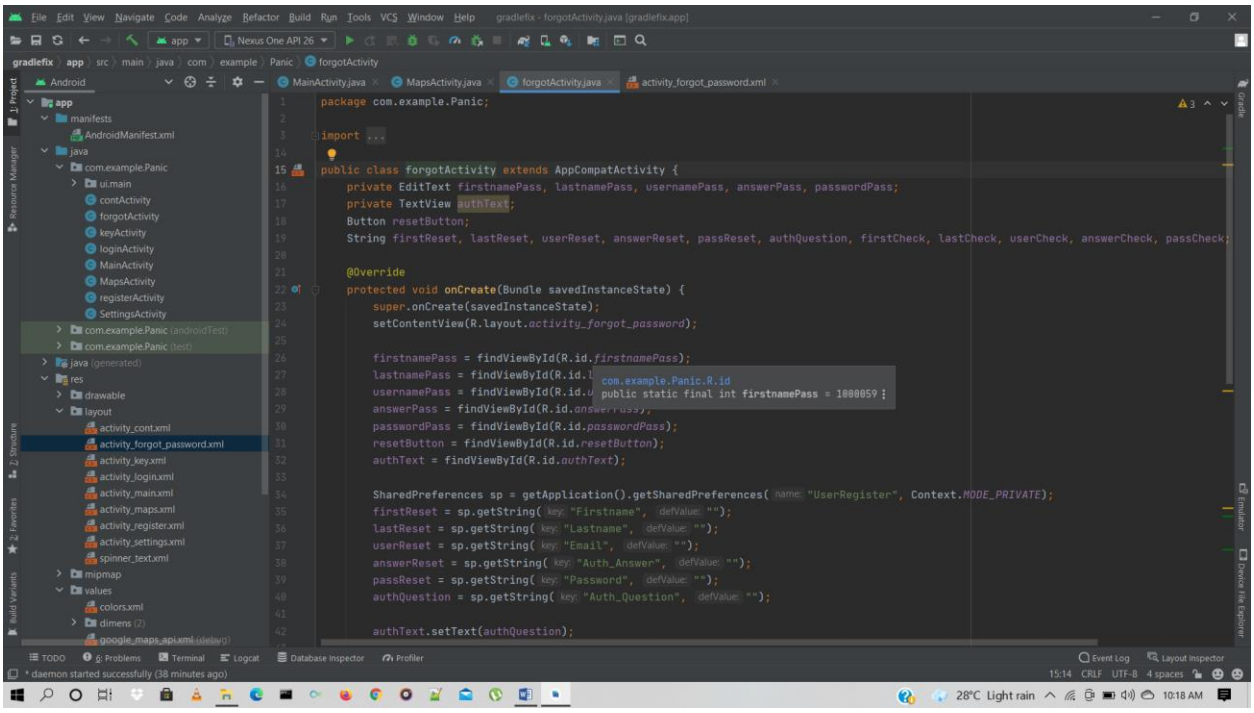

gradlefix - activity_cont.xml [gradlefix.app]
<?xml version="1.8" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:app="http://schemas.android.com/apk/res-au
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#F6F5F5"
tools:context=".contActivity">
<TextView
android:id="@+id/textView12"
android:layout_width="match_parent"
android:layout_height="70dp"
android:background="#A59D9D"
android:paddingHorizontal="9dp"
android:paddingVertical="9dp"
android:text="Contact"
android:textColor="#070707"
android:textSize="34sp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent
app:layout_constraintTop_toTopOf="parent"
app:layout_marginBaseline="76dp" />
<EditText
android:id="@+id/nameEditText"
android:layout_width="337dp"
android:layout_height="60dp"
android:layout_marginStart="28dp"
android:layout_marginTop="94dp"
Contact
Name
Phone No
Message
SAVE
Contact
nameEditText
phoneEditText
messageEditText
SAVE

```java
package com.example.Panic;

import ...

public class contActivity extends AppCompatActivity {

    EditText nameEditText, phoneEditText, messageEditText;
    Button addButton;
    SharedPreferences sp;
    String name, phone, message;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cont);

        nameEditText = findViewById(R.id.nameEditText);
        phoneEditText = findViewById(R.id.phoneEditText);
        messageEditText = findViewById(R.id.messageEditText);
        addButton = findViewById(R.id.addButton);

        sp = getSharedPreferences( name: "UserContact", Context.MODE_PRIVATE);

        addButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Log.d(TAG,  msg: "Here we go");
                name = nameEditText.getText().toString();
                phone = phoneEditText.getText().toString();
                message = messageEditText.getText().toString();
                Log.d(TAG,  msg: "Lets go");
```

```java
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cont);

        nameEditText = findViewById(R.id.nameEditText);
        phoneEditText = findViewById(R.id.phoneEditText);
        messageEditText = findViewById(R.id.messageEditText);
        addButton = findViewById(R.id.addButton);

        sp = getSharedPreferences( name: "UserContact", Context.MODE_PRIVATE);

        addButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Log.d(TAG,  msg: "Here we go");
                name = nameEditText.getText().toString();
                phone = phoneEditText.getText().toString();
                message = messageEditText.getText().toString();
                Log.d(TAG,  msg: "Lets go");

                SharedPreferences.Editor editor = sp.edit();

                editor.putString("Name", name);
                editor.putString("Phone", phone);
                editor.putString("Message", message);
                editor.commit();
                Log.d(TAG,  msg: "Here we go again");

                Toast.makeText( context: contActivity.this,  text: "Information Saved", Toast.LENGTH_LONG).show();


                Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                startActivity(intent);
```


