# Family Tracker

By

UMA MAHESHWAR REDDY MANDADI VENKATA

B.Tech., JNTU, 2015

A Report

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2017

Approved by:

Major Professor
Dr. Daniel Andresen

## Copyright

## Abstract

People always want know the whereabouts of their loved ones. They want to make sure everything is fine and that they are safe all the time. In this modern age, almost everyone owns a smart phone, which they tend to keep on them all the time. By using these smartphones, we can locate a person anywhere in this world. Android is open source software stack and has the highest smart phone user base. Hence, this application is developed in android.

The Family Tracker is an android Application used for finding whereabouts of a person in the most efficient and simple manner. Here we will be having guardians and subjects (users), where users add other people as their guardians. When the user does not lift the phone, the details of his location are sent to the guardian in the form of an SMS. Here the address along with the location link will be sent, using which we can locate it in the map. Guardian can also get the location of user by sending out an SMS to his phone. A one click messaging service is also implemented in this application through which guardian can send message to user's friends in one click for further inquiry. In order to make this application more helpful for the users, Geo-Fencing is implemented through which guardian can know whether the subject is in or around a particular place (location) of interest (Ex: a university).

# Table of Contents

# List of Figures

# List of Tables

## Acknowledgements

I would like to extend my sincere thanks to my academic advisor, Dr. Daniel Andresen, who encouraged and motivated me in developing this application and constant guidance throughout the development of this project. I take immense pleasure in expressing my sincere gratitude to my committee members, Dr. Mitchell Neilsen and Dr. Torben Amtoft, who despite of their busy schedule made time to serve on my committee and for providing their valuable feedback on the project.

I would like to extend my sincere gratitude for the academic and technical staff of the Computer Science at K-State for their support throughout my graduate studies by providing the required resources.

This would be incomplete without acknowledging the love and support of my parents and Friends who always stood by me at every point of life.

# Chapter 1 - Project Description

## 1.1 Introduction

Family Tracker is an android application, which serves users to track the location of their beloved ones and make sure they are safe all the time. The only means for people who does not own a smartphone to know the location of others, until now, is through direct communication between them. Parents can know the location of their children through call or SMS or some other kind of communication between them and their children. There will be certain times where their children will not communicate back for a long time for several reasons. This makes some parents nervous and worried. In order to overcome this limitation of manual communication regarding the whereabouts of their location this application is developed. This application sends the location of the children with exact address and a link to the Google Maps to their parents when their children does not communicate (take the call) with them. Parents can follow up the call with a message to a list of their child's friends to get more information in case needed with one click. The main aim of this application is to help parents know the whereabouts of their children all the time.

## 1.2 Motivation

My motivation to develop this application is from my very own experience. My mother is a bit more protective of me. She used to call me every hour to know my whereabouts when I step out of the house. Whenever I failed to take her call, she would be worried. She then used to call all my friends to check whether I was with any one of them. These events in my childhood motivated me to develop this application, which I think would be helpful for all the parents in this world.

The motivation behind choosing Android as the application development platform is its wide usage across the globe and it being an open source. I have hands-on-experience working with Java. Therefore, I was comfortable developing this application.

# Chapter 2 - Background

Android is an open source Linux based stack of software components comprising applications, an operating system, run-time environment, middleware, services and libraries [1]. These are used in wide array of devices such as smartphones, tablets etc. The following diagram shows the major components of the Android platform [2].

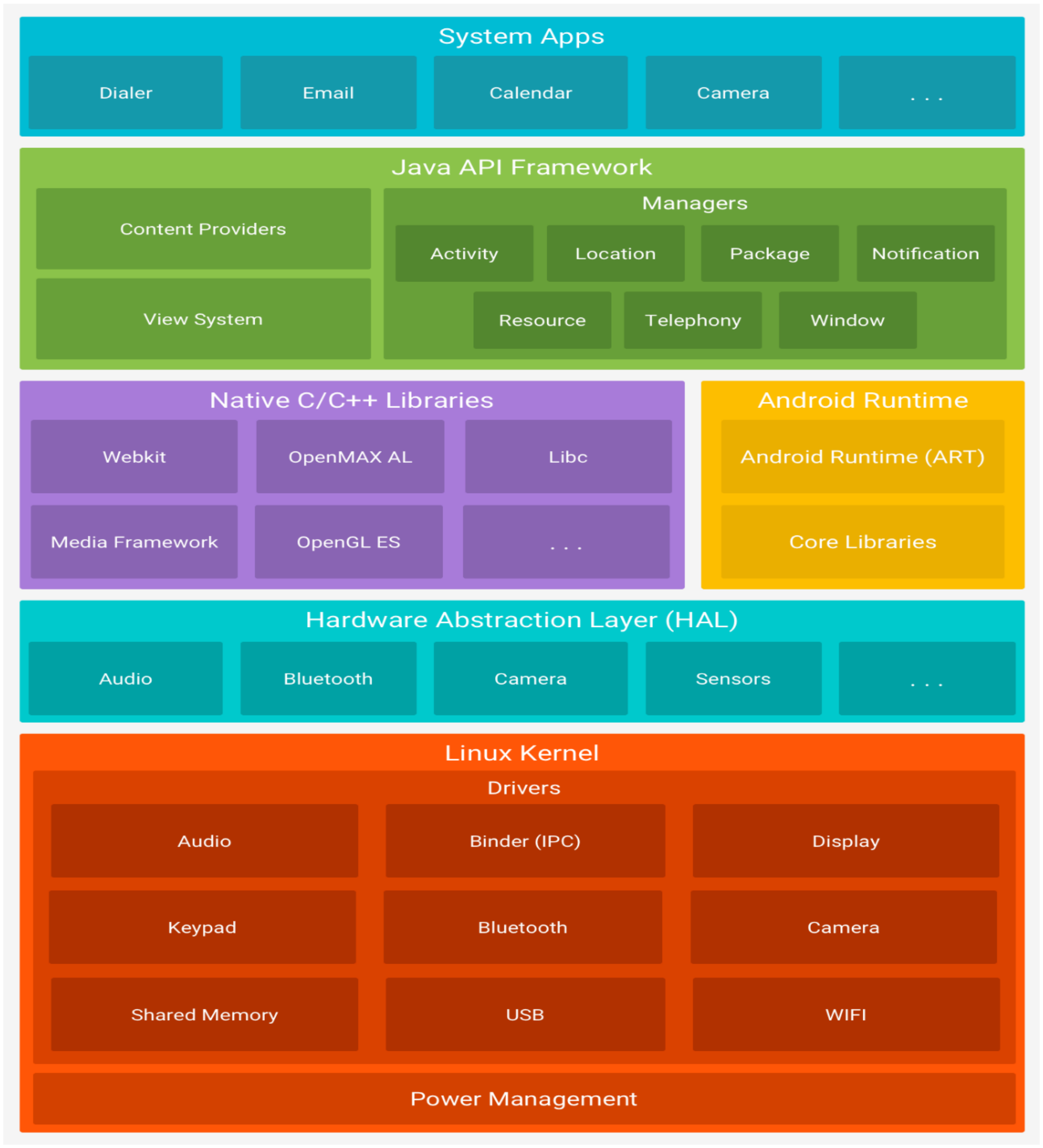


***Figure 1:*** *The Android software stack [2]*

Each layer is explained in detail in this chapter.

**Linux Kernel**

Linux kernel is the foundation of Android. It is the bottom layer of the Android software stack. It provides device drivers for hardware such as the device display, Wi-Fi, Bluetooth, camera and audio.

**Hardware Abstraction Layer (HAL)**

The hardware abstraction layer (HAL) contains interfaces that expose device hardware capabilities to the higher-level Java API framework. It consists of library modules, each of which implements an interface for a specific type of hardware component, such as the camera or Bluetooth module.

**Android Runtime**

Android runtime (ART) is the managed runtime used by applications and system services on Android.

**Android Libraries**

Android applications are mostly developed using the Java programming language. The Android platform provides Java framework APIs to expose the functionality of these native libraries to apps.

**Java API Framework**

Application framework provides the environment to manage and run android applications. The entire feature-set of the Android OS is available to you through APIs written in the Java language. In this application, I used location and Telephony services from Java API Framework in this application.

## System Applications

System applications are located at the top of the Android software stack. They comprise both the native applications for email, SMS, calendars, internet browsing, contacts and the third party applications installed by the user. The applications that we develop are installed in this layer.

# Chapter 3 - Related Work

## 3.1 Existing System

The existing system, to find the location of a person is by using application, uses GPS in mobile phones. All these applications requires the guardian (tracker) to be connected to the internet all the time. They integrate maps into their application, which results in application with higher CPU utilization. In most of the applications, every single time user need to open the application and track the person on map. Many people find it had to trace the actual location (street address) of the subject in the map. These applications also require Smartphone be be present at guardian's end. There is no application in the market, which gives the user location, implements geo-fencing, and providing a one-click messaging service, which I feel, is necessary to keep track of subject's whereabouts.

### 3.1.1 Disadvantages

- Time Consuming
- Stressful
- Challenging

## 3.2 Proposed System

The proposed system makes tracking simple. There is no need of an Internet connection at the guardian's end. No need for any kind of application to track the location of subject (child). Guardian need not have a Smart phone to get the location of the person. Location, the exact street address of the person is send to the guardian through a text message (SMS). A link to google maps is also provided with the SMS incase the guardian wants to track the subject's location on map. Geo-fencing is also implemented along with the location tracker to check whether the subject is inside specific area specified by guardian.

### 3.2.1 Advantages

- Works without Internet at Guardian's end

- Cost and Time efficient
- Hassle-free

### 3.2.2 Purpose of the System

Family Tracker is developed to provide the functionalities of a location tracker, messaging app on an Android powered device. This provides flexibility to user. Guardian can know the location of a person with a call to the person's mobile phone. Exact street address of the person is provided. Guardian does not require to be connected to the internet. Guardian does not have to own a smart phone.

### 3.2.3 Objective

The objective of this application is to provide the guardian with the location of the person (children) whenever the children are unable to take is call. This is developed as a security app keeping the concerned parents in mind.

### 3.2.4 System Overview

After the user downloads the Family Tracker application from the play store, he should install it on his Android powered device with a network connection. The user can check the options available on the home page and proceed with the required functionality.

