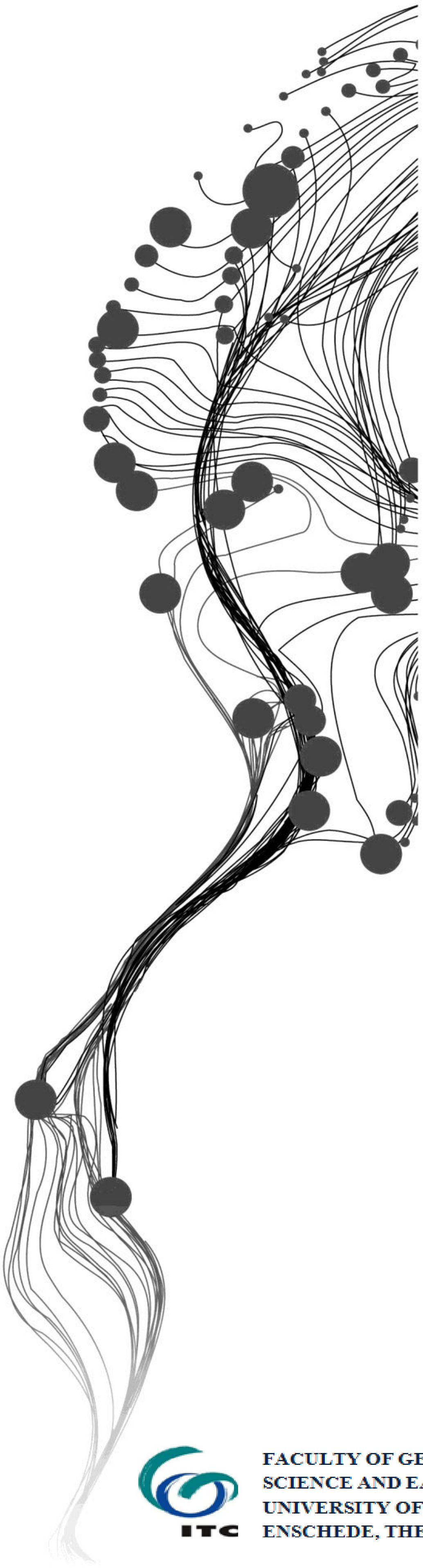
An Android application to support flash flood disaster response management in India

TANYA TEJASSVI
March, 2014

IIRS SUPERVISOR
Dr. Harish Karnatak

ITC SUPERVISOR
Dr. Ir. Rolf A. de By



An Android application to support flash flood disaster response management in India

TANYA TEJASSVI
Enschede, The Netherlands [March, 2014]

Dedicated to Papa

ABSTRACT

The scope of information technology is leveraged in every aspect of society. Spatial data and related information is a major requirement in disaster management. During rescue operations, availability of real-time information on a portable device would prove useful. The aim of this research was to develop and test a GIS-based mobile application on the Android platform with disaster specific software modules. This mobile application is expected to serve disaster management teams during the rescue operations for better co-ordination and information exchange. The rescue and relief management efforts for June 2013, Uttarakhand disaster have been reviewed and studied for this research. A formal interview and discussions with NDRF officials, helped form an understanding, envisage requirements for the application. The application RescueApp has been developed after careful evaluation of requirements and case scenarios.

The application exhibits Emergency/Distress Call, Reporting System, Disaster Alerts and Geo-visualization as its key features. The application allows field data reporting, sending geo-location SMS, viewing and retrieving weather and location information on the mobile device. The application has been tested for usability, time consumption and accuracy in different field and network availability conditions. An attempt, through this research, was also made to realise the utility of the application in limited availability or absence of communication network.