gradlefix - activity_forgot_password.xml [gradlefix.app]
MainActivity.java
MapsActivity.java
forgotActivity.java
activity_forgot_password.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android
xmlns:app="http://schemas.android.com/apk/res-a
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#F6F5F5"
android:orientation="vertical"
android:paddingHorizontal="20dp"
tools:context=".forgotActivity">
<LinearLayout
android:layout_width="match_parent"
android:layout_height="60dp"
android:layout_marginTop="50dp"
android:orientation="vertical">
<TextView
android:id="@+id/textView15"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center"
android:text="Forgot Password"
android:textColor="#008577"
Formats: color, reference
Text color.
Forgot Password
First name
Last name
Username
Auth question here
Answer
New passord
RESET
firstnamePass
lastnamePass
usernamePass
answerPass
passwordPass


gradlefix - forgotActivity.java [gradlefix.app]
MainActivity.java
MapsActivity.java
forgotActivity.java
activity_forgot_password.xml
answerReset = sp.getString(key: "Auth_Answer", defValue: "");
passReset = sp.getString(key: "Password", defValue: "");
authQuestion = sp.getString(key: "Auth_Question", defValue: "");
authText.setText(authQuestion);
resetButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
firstCheck = firstnamePass.getText().toString();
lastCheck = lastnamePass.getText().toString();
userCheck = usernamePass.getText().toString();
answerCheck = answerPass.getText().toString();
passCheck = passwordPass.getText().toString();
if ((firstCheck.equals(firstReset)) || (lastCheck.equals(lastReset)) || (userCheck.equals(userReset)) || (answerCheck.equals
SharedPreferences.Editor editor = sp.edit();
editor.putString("Password", passCheck);
editor.commit();
Toast.makeText(context: forgotActivity.this, text: "Password reset successfully", Toast.LENGTH_LONG).show();
Intent intent = new Intent(getApplicationContext(), loginActivity.class);
startActivity(intent);
}else {
Toast.makeText(context: forgotActivity.this, text: "Authentication failed", Toast.LENGTH_LONG).show();
}
}
});
}
}


gradlefix - forgotActivity.java [gradlefix.app]
gradlefix > app > src > main > java > com > example > Panic > forgotActivity
MainActivity.java
MapsActivity.java
forgotActivity.java
activity_forgot_password.xml
package com.example.Panic;
import ...
public class forgotActivity extends AppCompatActivity {
private EditText firstnamePass, lastnamePass, usernamePass, answerPass, passwordPass;
private TextView authText;
Button resetButton;
String firstReset, lastReset, userReset, answerReset, passReset, authQuestion, firstCheck, lastCheck, userCheck, answerCheck, passCheck;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_forgot_password);
firstnamePass = findViewById(R.id.firstnamePass);
lastnamePass = findViewById(R.id.l
usernamePass = findViewById(R.id.u
answerPass = findViewById(R.id.answerPass);
passwordPass = findViewById(R.id.passwordPass);
resetButton = findViewById(R.id.resetButton);
authText = findViewById(R.id.authText);
com.example.Panic.R.id
public static final int firstnamePass = 1080059
SharedPreferences sp = getApplication().getSharedPreferences(name: "UserRegister", Context.MODE_PRIVATE);
firstReset = sp.getString(key: "Firstname", defValue: "");
lastReset = sp.getString(key: "Lastname", defValue: "");
userReset = sp.getString(key: "Email", defValue: "");
answerReset = sp.getString(key: "Auth_Answer", defValue: "");
passReset = sp.getString(key: "Password", defValue: "");
authQuestion = sp.getString(key: "Auth_Question", defValue: "");
authText.setText(authQuestion);
* daemon started successfully (38 minutes ago)
28°C Light rain
10:18 AM