# Chapter 4 - Requirement Analysis

Requirement Analysis is the first and important phase of the software developing activity in developing any kind of application effectively. I started to list out what are all the functionalities that my application should provide. There has been some minor changes with respect to the functionalities over the course of development of application. After few a meetings with my Professor Dr. Daniel Andresen, we came up with the following requirements for the application.

## 4.1 Functional Requirements

- Add Guardians: Application should have a module for registering list of guardians to it.
- Add Friends: Application should have a module for registering list of friends to it.
- Location sender: Application should have a module that can send location of user to guardian through SMS whenever user is unable to attend the guardians call.
- Messaging Service: Application should be able to send dynamic messages to list of friends with one click.

## 4.2 Non-Functional Requirements

Non-functional requirements are not directly related to the functional behavior of the system.

- This application is user friendly and very interactive.
- The user interface is designed in such way that novice users with little knowledge of Android will be able to access this application.
- Users are required to have some knowledge regarding google maps.

## 4.3 Software Specifications of development system

- Operating System: Windows 8.1
- Platform: Android SDK Framework 10 or higher
- Database: SQLite Database

- IDE: Android Studio IDE
- Technologies used: Java, Android, SQLite
- Emulator: SDK version 3.0 or higher

## 4.4 Hardware Specifications of development system

- Processor: Pentium IV or higher
- Processor speed: 1.6GHz
- RAM: 512 MB
- Disk Space: 250 MB or higher

## 4.5 Software Specifications of deployment system

Works with any android smartphone with V4.0 - 4.0.4 (Ice Cream Sandwich) or higher version of Android OS.

## 4.6 Hardware Specifications of deployment system

This application utilizes very less memory, approx. 8MB and requires 55MB of disk Space. Hence, it works with almost all current android Smartphones available in the market.

# Chapter 5 - System Design

## 5.1 System Design

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements [3]. Overall product architecture, the subsystems that compose the product, and the manner in which subsystems are allocated to processors are depicted using the System Design. UML is used to model system designs. Unified Modelling Language is a standard object-oriented analysis and design language. Use Case diagram and Class diagram, which are types of UML diagrams, of the application are shown below.

### 5.1.1 Use Case Diagram

A Use Case Diagram consists of set of elements and the relationships between them [4]. It depicts all the scenarios, regarding how our application interacts with users and other external systems to achieve the goals of application. The main components of a use case diagram include actors, use cases and their relationships. The use case is an external view of the system that represents some actions that the user performs in order to get a job done. Actors are the users who interact with the application.

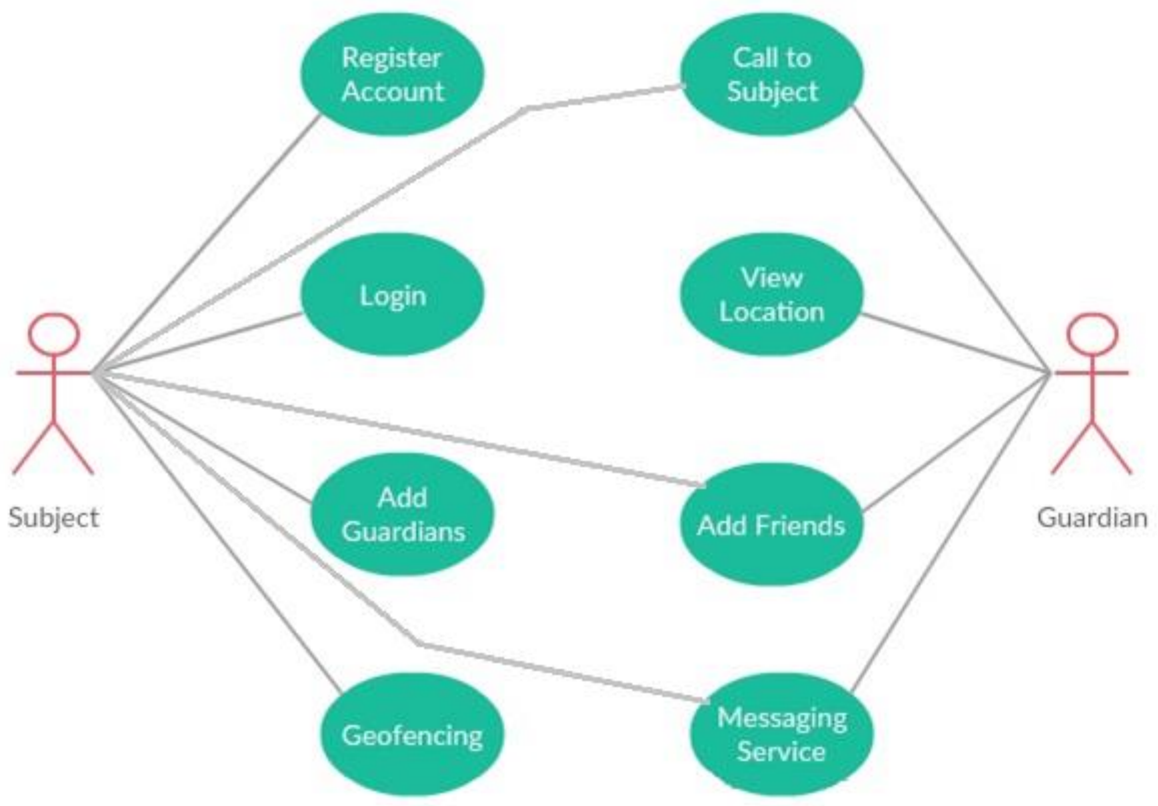


***Figure 2:*** *Use Case Diagram*

**Actors:**

The Actors of the system are Guardian and Subject (child)

**Use cases:**

I have identified a set of use cases based on the functionalities and goals of the application.

- **Register Account-** This use case denotes a set of actions required for Subject to register with the application.
- **Login-** This use case denotes a set of actions required for Subject to login into the application.
- **Add Guardians-** This use case denotes a set of actions required by Subject to add guardian contact to the application

- **Geo-Fencing-** This use case denotes a set of actions required by Subject to add a location of interest to the application and implement Geo-Fencing with a certain radius around it.
- **Call to Subject-** This use case denotes a set of actions required by Guardian to place a call to subject.
- **View Location-** This use case denotes a set of actions required by Guardian to locate subject on map after receiving his location details.
- **Add Friends-** This use case denotes a set of actions required by Guardian to add Subject’s friends contacts to the one click messaging service module of application
- **Messaging Service-** This use case denotes a set of actions required by Guardian to send a template message to subject’s friends with one click. Template message can be edited.

### 5.1.2 Class Diagram

A class diagram is a type of static structure diagram, which describes the structure of a system by depicting the system's classes, their attributes, methods, and the relationships among objects.

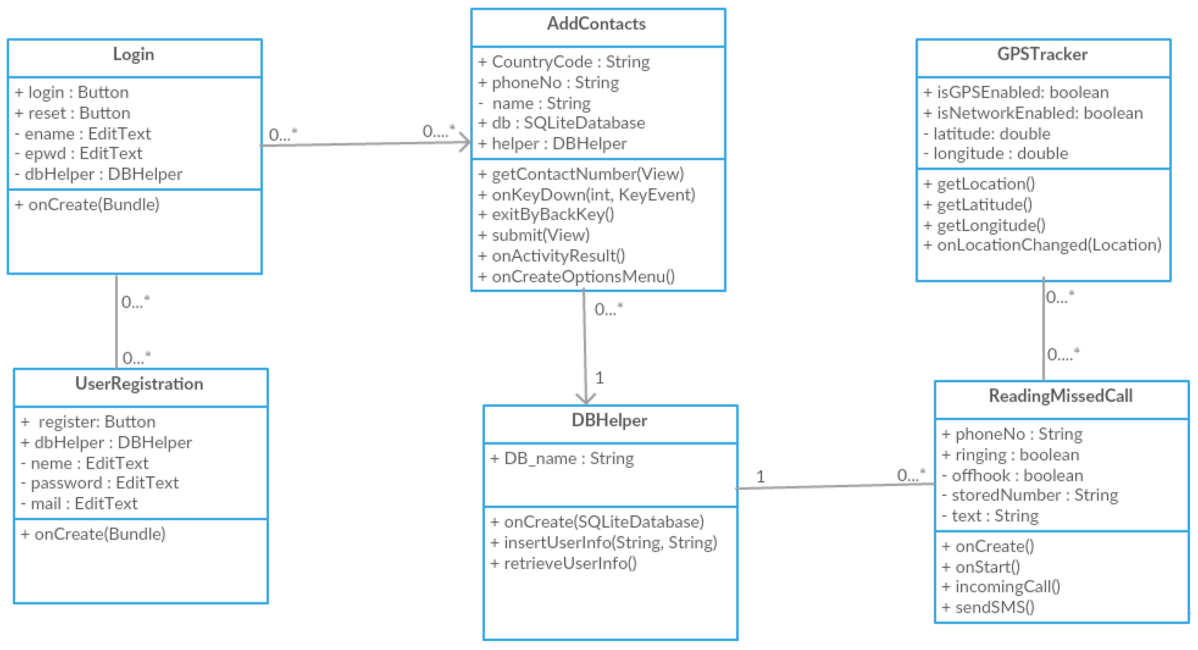


*Figure 3:* *Class Diagram*

The class diagram for Family Tracker contains Login, UserRegistration, AddContacts, GPSTracker, DBHelper, ReadingMissedCall classes

**Login:** Login class is responsible for verification of user login details. It prevents access to anyone other than the registered user to alter the contact details of the guardians.

**UserRegistration:** UserRegistration class is responsible for allowing users to create new account so that they can login into the application.

**AddContacts:** AddContacts class is responsible for adding guardians and friends to the application.

**GPSTracker:** It is responsible to get exact location of user to the application.

**ReadingMissedCall:** This class is responsible for reading missed call to the subject's phone. It then compares the missed call number to the guardian's list. If any match is found, the application sends out the location of the subject to list of guardians.

# Chapter 6 - Android Application Components

Android Application Components play a crucial role in developing android applications [5]. The main components of Android application are discussed below before we go into the implementation part of my application. The following are four main components that can be used within an Android application

## 6.1 Activities

Activity is an android component that provides a single screen with a user interface. Main Activity is the screen that will be displayed when the application is launched. Every time a new activity starts, the previous activity is stopped and the data is saved using a stack-based architecture.