***Keywords*: Android, disaster management, API, Field data reporting, response, relief**

ACKNOWLEDGEMENTS

"*Surround yourself with dreamers and the doers, the believers and the thinkers, but most of all surround yourself with those who see greatness within you, even when you don't see it yourself.*" - Edmund Lee

I express my sincere gratitude to both my supervisors for their patience and perennial support and guiding me through my MSc. Thesis. I would like to thank Dr. Ir. Rolf A. de By, my ITC supervisor, for his visionary approach towards problem definition of this research and critical review of thesis. Dr. Harish Karnatak, my IIRS supervisor, with his profound technical knowledge and field experience, helped me frame constructive ideas and in successful development of the application. I am thankful to ITC faculty, Bas Retsios for his help and guidance on setup and implementation of development environment.

I am very grateful to Shri R. K. Srivastava, Deputy Commandant, NDRF, New Delhi for sharing his knowledge and experience about rescue operations. I appreciate his support and motivation, which gave a meaningful direction to my research.

This joint course was a wonderful platform to meet people across the globe and experience knowledge sharing in a multi-cultural environment. I would like to thank all the faculty members at IIRS, Dehradun and at ITC, Enschede. I would like to thank Dr. Y.V.N Krishnamurthy, Director IIRS for providing state of the art infrastructure and facilities, which helped nurture and enhance our knowledge. Mr.P.L.N Raju has been a great source of motivation and Dr. Srivastav has consistently supported me and helped channelize my efforts into the right direction.

I am thankful to my family for being my pillars of strength. Dad has stood by me and supported all my decisions while I owe each success to my Mom's prayers for me; My soul mate Amrita, for always keeping my spirits high, encouraging me to bring out the best in me and Jiju for motivating me to always aim high and strive for perfection. And Ayush, for being by my side always, to cheer me up. You are my Guardian Angel. I truly am blessed to have such a wonderful family.

Worth a special mention is Vishnu Nandan, an alumnus of IIRS who has been a friend, philosopher and guide throughout my research. He has helped me successfully propose the research topic and take it to completion. Thanks a lot for being there.

My friends; Abhinandan, Sumit, Waibhav, Komal, Nikhil, Simrat, Ipi , Jayesh, Amol and Varun Garg have always been so encouraging and motivating. I owe a big treat to my friend Akash who has been very helpful throughout the development phase of my project. From IIRS, Special thanks to Akhil for always being there, spreading happiness, being available for my App testing and a consistent partner in burning the midnight oil in computer lab. I was lucky to have wonderful IIRS juniors who have been instrumental in keeping my positivity intact and my spirits high even when things were not looking too bright; Sakshi, Shradha, Garima, Vikrant and Surya - Thank you so much for just being there and being yourselves.

And above all I thank God, for giving me the strength in having embarked and completed this journey beautifully so. Even through the dark corners, he left a gleaming ray of hope each time.

P.S. I heartily thank my father for financing this course and especially the cherished Euro-Trip

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Background

For a long time disaster management in India has been reactive and relief-centric. In the wake of serious disasters in the past decade, a new approach has been adopted by India to give disaster management its due importance. The emphasis now is not just on relief but also on prevention, mitigation and preparedness. Such efforts are expected to help minimize loss to lives and property. On 23 Dec, 2005, the Disaster Management Act was enacted by the Government of India and led to the formation of National Disaster Management Authority (NDMA) headed by the Prime Minister and State Disaster Management Authorities (SDMAs), which are headed by the State Chief Ministers. The aim is to follow an integrated and holistic approach towards disaster management.