## 6.2 Services

Services handle background processing associated with an application. Services performs operation without user interaction in the background In our application services are the major components helping in running the application all the time in the background.

## 6.3 Broadcast Receivers

Broadcast receiver responds to broadcast messages from various applications or the system like a system notification when the memory is full.

## 6.4 Content Providers

A content provider component allows our application to use data from other applications having necessary permissions. Generally, the data associated with any Android application is stored in SQLite database, which embedded in Android. We access system contacts data from our application

# Chapter 7 - Database Design

SQLite is an open source SQL database that stores data to a text file on a device [6]. Android comes in with built in SQLite database implementation. SQLite supports all the relational database features. There is no need to establish any kind of connections for accessing this database like JDBC, ODBC etc. Android.database.sqlite package helps us creating and managing our own databases with the help of the classes provided the package.

In this application development, we have used a SQLite database. In the database, five tables are present, one to store contacts of the guardians i.e., name of the guardian along with his phone number in one table and also one more table to store the details of subject friend's contacts along with a message used in messaging service module. Third table is used to store the user registration details- name, email, phone number, and password. When the user tries to login into the module, then the application checks with its user registration table. If the login details match to any one of the rows in the table, then he will be given access to module. Fourth table contains Geo-fencing details that are send to guardian in specific cases. There is one last table for storing the message that needs to be sent to subject's friends.

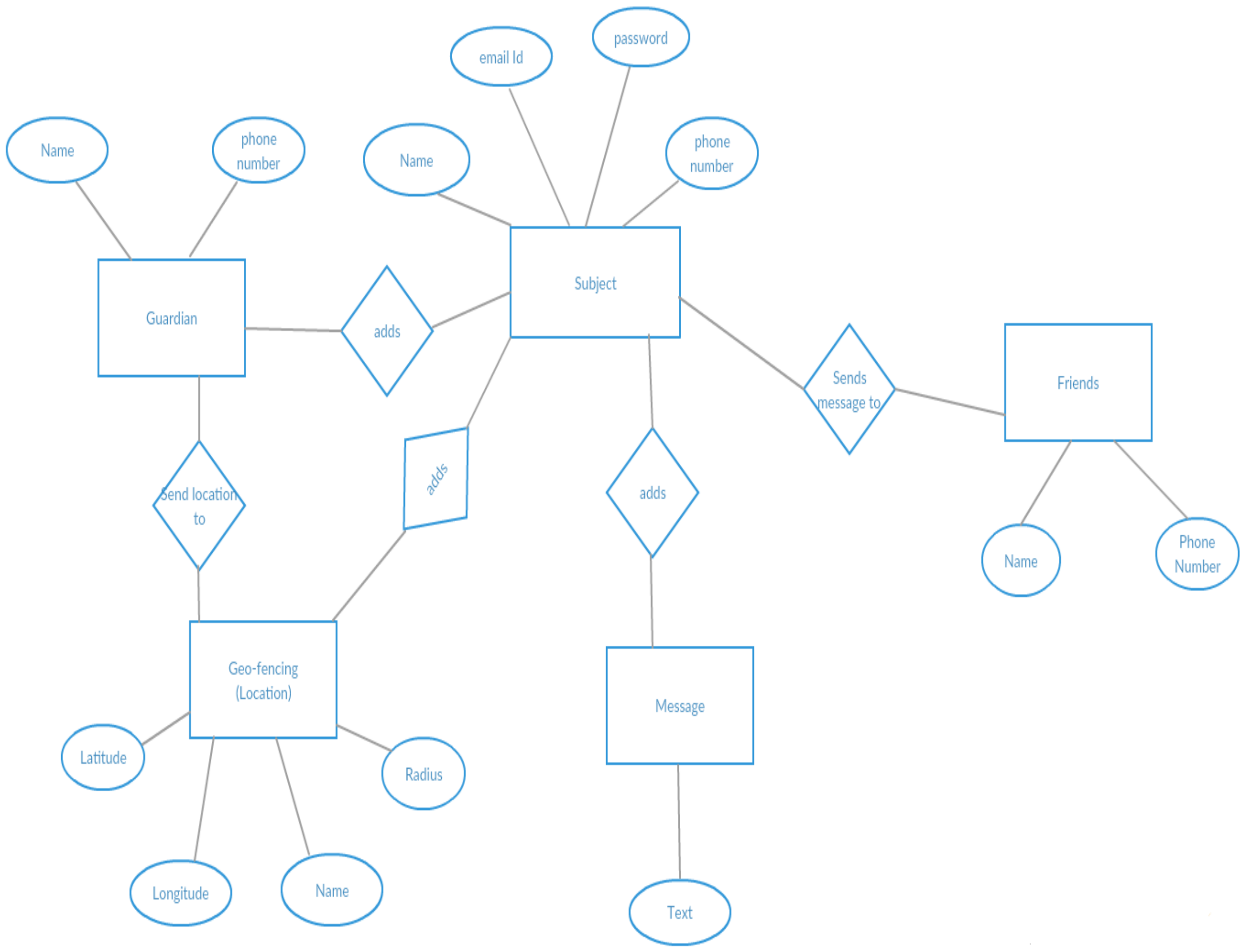


***Figure 4:*** *E-R Diagram*

# Chapter 8 - Implementation

Family Tracker is designed as an android application, which helps the user to know the whereabouts of a subject by installing the application in subject's smart phone. It provides much better way to track a location of person and to ensure that the subject is safe at all times. This application is developed keeping in mind the needs of the parents (guardians) who wants to keep an eye out for their children to ensure their safety. The application provides a flexible and easy to use environment in smart phones for the users to achieve their respective objective.

The modules implemented in this application are

- Subject/Child
- Parent/Guardian
- Location Tracker
- Geo-Fencing
- Messaging service

## 8.1 Subject/Child

Subject is the one with the application in his phone. Only subject has a username and password to log in to the application. He is the one who can add guardian's Contact (phone number) to the application. Any number of guardians can be added to application. Location of subject is send to the guardian.

## 8.2 Parent/Guardian

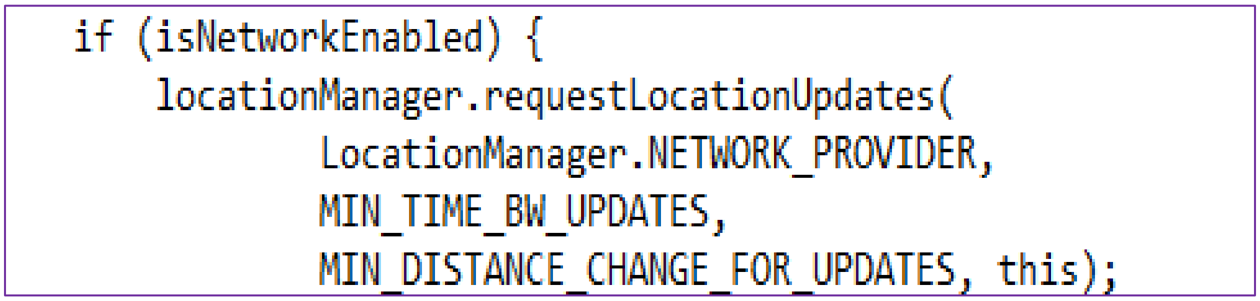
Parent is the one who gets the location of subject when the subject is unable to attend to the guardian's call on his phone. Application need not have to be installed in guardian's phone to get the location of subject. However, the guardians with access to this application on their phone are provided with a 'one-click enquiry through SMS' service.

## 8.3 Location Tracker

This module retrieves the exact location of the subject with latitude and longitude coordinates along with the corresponding street address and sends it to parent when it reads a missed call from guardian on subject’s phone.

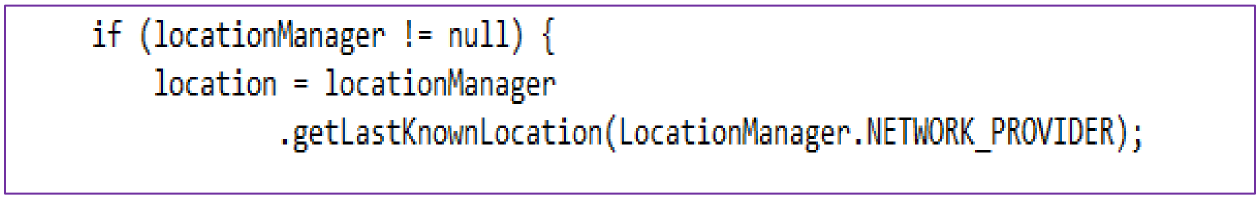
This location awareness application Android's Network Location Provider and GPS for retrieving the location of the subject [7]. All required classes for location tracking are provided with android.location package. I used both the GPS provider and Network provider to get accurate location of subject.

The LocationManager class provides access to the system location Services. It gives periodic location updates by calling requestLocationUpdates() method.



```
if (isNetworkEnabled) {
    locationManager.requestLocationUpdates(
            LocationManager.NETWORK_PROVIDER,
            MIN_TIME_BW_UPDATES,
            MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
```

Location can be obtained from getLastKnownLocation() method.

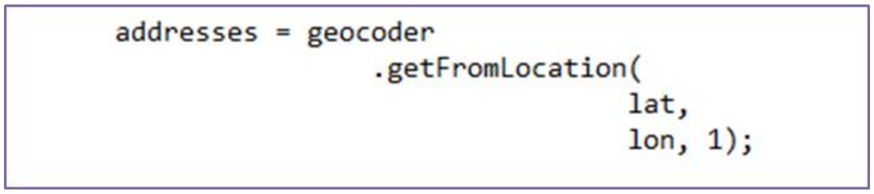


```
if (locationManager != null) {
    location = locationManager
            .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
```

The latitude and longitude of the location can be retrieved from this location using getLatitude() and getLongitude() methods.

**Reverse Geocoding**

This application main goal is to send out the subject’s location in the form of a street address rather than a location with latitude and longitude. For this purpose Reverse Geocoding is used. Geocoder class in the Android framework location APIs helps to convert a geographic location to an address. getFromLocation() method is used to get a street address to a geographic location.



```
        addresses = geocoder
                        .getFromLocation(
                                        lat,
                                        lon, 1);
```

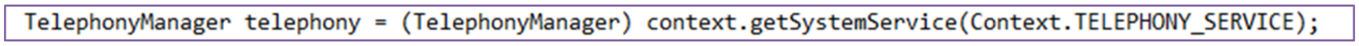
## 8.4 Geo-Fencing

Geo-Fencing is implemented in this application. Application asks for the latitude, longitude of the location of interest along with the radius, so that a perimeter around the location of interest is formed. The user based on his current requirement gives these details. When the subject is in this area, application sends out user given input location (place of interest) as the subject's location regardless of the actual subject location. When the subject is outside the area, then actual user location is sent.

## 8.5 Messaging Service

One click Messaging Service is provided by the application to help the guardian in conducting further enquiry regarding the whereabouts of his child by sending out a template message to child's friends with one click. This message template can be edited by guardian whenever he wants to. Guardian adds list of friends. This module is optional. Application should be present at guardian's end for this module to work.

For the implementation of this Service, we need access to the telephony services of the Smart phone. All Services related to call, SMS that are required in our application are provided in TelephonyManager class [8]. Reference to an instance of this class is retrieved through Context.getSystemService(Context.TELEPHONY_SERVICE).



```
TelephonyManager telephony = (TelephonyManager) context.getSystemService(Context.TELEPHONY_SERVICE);
```

The TelephonyManager class provides the states of a call, ringing or off hook, which we use in our application to detect whether the subject attended to guardian’s call or not.

- CALL_STATE_RINGING
- CALL_STATE_OFFHOOK

# Chapter 9 - Graphical User Interface

The application is developed to make sure that the user interactive screens are easily understandable and the navigation through the application is obvious. The screens of the application are discussed in detail in this chapter.

## 9.1 Splash Screen

The splash screen appears on the start of the application. This screen will display the different functionalities- Location tracker (adding guardians), one click messaging service (adding friends), Geo-Fencing (adding a location) available for the user. When the user clicks the button for required functionality, they will be navigated to the respective screens.

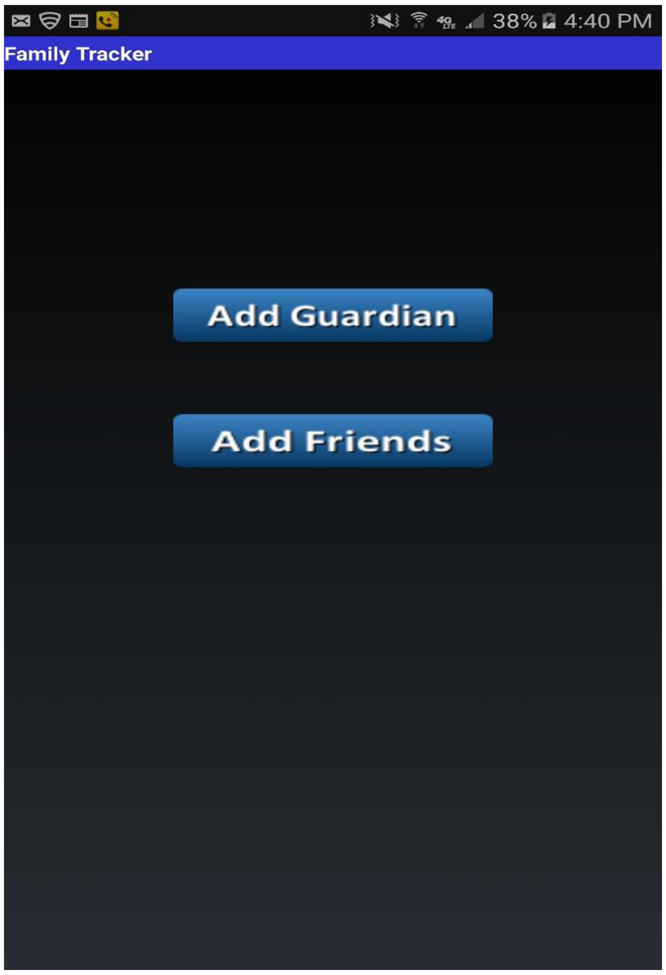


***Figure 5:*** *Splash screen*

## 9.2 Registration Screen

The subject registers with the application by providing necessary details- username, email Id, password and phone number.

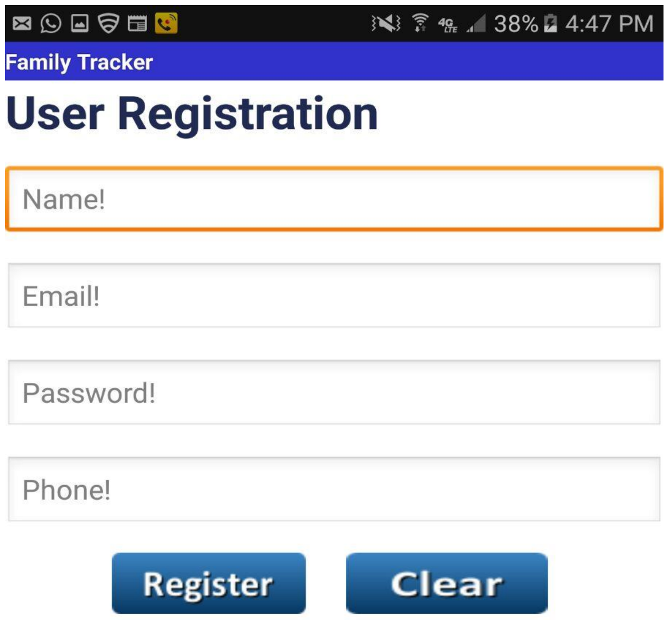
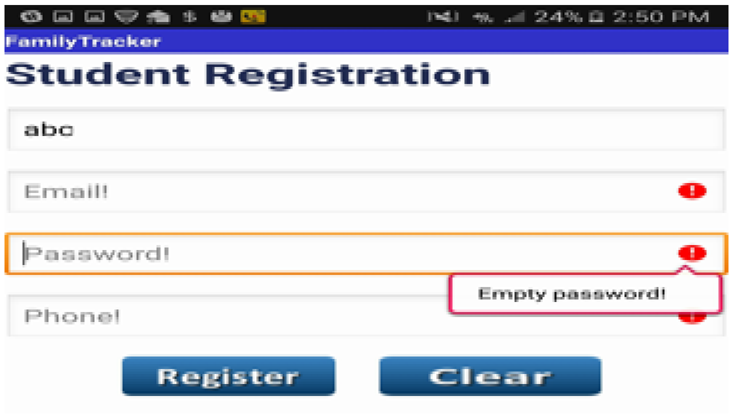
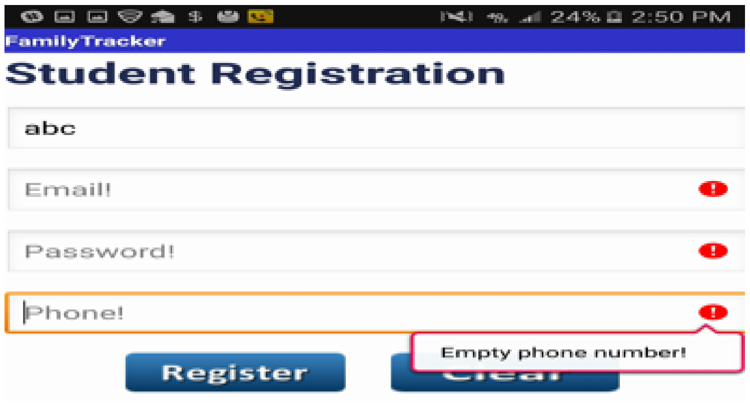
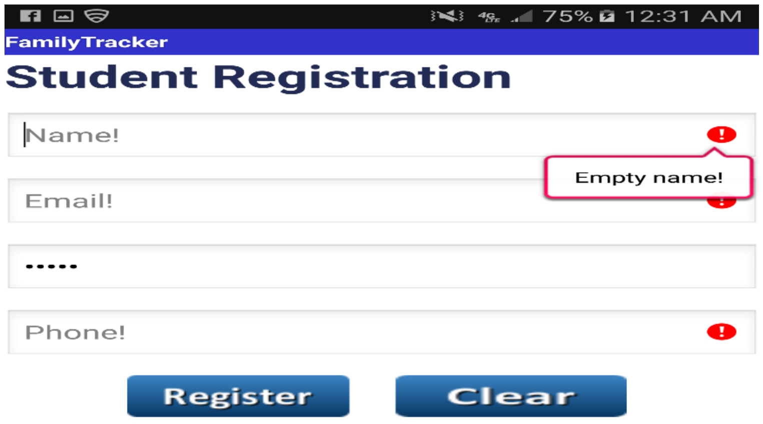


***Figure 6:*** *Registration Screen*

Validation is implemented for subject registration. User cannot register without a password or username or email Id or a phone number.







## 9.3 Login Screen

The subject logs into the application by providing his registered email Id and password.

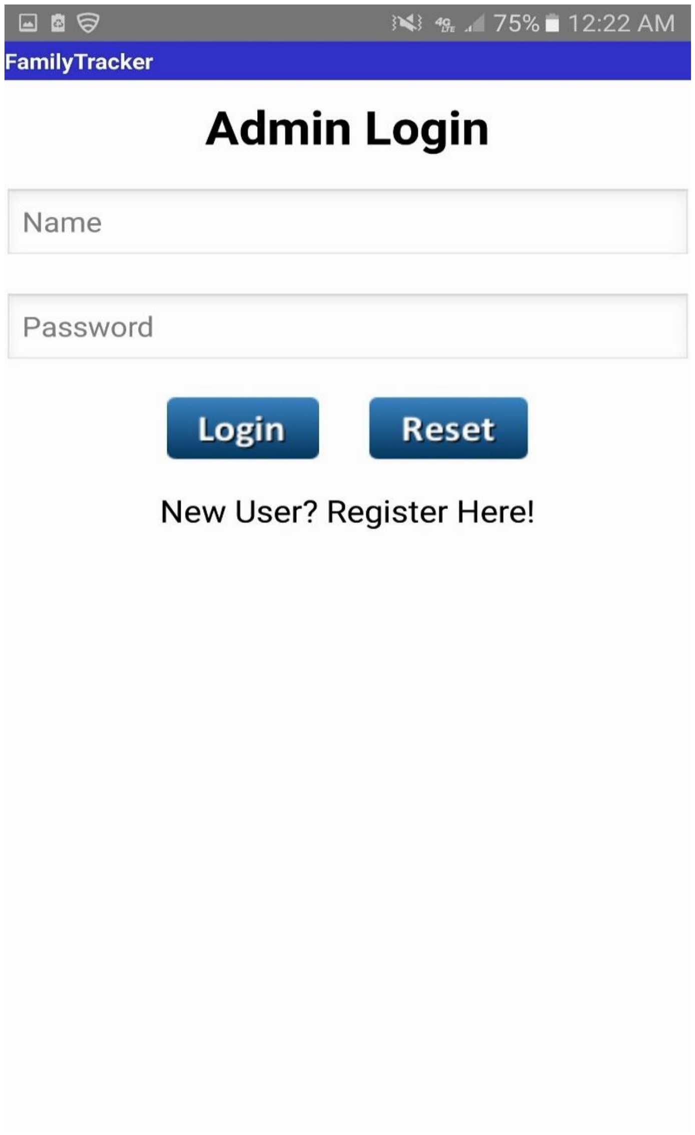


*Figure 7: Login screen*

## 9.4 Add guardian Screen

Subject adds the list of guardians to whom he wants to share his location.