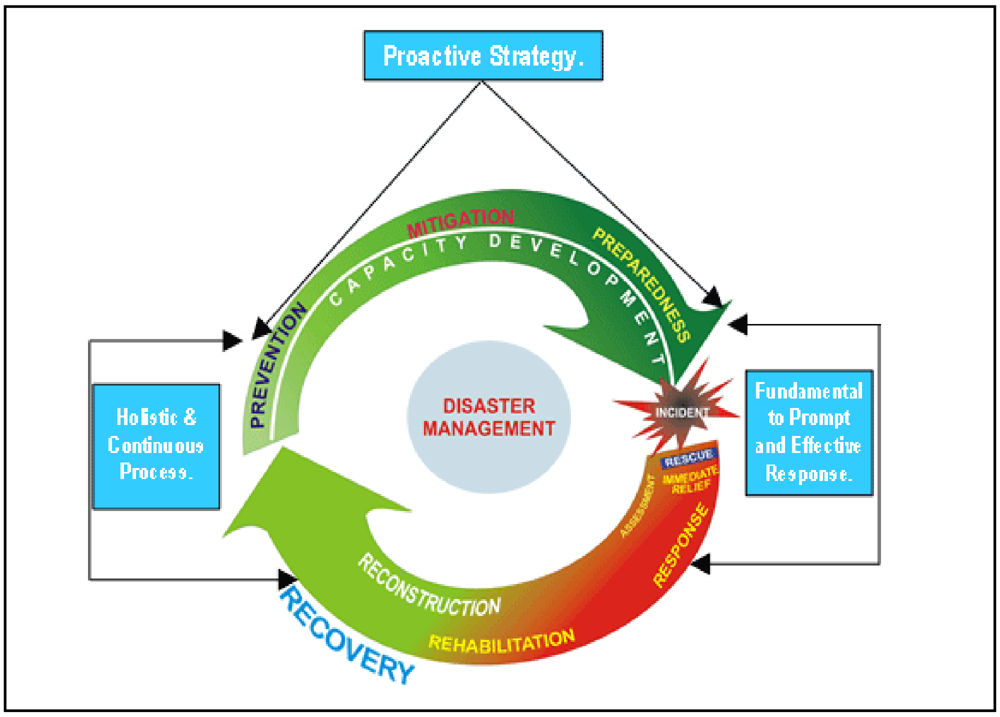


Figure 1-1: Disaster Management Continuum as adopted by NDMA, Govt of India [1]

Disaster management involves four phases or components: Mitigation, Preparedness, Response and Recovery. NDMA as an organization is involved in all the four phases of disaster management. **Mitigation** can be described as reducing the likelihood of a disaster. **Preparedness** is about informing and equipping people who may be affected by a disaster. **Response** phase involves taking steps to reduce or eliminate the impact of disasters that have occurred or are currently occurring, in order to prevent further loss. Relief is also a component of response. **Recovery** is a long term goal to restore the affected livelihood back to normal [1].

Figure 1.1 is the disaster management continuum as adopted by NDMA. It displays the four stages as six elements i.e., Prevention, Mitigation and Preparedness in the pre-disaster phase, and Response, Rehabilitation and Reconstruction in the post-disaster phase, thus defining a complete approach to Disaster Management. NDMA lays guidelines specific to disaster types. These guidelines explain the course of action to be followed when disaster strikes. Standard Operating Procedures (SOP) for each kind of disaster are formulated by NDMA, and are executed at the time of disaster [2]. The NDRF (National Disaster Response Force) is responsible for specialized response to disasters.

For the purpose of this research, the focus is on **Response** and **Relief**. The response phase in disaster management requires effective means of communication. Information at this time is crucial and needs reliable, yet quick and efficient ways to be delivered [3]. It is a challenging situation to work in. To understand the role that information plays during disaster management, it was important to take into account a particular type of disaster and review its rescue activities. Hence we will make a review of a disaster that occurred recently in India and was reported to be massive on account of the damage it caused.

Uttarakhand (North India) experienced a disaster in June 2013, of a magnitude that has not been reported in the recent disaster history. The state received torrential rainfall measuring 64.5 mm to even 244.4 mm as reported by IMD (Indian Meteorological Department) [4]. This was considered to be the country's worst natural disaster since the tsunami of 2004. The Uttarakhand disaster is termed as flash flood, and it is the disaster type that we shall focus upon, for the purpose of this research project. Sudden heavy downpour called cloudbursts, over a small region cause devastating flash floods[5]. They are characterized by fast rise and recession of water flow, thereby causing damage due to suddenness. For the case of Uttarakhand, the combined effect of massive landslides and continuous rainfall in the peak pilgrimage season was the reason for enormous loss of life and property.