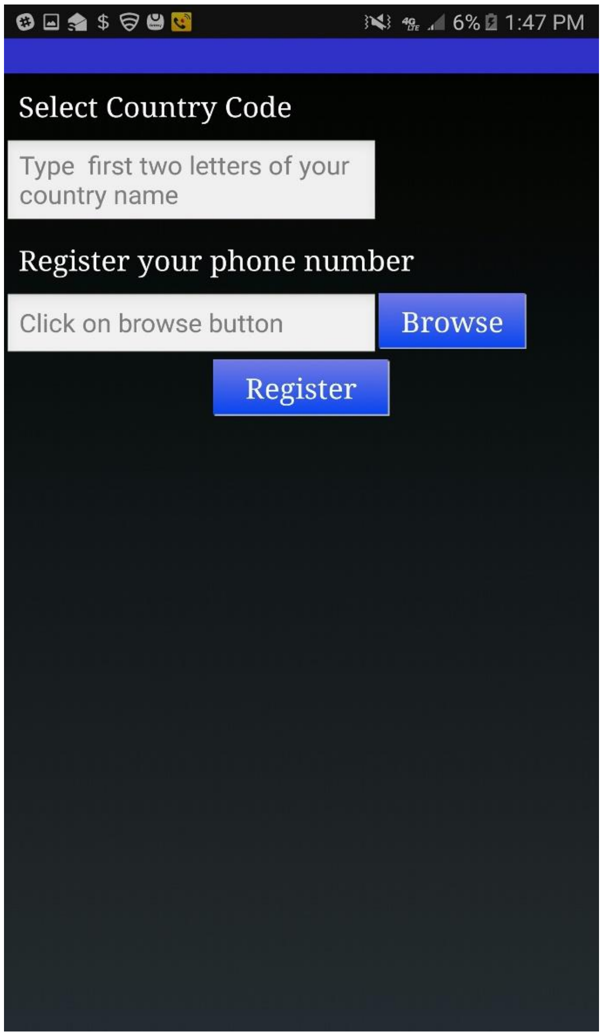

6% 1:47 PM
Select Country Code
Type first two letters of your country name
Register your phone number
Click on browse button
Browse
Register

***Figure 8:*** *Add Guardian Screen*

## 9.5 Add Friends Screen

Guardian adds three of subject's friends to whom he wants to send a message to enquire about subject whereabouts.

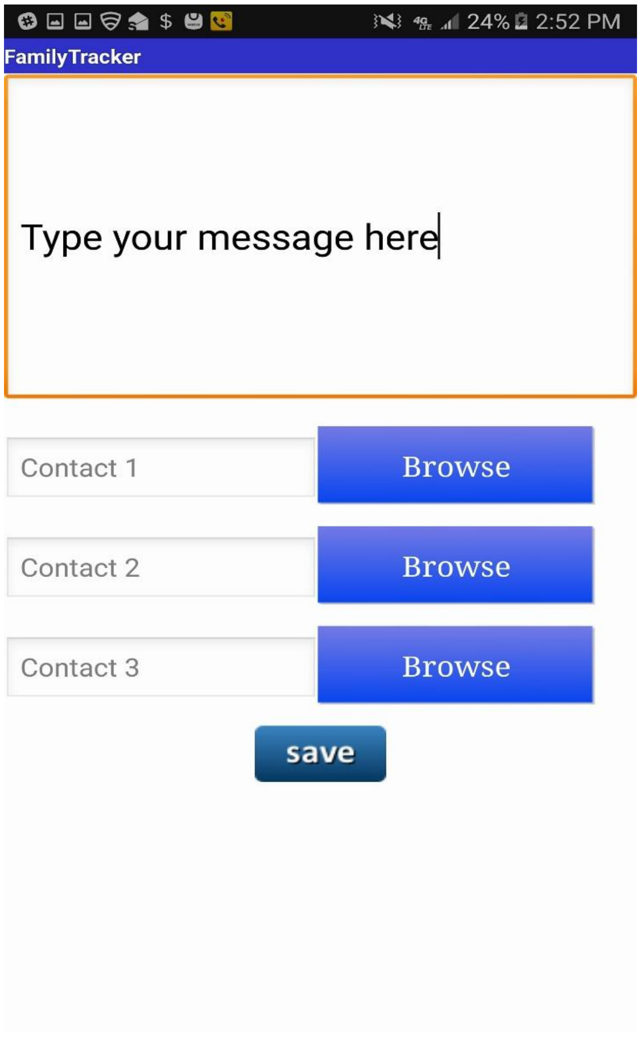


***Figure 9:*** *Add Friends Screen*

## 9.6 One Click Messaging Service

This screen is has options to navigate to add friends screen and also contains a send button, which on click sends out message to subject’s friends

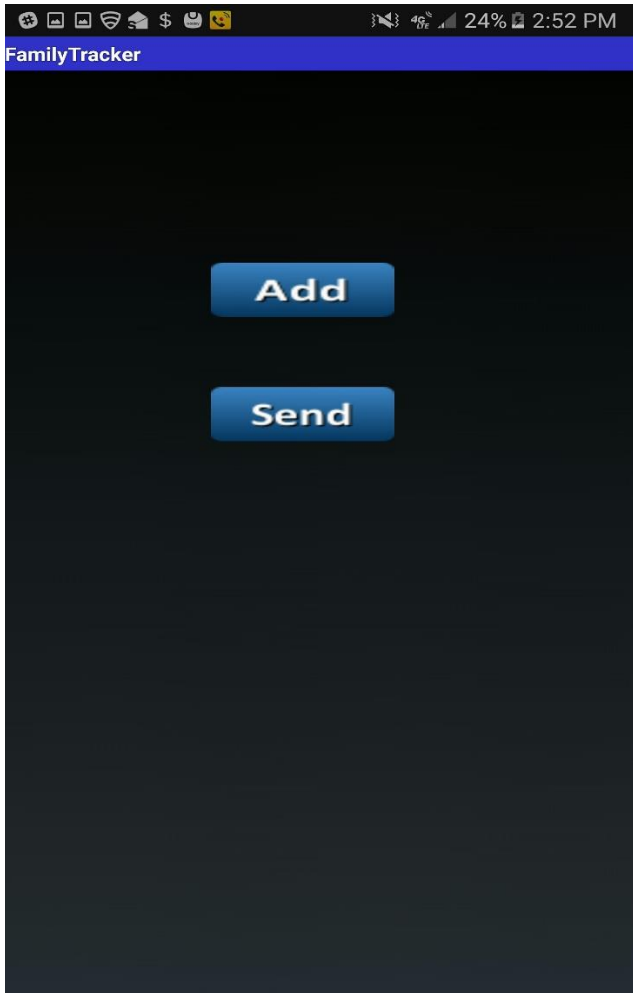
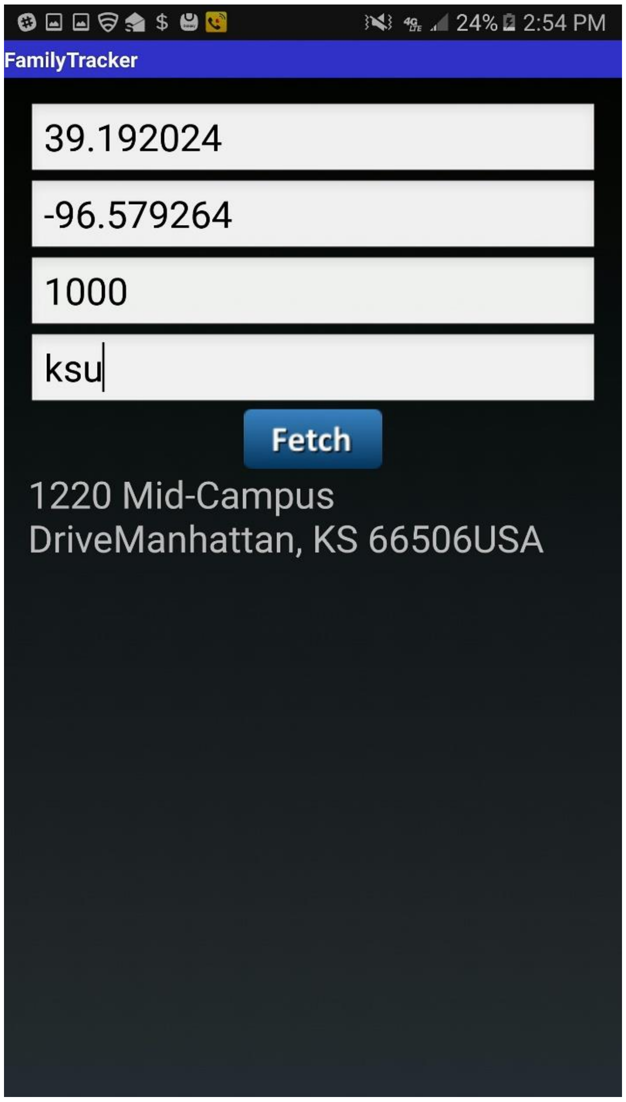


***Figure 10:*** *One Click Messaging Service*

## 9.7 Geo-Fencing Screen

User gives a location of interest to the application along with certain radius so that whenever the subject is inside the area around the location of interest, the guardian will receive message with the location of interest details but not the actual location of subject.



39.192024

-96.579264

1000

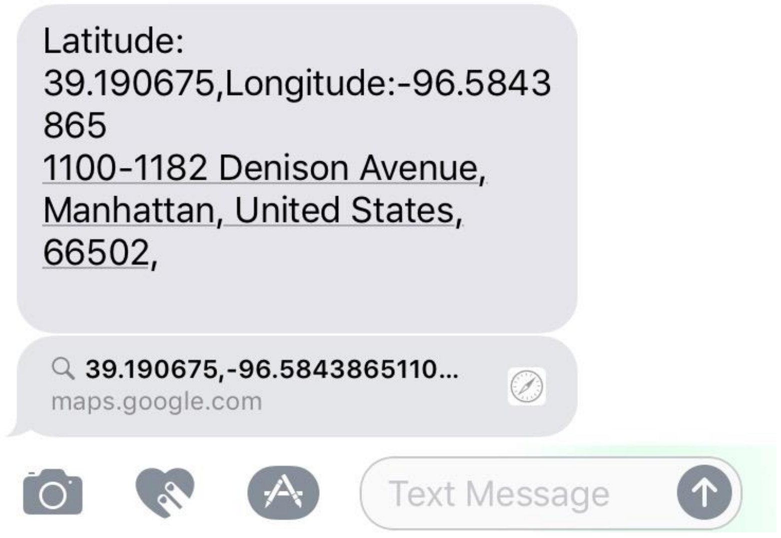
ksu

1220 Mid-Campus
DriveManhattan, KS 66506USA

***Figure 11:*** *Geo-fencing Screen*

## 9.8 Results (Output)

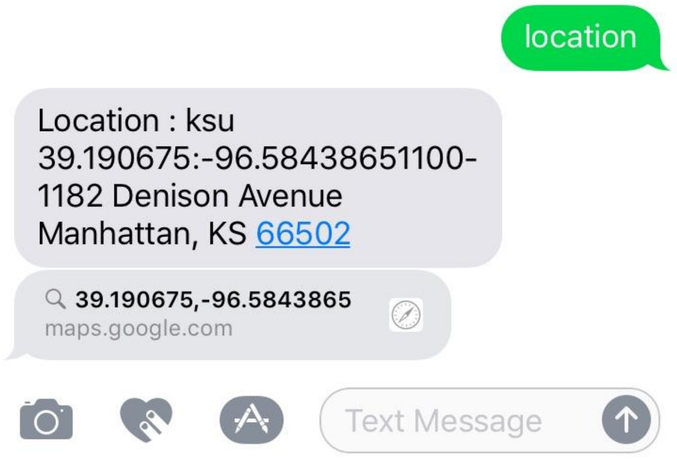
Subject location with geo-fencing send to guardian's phone through SMS. The cases where subject is outside the geo-fence and when subject is inside geo-fence is shown below. When subject is inside geo-fence 'Location: Name' is the addition field that will be sent to guardian.



Latitude:
39.190675,Longitude:-96.5843
865
1100-1182 Denison Avenue,
Manhattan, United States,
66502,

39.190675,-96.5843865110...
maps.google.com

***Figure 12:*** *Subject outside geo-fence*



location

Location : ksu
39.190675:-96.58438651100-
1182 Denison Avenue
Manhattan, KS 66502

39.190675,-96.5843865
maps.google.com

***Figure 13:*** *Subject inside geo-fence*

# Chapter 10 - Testing

Software Testing is a process of executing the application with an intent to find any software bugs [9]. It is used to check whether the application met its expectations and all the functionalities of the application is working. The final goal of testing is to check whether the application is behaving in the way it is supposed to under specified conditions. All aspects of the code are examined to check the quality of application. The primary purpose of testing is to detect software failures so that defects may be uncovered and corrected. The test cases are designed in such way that scope of finding the bugs is maximum.

## 10.1 Testing Levels

There are various testing levels based on the specificity of test.

- ***Unit testing***: Unit testing refers to tests conducted on a particular section of code in order to verify the functionality of that piece of code. This is done at the function level.
- ***Integration Testing***: Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Its primary purpose is to expose the defects associated with the interfacing of modules.
- ***System Testing***: System testing tests a completely integrated system to verify that the system meets its requirements.
- ***Acceptance testing***: Acceptance testing tests the readiness of application, satisfying all requirements.

## 10.2 Test Cases

A test case is a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

I have designed and executed a few test cases to check if the application meets the functional requirements. I have tested the application on Samsung note Smart phone, whose results are summarized as below.

**Test Objectives:** Navigation from Splash screen to Login screen

| **TEST CONDITION** | **INPUT SPECIFICATION** | **OUTPUT SPECIFICATION** | **PASS/FAIL** |
|---|---|---|---|
| The user is currently on the Splash screen | User clicks on Add Guardian button | Directs to Login page | PASS |

***Table 1:*** *Test case for navigation from Splash screen to Login screen*

**Test Objectives:** Navigation from Login Screen to Add Guardians screen

| **TEST CONDITION** | **INPUT SPECIFICATION** | **OUTPUT SPECIFICATION** | **PASS/FAIL** |
|---|---|---|---|
| The user is currently on the Login screen | User enters credentials and clicks on login button | Directs to Add guardians screen | PASS |

***Table 2:*** *Test case for navigation from Login Screen to Add Guardians screen*

**Test Objectives:** Navigation from Login screen to Registration screen

| **TEST CONDITION** | **INPUT SPECIFICATION** | **OUTPUT SPECIFICATION** | **PASS/FAIL** |
|---|---|---|---|
| The user is currently on the Login screen | User clicks on New User link | Directs to Registration page | PASS |

***Table 3:*** *Test case for navigation from Login screen to Registration screen*

**Test Objectives:** Navigation from Registration screen to Login screen

| TEST CONDITION | INPUT SPECIFICATION | OUTPUT SPECIFICATION | PASS/FAIL |
|---|---|---|---|
| The user is currently on the Register screen | User enters all details and clicks on Register button | Directs to Login screen | PASS |

***Table 4:*** *Test case for navigation from Registration screen to Login screen*

**Test Objectives:** Navigation from Splash screen to Add Friends screen

| TEST CONDITION | INPUT SPECIFICATION | OUTPUT SPECIFICATION | PASS/FAIL |
|---|---|---|---|
| The user is currently on the Splash screen | User clicks on Add friends button | Directs to Add friends screen | PASS |

***Table 5:*** *Test case for navigation from Splash screen to Add Friends screen*

**Test Objectives:** Navigation from Splash screen to Track Location screen

| TEST CONDITION | INPUT SPECIFICATION | OUTPUT SPECIFICATION | PASS/FAIL |
|---|---|---|---|
| The user is currently on the Splash screen | User clicks on Track Location button | Directs to Track Location screen | PASS |

***Table 6:*** *Test case for navigation from Splash screen to Track Location screen*

**Test Objectives:** User checks for functionality of Add Guardians Screen

| **TEST CONDITION** | **INPUT SPECIFICATION** | **OUTPUT SPECIFICATION** | **PASS/FAIL** |
|---|---|---|---|
| The user is currently on the Add Contact page | User selects the country from list provided and clicks on browse button to select a contact from phone contacts and clicks on 'Add Contact'. | The application saves the contact and displays on Guardians page. | PASS |

***Table 7:*** *Test case for checking the list of contacts added*

**Test Objectives:** User checks for functionality of One Click Messaging Service

| **TEST CONDITION** | **INPUT SPECIFICATION** | **OUTPUT SPECIFICATION** | **PASS/FAIL** |
|---|---|---|---|
| The user is currently on the One Click Messaging Service Page | User clicks on 'Send' button | All contacts stored Add Friends receives message from the guardian | PASS |

***Table 8:*** *Test for One Click Messaging Service*

# Chapter 11 - Performance Profiling

Performance of the application can be determined based on the application's responsiveness under all kinds of load. Android Studio and the mobile device I have used provide Profiling tools to record and visualize the rendering, compute memory and battery performance of the application are provided by the android smart phone and android studio that are used by me to develop and launch my application.

## 11.1 Rendering Analysis Tools

Developer options has profiling tools, which are used to visualize the rendering of my application. I have tested this application on Samsung Galaxy Note 5 having the specifications as given below.

### 11.1.1 System Configuration

| | |
|---|---|
| Operating System | Android OS, v6.0 (Marshmallow) |
| CPU | Octa-core (4x2.1 GHz Cortex-A57 & 4x1.5 GHz Cortex-A53) |
| GPU | Mali-T760MP8 |
| Internal | 64 GB, 4 GB RAM |

***Table 9:*** *System Configuration - Android Phone*

### 11.1.2 Debug GPU Overdraw

An app may draw the same pixel more than once within a single frame, an event called overdraw. The Debug GPU Overdraw tool uses color-coding to show the number of times your app draws each pixel on the screen [10]. The higher this count, the more likely it is that overdraw affects your app's performance. It shows how to visualize overdraw on your mobile device by color-coding interface elements based on how often they are drawn underneath. It helps by showing where an app might be doing more rendering work than necessary and guiding where you might be able to reduce rendering overhead.

I used the developer options available in my device to turn on Debug GPU overdraw option. The colors on the screen hints the amount of overdraw on the screen for each pixel.