The areas affected were major pilgrimage locations including Gangotri, Yamunotri, Kedarnath, Badrinath, the Valley of Flowers, Roopkund and Hemkund. Hundreds of people lost their lives and massive destruction to the infrastructure was reported. Over 100,000 people were reported stranded. Rescue operations became all the more difficult due to damaged modes of transport, rough terrain, heavy fog and rainfall. Government organizations, Non-Government organizations (NGOs) and local administrative bodies worked together for quick rescue operations. The relief and rescue operations continued for many days after the disaster. Table 1.1 summarizes the rescue operations conducted by various organizations [5].

Table 1.1: Summary of rescue operations in Uttarakhand disaster

Rescue operation	Number of people	Conducted by
Trapped people were rescued and moved to safe places. Also relief supplies and essential commodities were air dropped at several places.	23775	IAF (Indian Air Force) – Rescue Operation named "Surya Hope"
Rescued people to safe places	38750	Indian Army for the operation "Surya Hope"
Rescued people to safe places	33000	ITBP (Indo-Tibetan Border Police) for the operation "Surya Hope"
Rescued people to safe places	9000	NDRF (National Disaster Response force) for the operation "Surya Hope"
Airlifted people to safer places	13000	Civil Aviation helicopters
Helped in rescue and relief activities	——	State government agencies like police department, district authorities, (NGOs) and volunteers

1.2. Motivation and problem statement

Reporting information is an important aspect of disaster management especially in the rescue and relief phases. The rescue workers need current information and also need to report field data to stake-holders such as government agencies and news media. The use of mobile technology as an alternative to the conventional modes adopted for field data collection i.e. paper-based surveys and reports can be a breakthrough. Paper-based methods are time-consuming, labour-intensive and susceptible to error. The need for acquiring and visualizing spatial and non-spatial data on portable devices has been realized in research and in industry. Various Remote sensing and GIS techniques are used in natural disasters to report and retrieve information [6]. This addresses the bottle necks in the task of disaster management in India which is primarily the availability of both geo-spatial and non-spatial data on portable and less costly devices.

Flash floods cause massive infrastructural damage due to which the communication channels are usually disrupted. But for the rescue teams, it is fundamentally important to communicate and retrieve information. Constant efforts were made by the service providers in Uttarakhand region to revive communication channels after the disaster. It became an important aspect of the rescue operation to restore the communication links. This implies network and communication restoration is also given high importance during rescue operations. Though communication failure seems to be a major hindrance, to mobile device's full operation but these devices can be of use even in offline mode. The GPS (Global Positioning System) and some other sensors remain functional and are independent of network access. The mobile devices can still serve as portable map and reporting devices in cases of non-connectivity. The application must be able to work in both connected and disconnected mode. While network access is unavailable, the user can use the application for data collection. Once the communication channels are functional again, the devices can be used to send/retrieve information that needs to be exchanged. The mobile application has to be designed and used in a way that proves functional while network access is intermittent and of course also when communication channels are restored.

GeoTools, an Android application for geologists, demonstrated the ease of use that portable devices offer for field data collection as compared to conventional non-digital methods like paper forms and reports for data collection [7]. Similar to geologists, rescue workers also need to collect field data for disaster management. They need to record location, report incidents and sketch affected areas. Separate devices and equipment are available for the above needs but it is not always convenient to carry all instruments and devices to field. It would be better comparatively to have one single portable device with multiple functions. Digital methods help overcome human errors, which are introduced while collecting data through non-digital methods. Mobile devices also have limitations like dependency on network, battery and sometimes require a certain skill to work with them. This research will take into account these limitations and propose ways to overcome them.

Shu *et al.*, 2009 [8] presented a design method of a location-based mobile service and pointed out the simplicity that Android offers for developing map and location applications. Flexible displays of map and control functions are features provided by Android that support the design of location-based applications.

Apart from this, it is critical in disaster management to keep decision-makers informed. This is different from field data collection, where the field workers can return to office and analyze the data collected. Hence a portable and light weight device that has access to communication network can serve the purpose. Though mobile devices find ubiquitous use in India, the use of such devices is quite limited for the cause of disaster management. A single mobile application if made available to the rescue teams will provide a better platform and help improve co-ordination and management. This research work is to develop an application with a hope to make a great impact on the way information is dealt with, during disaster management operations in India. The proposed application will exhibit spatial functionality with a user-friendly interface and is expected to help in disaster management rescue operations.

1.3. Research identification

The research project aims to study the requirements of the teams working in disaster management to understand the situations in which they operate, and provide supportive functions realized in an Android application to improve disaster management. Refining the disaster type to flash floods and the disaster management phase to response phase , was a conscious decision and allows us to gather specific requirements. The Android application developed as part of this research, is intended to support rescue workers' co-ordination, information exchange and consequently better disaster management. It should provide spatial functionality, effective visualization of spatial data and ability to send/retrieve essential information through the device.

The geographic phenomena and the climatic conditions that cause the floods are not the objectives of study because that is another broad domain. The measures for effective predictions and warning of floods are also out of scope of this research work. The work will involve understanding of information requirements for response phase in disaster management during flash floods and development of an Android application fulfilling them.

1.4. Research objectives

1. To study the role of information in the rescue phase of flash flood disaster management.
2. To study the administrative and technical obligations for communication in flash flood disaster management.
3. To develop a mobile application meeting information and communication needs during such disasters.
4. To evaluate and test the mobile application in a virtual scenario.

1.5. Research questions

1. What is the essential functionality required in the mobile application?
2. Is the spatial functionality achievable through a simple interface?
3. What can be the criteria for application evaluation?
4. How to make the application serviceable in the event of network failure?

1.6. Innovation aimed at

This research focuses on exploring the potential of an Android mobile application in the response phase of flash flood disaster in India. It aims to enable well-coordinated information exchange and therefore better disaster management.

1.7. Thesis structure

The research work is organized into the following chapters. Chapter 1, Introduction explains the need and purpose of the research. It consists of background of work, motivation and research identification. Chapter 2 summarizes the various research efforts already done in the field. Subsequently in Chapter 3, the research plan and methodology is discussed. It includes the details of the interview and discussion with rescue workers. It also lists out the analysis made and the requirements extracted, to be developed as functionality in the mobile application. Design and implementation of the Android application is presented in Chapter 4. Results and discussions are explained as part of Chapter 5 while Chapter 6 presents conclusions and recommendations for further work.

2. LITERATURE REVIEW

2.1. Disaster Management

Over the past decade, there has been an alarming increase in the frequency and also in the intensity of disasters in various parts of the world. The Haiti earthquake, Katrina hurricane, Indian Ocean Tsunami and the Nargis cyclone have been major disasters over the last decade. These disasters force mankind to think of ways to cope up with such calamities. The increased number of affected people and spatial extents of disaster impact due to disaster and the high economic losses from natural disasters are a cause of great concern. The economic growth of the affected country slows down and takes decades to recover completely. Especially the developing and under-developed countries take a longer time to recover from such losses. This calls for enhanced methods and techniques for disaster management. Every country has to realize this need and think of better ways to manage disasters. It is true that natural disasters occur at a scale that causes much destruction and many losses which are unavoidable. Even though scientists and researchers work constantly on predictions and weather forecasts but sometimes they do fail to foresee the impact a certain disaster would have. Sometimes they are caught totally unaware by such calamities. But management of rescue and relief operations can be planned before-hand and executed at the time of disaster. The disaster management techniques, even if already existing, need to be constantly revised and updated. This is because new technologies keep coming up and should be leveraged for disaster management to produce profitable results.