- Color - Overdraw
- True Color - No overdraw
- Blue - Overdrawn once
- Green - Overdrawn twice
- Pink - Overdrawn thrice
- Red - Overdrawn four or more times

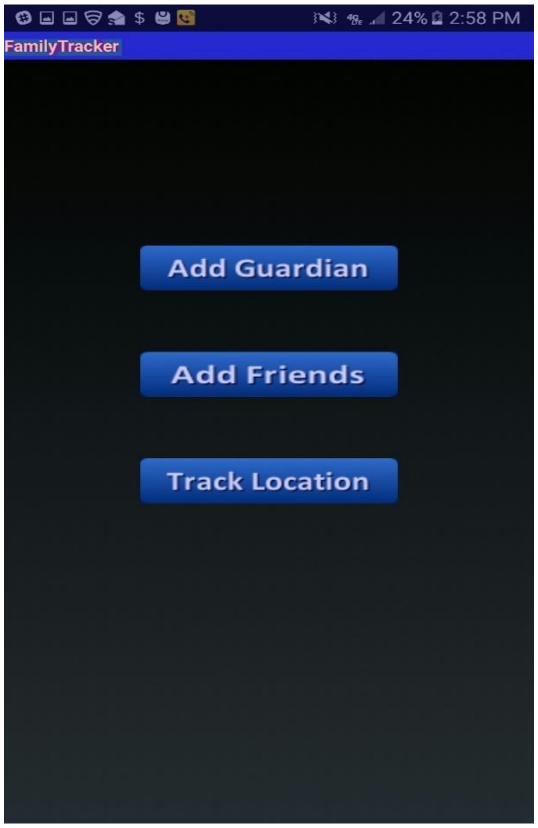


***Figure 14:*** *Debug GPU Overdraw for Splash Screen*

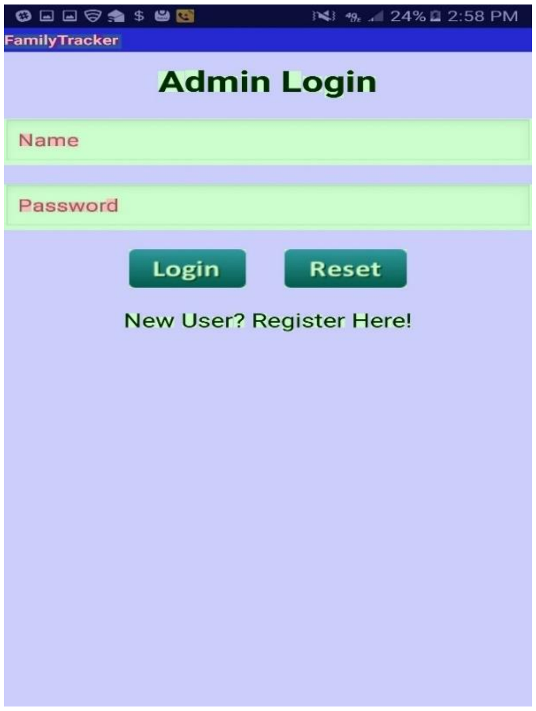


***Figure 15:*** *Debug GPU Overdraw for Login page*

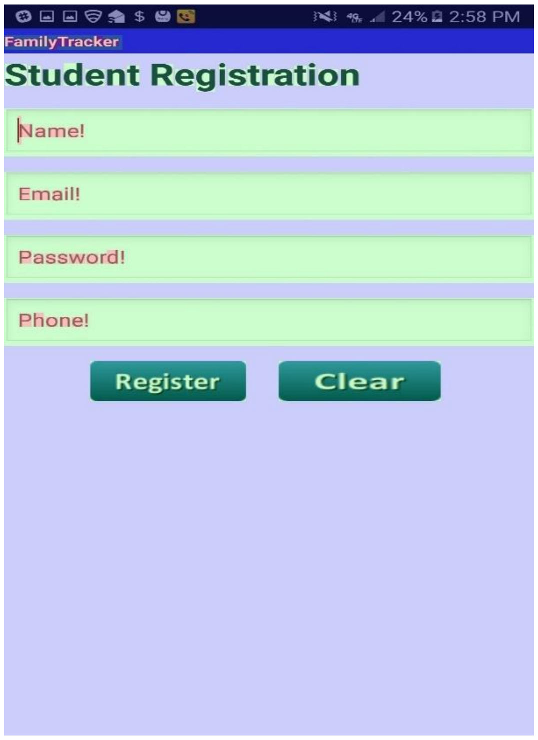


***Figure 16:*** *Debug GPU Overdraw for Registration page*

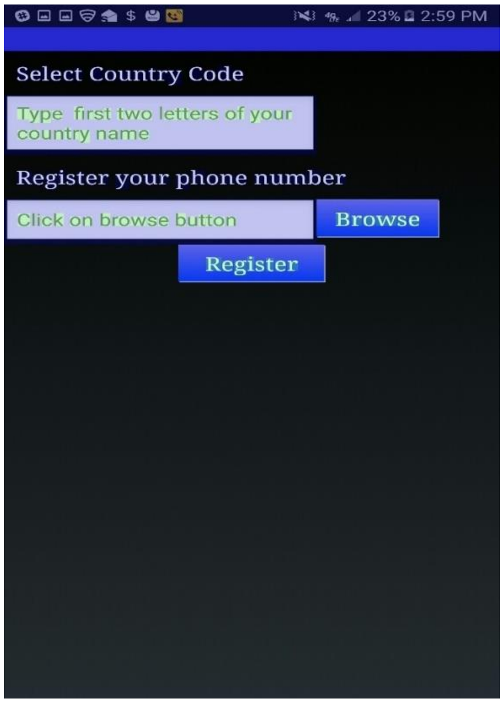

23% 2:59 PM
Select Country Code
Type first two letters of your country name
Register your phone number
Click on browse button
Browse
Register

***Figure 17:*** *Debug GPU Overdraw for Add Guardians page*

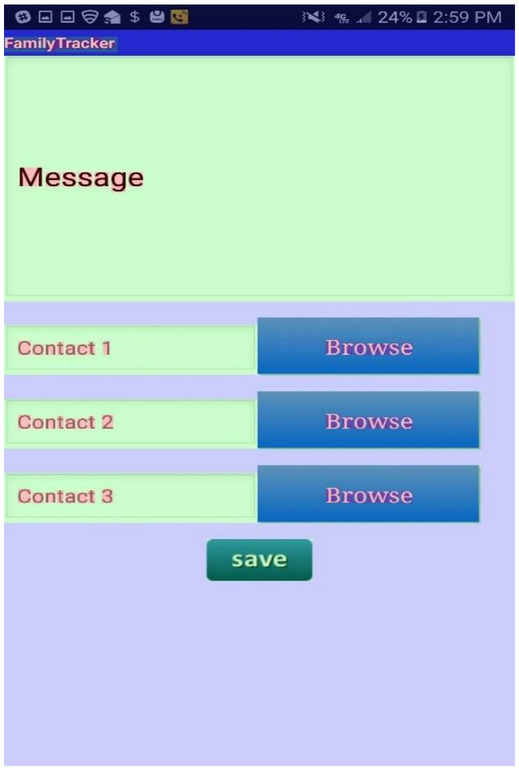


***Figure 18:*** *Debug GPU Overdraw for Add Friends page*

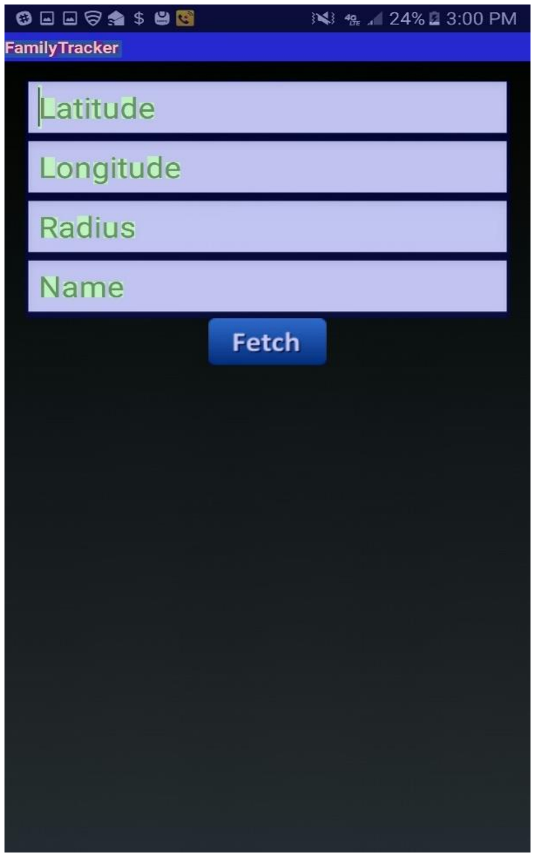


***Figure 19:*** *Debug GPU Overdraw for Geo-Fencing page*

As we can see, the Debug GPU Overdraw for several screens of this application point out that there is no high level of overdraw associated with any screen, as none of the screens are focused 'red'. Most of the screens are overdrawn twice or thrice. The screens, which connect with database, are highly effected (overdrawn thrice) and needs this rendering overhead to be reduced.

### 11.1.3 Profiling GPU Rendering

Profile GPU Rendering gives you a quick visual representation of how much time it takes to render the frames of a UI window relative to the 16-ms-per-frame benchmark [11]. It helps in quickly seeing how a UI window performs against the 16-ms-per-frame target and identifying whether any part of the rendering pipeline stands out in using processing time. It also helps in identifying spikes in frame rendering time associated with user or program actions.

For each visible application, the tool displays a graph. The horizontal axis shows time elapsing, and the vertical axis time per frame in milliseconds. The vertical bars show up from left to right on screen over time giving graphing frame performance. Each vertical bar represents one frame of rendering. The taller the bar, the longer it took to render. The graph has colored sections representing the phase of the rendering.

- The **green horizontal line** represents 16 milliseconds. Any time a bar pushes above this line; there may be pauses in the animations.
- The **Orange** section of the bar represents the time CPU is waiting for GPU to finish its work.
- The **blue** section of the bar represents the time used to create and update the View's display lists.
- The **purple** section of the bar represents the time spent transferring resources to the render thread.
- The **red** section of the bar represents the time spent by Android's 2D renderer issuing commands to OpenGL to draw and redraw display lists.