Doner and Yomralioglu (2008) [9] made an assessment of using MGIS (Mobile Geographic Information Systems) as a tool for acquisition of geo-data. They described and discussed an MGIS developed for field data collection that also utilizes GPS. It resulted in a noticeable gain in efficiency, accuracy and reduced cost while meeting the project completion time. It designated MGIS as an appropriate tool for field data surveys showing the kind of efficiency and accuracy that would be expected from a disaster management application.

A project worth a mention in this aspect is Sahana Eden [10]. It was initially conceptualized as Sahana Software and developed by the IT community of Sri Lanka to help in the relief efforts after the 2004 Indian Ocean Tsunami. This software package evolved to be known as Sahana Eden. It was supported by a voluntary community that includes disaster management practitioners, academicians, students and companies. The communities promoted customization of the software by making it available as free open-source software, with related standards. The project was a successful implementation of a disaster management solution to help manage relief organizations, people involved, assets and inventory. The project also used maps to make assessments and bring about awareness of the situation. This set a perfect example and motivation to think of a technology-based solution to help improve the way disasters are addressed.

Karnatak *et al.,* 2011 [11] presented use of mobile application for flood disaster management in Assam, India. The mobile application developed under this study used Windows as a platform for application development. The data from field to control room was transferred through an XML message containing different types of data including geo-location and images. The XML-based data transmission from mobile application to server proved to be an effective mode of communication. The data received from mobile device was integrated in Web-GIS application in a real-time environment and other flood specific GIS data were also accessed from remote servers.

2.2. Role of information in disaster management

"Experience without theory is blind, but theory without experience is mere intellectual play."

-Immanuel Kant

Kevany (2008) [12] unfolded interesting information through his own experience in the disaster management operations that took place after the World Trade Center mishap on 09/11/2001 in New York. The author highlighted the importance of a strong relationship between the technologies applied for disaster management and the needs of disaster response representatives. This came from an observation that the design and deployment of technology to serve disaster management, was performed by technologists without any participation of end users. The technologists lacked field experience in disaster management and vice versa. To merge the knowledge of the two was critical for effective utilization of the technology. This particular disaster commonly referred to as 9/11, had a great impact on the way geo-information is used for disaster management. The author emphasised the relevance of geo-information i.e. location information. Most importantly the location of disaster impact, response teams and resources should be known. An extensive list of geo-information data requirements was provided in the paper. It also discussed the requirements for geo-information management.

Preparation of information is essential for relief management. Tsai *et al.*, 2013 [13] discussed three problem scenarios that require adequate and timely information during a disaster response and recovery mission: (a) inadequate escape guidelines for people, (b) incomplete geographical information for relief workers and (c) insufficient on-site information for disaster managers. This research work signified the importance of real-time exchange of both on-site and off-site information for disaster management. It involved three entities in the example scenarios: rescue/relief workers, victim and disaster manager. On-site information was required by the relief workers to plan evacuation routes and perform rescue operations whereas off-site information was used by disaster managers for better decision-making and management. For example disaster managers retrieved location information from relief workers and plotted them as point locations on the guidelines maps. As soon as they received emergency assistance requirement from victim, they could locate a rescue worker closest to the spot and convey the information. This highlighted the need for simultaneous execution of the on-site and off-site operations resulting in updated information at both ends and prompt emergency response. A mobile application was designed that displayed guidelines for on-site escape and rescue. Successful results were achieved after testing the integrated information on a mobile platform in a simulated scenario. Participants enacted as victims, on-site rescue workers and off-site disaster managers. Disaster managers updated current information as received from on-site rescue workers. Victims saved time in performing self evacuation with help of emergency guidelines received from off-site disaster managers. The study was conducted for the Typhoon disaster type, but indicated applicability to other disaster scenarios.

Mobile technology has shown a considerable impact on society over the past few years. Gradually, this technology has become a part of every person's life and is the easiest mode to deliver information. People do not get tired of experimenting and exploring new apps and developers leverage this advantage to create and innovative and challenging apps. Different operating systems are available for different mobile devices. Google's Android, Apple's iOS, Blackberry, Symbian and Windows are all available for development. Considering Android as a target platform for application comes from the review of various works using Android technology. The Android SDK allows application development with great ease. There are many inbuilt features and tools in an Android device which can be integrated and programmed to be used as and when required from within the application.

Location-based services are described as information services depending on user's current location and are being used judiciously in every aspect [14]. Be it traffic information, advertisements and promotions depending on locations or knowing amenities nearest to one's own location. Such services can be leveraged for disaster management scenarios. They are implemented in mobile devices through A-GPS (Assisted GPS) technology. This technology integrates the location information through mobile network with that retrieved from the device's GPS. This helps to quickly know location, consumes less battery, has better coverage and can, in some cases, be used inside buildings [15]. The application can also make use of camera, compass, calls, SMS, email facilities and the developer can think of a number of ways to create features in an application that utilize these inbuilt facilities.

Geographic information services through mobile can help emergency response teams to identify areas under threat or impact of disaster. Mobile GIS information combined with GPS and satellite imagery can help retrieve important results like quick execution of evacuation plans and tracking of emergency response vehicles [16]. Quick reporting of incidents and information is a critical task and can be accomplished through smartphone applications. A disaster management application is helpful if it captures important data like location of injured and non-injured people, images of the damaged area, weather updates for the particular location, alarms and alerts, quick access to maps and location-based services [17].

2.3. Related work in Android technology

The Google Play Store, a virtual marketplace that provides services and applications to Android devices, presents a number of applications upon searches by the keyword 'emergency'. Gomez *et al.*, 2013 [18] analyzed 250 emergency-based Android applications. For each application, the data (description of the application, version of Android targeted, price, rating, and application technology) was collected. These applications were dedicated to either detection of emergency, notification or management. Depending on the target end users of each application, the applications were divided into categories to allow further analysis. The categories were:

- Victim (59%)
- Professionals Rescue Team (14%)
- Rescue Voluntary (14%)
- Witness (7%)
- General Public (6%) – This is for those not affected by the disaster

The study and review of these applications resulted in the conclusion that more than half of these applications were designed for victims. There were fewer applications addressing rescue workers to support in rescue operations. Also the major features of the applications were studied. These included:

- Location data access (Fine, coarse or mock location)
- Communication resources access (Bluetooth, internet access)
- Communication tools access (SMS, email, phone numbers)

The location service was calculated to be used by 81.2% of the applications. 106 apps use GPS sensors whereas 81 apps rely on the network communication. Most of them combine both technologies. This is an important consideration since GPS does not work without sky visibility. Through this research work [18] an application was proposed for better organization of relief activities. The study also considered the possibility of using internal sensors within device for example shaking the device to raise an alarm and changes in user position. The framework was designed to work in different modes (victim mode, volunteer mode, witness mode and citizen mode) to target different needs. This study revealed the scope of experimenting with application development in the context of disaster management by targeting them for specific end users.

Palmer *et al.*, 2012 [19] proposed a framework to help overcome the challenges in the domain of disaster management. RAVEN – a framework to construct an application was provided with the utility to collect data using smartphones. The framework was built with the idea to allow developers to construct a user interface and define a schema at runtime. The aspect of technological advancements in the field of disaster management was also given importance. Technology can help manage the operations and can also act as a networking channel. The framework allowed defining a schema for a structured data store at runtime and generating a user interface to edit instances of that schema on the mobile devices. Developers could quickly create data-oriented applications on the phone itself in a short span of time. A "People Finder" application using this framework was also implemented. To be able to compare and contrast the development effort required and the usability of the application, the application was developed both by using a standard Android implementation and by using a framework implementation.