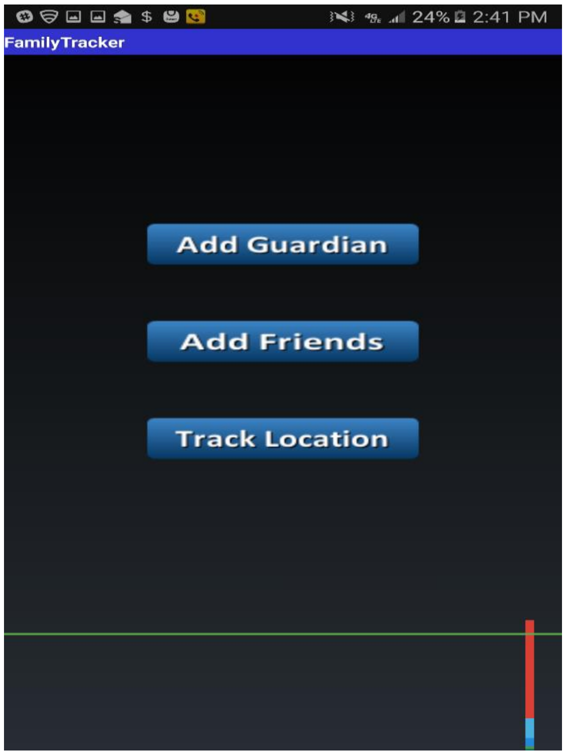


*Figure 20:* *Profile GPU Rendering for Splash Screen*

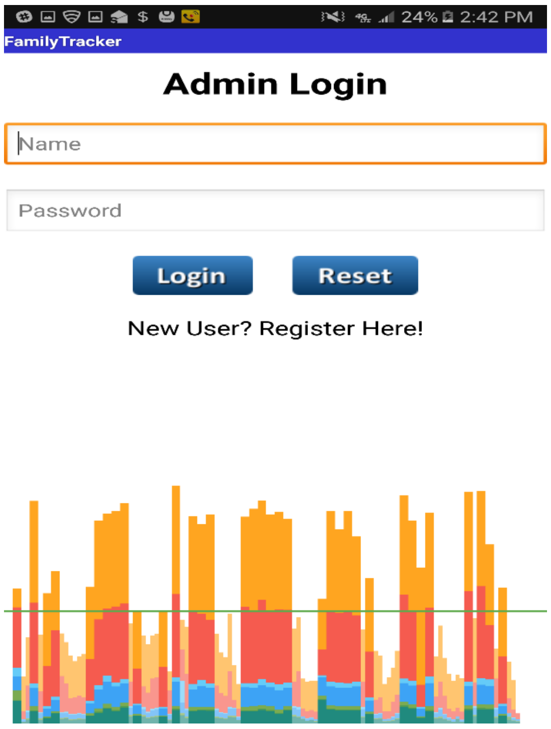


*Figure 21:* *Profile GPU Rendering for Login page*

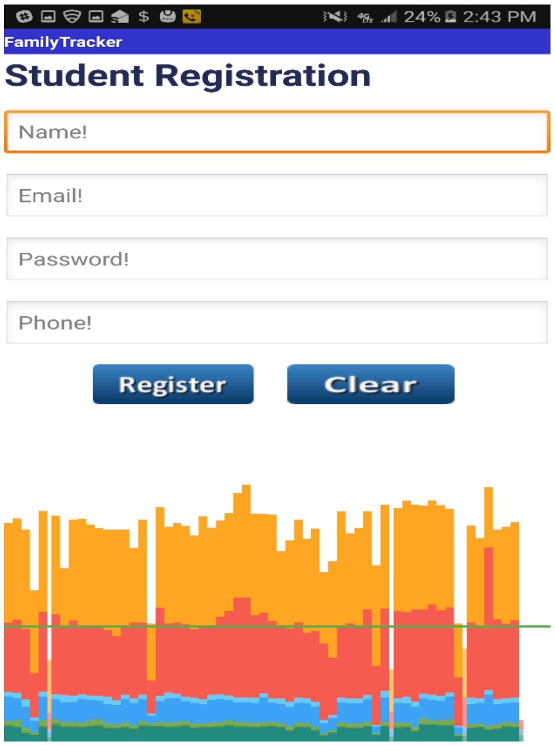


***Figure 22:*** *Profile GPU Rendering for Registration page*

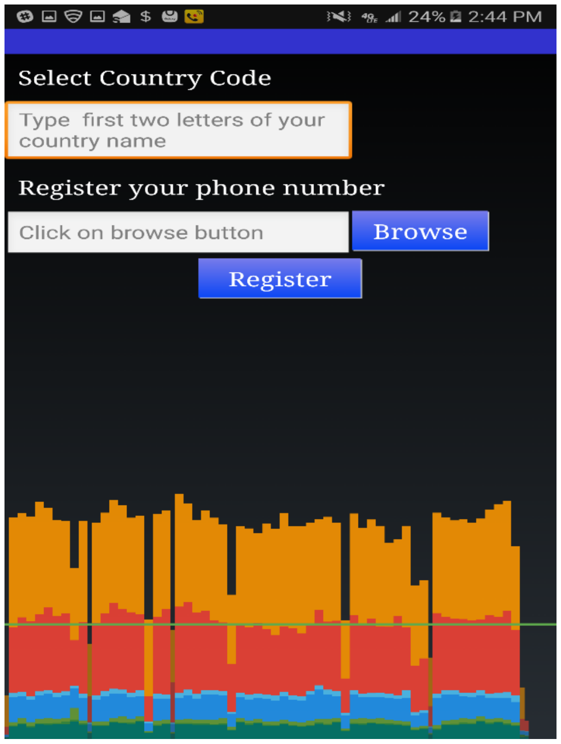

24% 2:44 PM
Select Country Code
Type first two letters of your country name
Register your phone number
Click on browse button
Browse
Register

***Figure 23:*** *Profile GPU Rendering for Add Guardians*

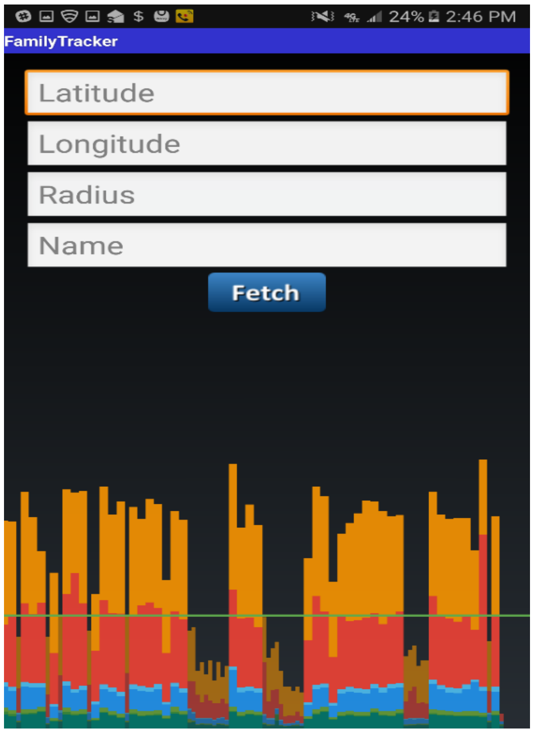


***Figure 24:*** *Profile GPU Rendering for Geo-Fencing*

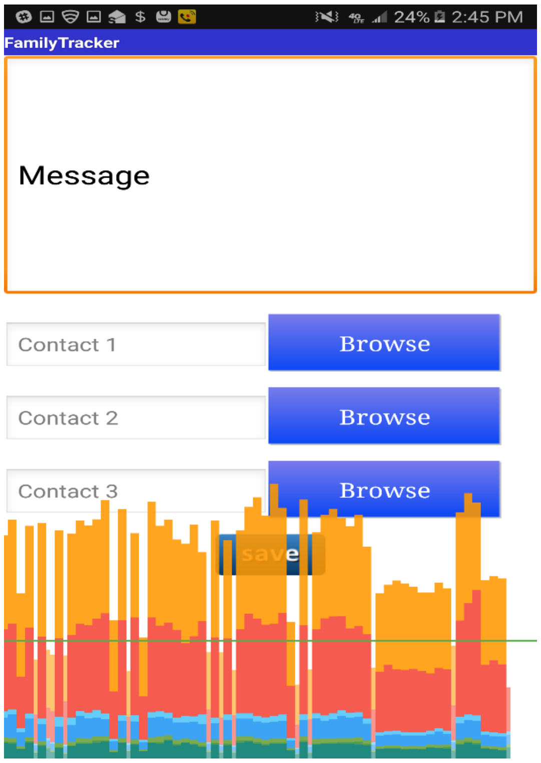


***Figure 25:*** *Profile GPU Rendering for Add Friends*

As we can see, the main screen does not have much graphics associated with it and hence the rendering is just above the 16 milliseconds mark. As we go further into other application screens, there is some increase in the graphic content compared to home screen. This results in increase of time waited by CPU for GPU to complete its work. When compared with other similar applications, this application has very low level of graphics for all screens. The operations such as typing into the text editors of each screen is causing the increase in the rendering time.

# Chapter 12 - Conclusion and Future Work

## 12.1 Conclusion

Though there are many applications which provides you the functionalities provided by this application, there is no other application which provides both the functionalities of Location tacking and Geo-Fencing. There are also very few applications that provides user, the location of his subject through SMS. This Android application provides the flexibility of installing this application on subject’s Android powered smart phone but not the user. This can be really helpful for user who doesn’t own a smart phone, as the application need not to be installed at the user end. User gets the location of the subject though an SMS when the subject is unable to attend to the user’s call. Geo-Fencing is also implemented in this application that helps the user know whether the subject is at the place of interest. There is also one more module ‘user enquiry’, which sends a predefined message to list of friends of subject for further enquiry of subject’s whereabouts, which is optional based on the availability of an android smart phone at user’s end.

## 12.2 Future Work

The Family Tracker application can be enhanced coming with a better method of plotting an area of interest in Geo-Fencing module instead of having a circular area around the location. Maps can be integrated into the application making easy for the user to select the location on map as a place of interest rather than typing in latitude and longitude details. Keeping track of all the movements of the subject in a particular interval can also be implemented in future work.

# References

[1] “An overview of the Android Architecture”
http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture [Feb. 1, 2017]

[2] “Platform Architecture”
https://developer.android.com/guide/platform/index.html [Feb. 10, 2017]

[3] “Systems design”
https://en.wikipedia.org/wiki/Systems_design [Feb. 17, 2017]

[4] “UML - Standard Diagrams”
https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm [Feb. 18, 2017]

[5] “Android - Application Components”
http://www.tutorialspoint.com/android/android_application_components.htm [Feb. 25, 2017]

[6] “Android - SQLite Database”
http://www.tutorialspoint.com/android/android_sqlite_database.htm [March 4, 2017]

[7] “Location Strategies”
https://developer.android.com/guide/topics/location/strategies.html [March 5, 2017]

[8] “TelephonyManager”
https://developer.android.com/reference/android/telephony/TelephonyManager.html [March 9, 2017]

[9] “Software Testing”
https://en.wikipedia.org/wiki/Software_testing [March 20, 2017]

[10] “Debug GPU Overdraw Walkthrough”
https://developer.android.com/studio/profile/dev-options-overdraw.html [March 29, 2017]

[11] “Profile GPU Rendering Walkthrough”
https://developer.android.com/studio/profile/dev-options-rendering.html [April 2, 2017]