- Standard Android implementation

Standard Android development tools and practices were used by application developers to develop the People Finder application. This implementation consumed around 3500 lines of code. This implied that this kind of implementation would require experienced application developers to complete the task and take time for completion. This contradicted the fact that rapid application deployment is required when disaster strikes. Hence the author considered another implementation as well.

- Raven implementation

[19] This implementation was entirely done using the mobile device. It used Schema Creator application to construct a schema which is not visible to the user but used internally to create the database. This enabled an edit-user list with a simple user interface similar to the one developed using Android to be created on the fly. The paper then made a comparison between the two approaches and concluded that time consumed to develop the application using the Raven framework was lesser than the one using the standard Android tools. Also the framework implementation required just a smartphone. Thus this framework offered rapid application development in case of disaster.

Asif *et al.*, 2012 [20] presented a framework through smartphones for disaster management to cope with mitigation activities post disaster. A smartphone application was proposed to enable information gathering and retrieval by the general public and decision-makers. This would help in the rehabilitation and rescue processes involved in disaster management. This framework allowed data collection from the disaster struck areas even in offline mode i.e. when the cellular network is unavailable. The application was able to transfer data to a central database through a web application. This data resided on a central database and used for further analysis.

Similar to India's problem of absence of an efficient technological disaster management system, Fajardo *et al.*, 2010 [21] discussed the need for such a system in The Philippines. An Android application named MyDisasterDroid was developed for improved co-ordination in relief and rescue activities. Geographic locations were entered by the user through the application interface and stored in the application database. Paths between these locations were calculated with a travelling salesman algorithm while genetic algorithms were used to decide the optimal path. The application interface consisted of two views: the map view and the list view. While the map view showed the locations overlaid on a Google map, the list view showed the list of people who needed help and their locations. The application displayed an optimal route among the locations with a single button click. The user could prioritize the routes either by closer location or the route with more number of people in need. But the routes did not address the safety and concerns particularly of the rescue worker.

2.4. Dealing with disrupted communication

In disaster-struck areas, there is an adverse effect on communication infrastructure due to congestion or physical damage. Not many applications address this issue or provide ways to overcome it. Oppcoms (or opportunistic communications) were suggested by Gorbil and Gelenbe (2013) [22] as a method of information exchange when mobile communication is unavailable. In this approach, GPS and Bluetooth were used to exchange messages at close ranges and carried over multiple hops in a store-carry-forward manner. Also a model of communication through aerial wireless technology was discussed by Shao *et al.*, 2011 [23]. It proposed a BTS (Base Transceiver Station) to be installed in an air craft. This BTS acted as a temporary aerial base station. One such base station could span a large area thereby providing network coverage.

Eleiche and Markus (2011) [24] discussed the importance of mobile devices in acquisition and processing of geospatial data and designing mobility applications. To enable users to perform geospatial functions in an offline mode, a conceptual stand-alone framework was proposed. This framework was devised to provide spatial functionality in a mobile device, independent of the communication network. The proposed solution was to store the objects of interest (images or data) in the mobile device memory instead of the server. These objects of interest once accessible from the mobile device could be used for processing and analysis.

As soon as the relief and rescue work is initiated in disaster-struck regions, it also becomes inevitable for the communication network to recover since it has a prime importance in such situations. Smartphones have the privilege of being able to connect to different networks such as cellular network (3G, GPRS, EDGE) or Wi-Fi. These devices provide high utility in different kinds of emergency situations. A study by Farnham *et al.* 2006 revealed that cell phones network did recover quickly after the Hurricane Katrina disaster [25]. Thus it would be a wise decision to implement an application in such a way that it shows usability in absence of network communication and quickly notify of available networks to perform tasks that need network access.

3. RESEARCH METHODOLOGY

This chapter identifies the methodology that was applied for this research project. Figure 3.1 depicts an overview of the methodology and approach used for the research. It consists of four stages *viz* Requirement analysis, Design and Development of the software application, Configuration of deployment environment and finally the Testing and Evaluation stage. Each stage and its sub-parts are explained subsequently.

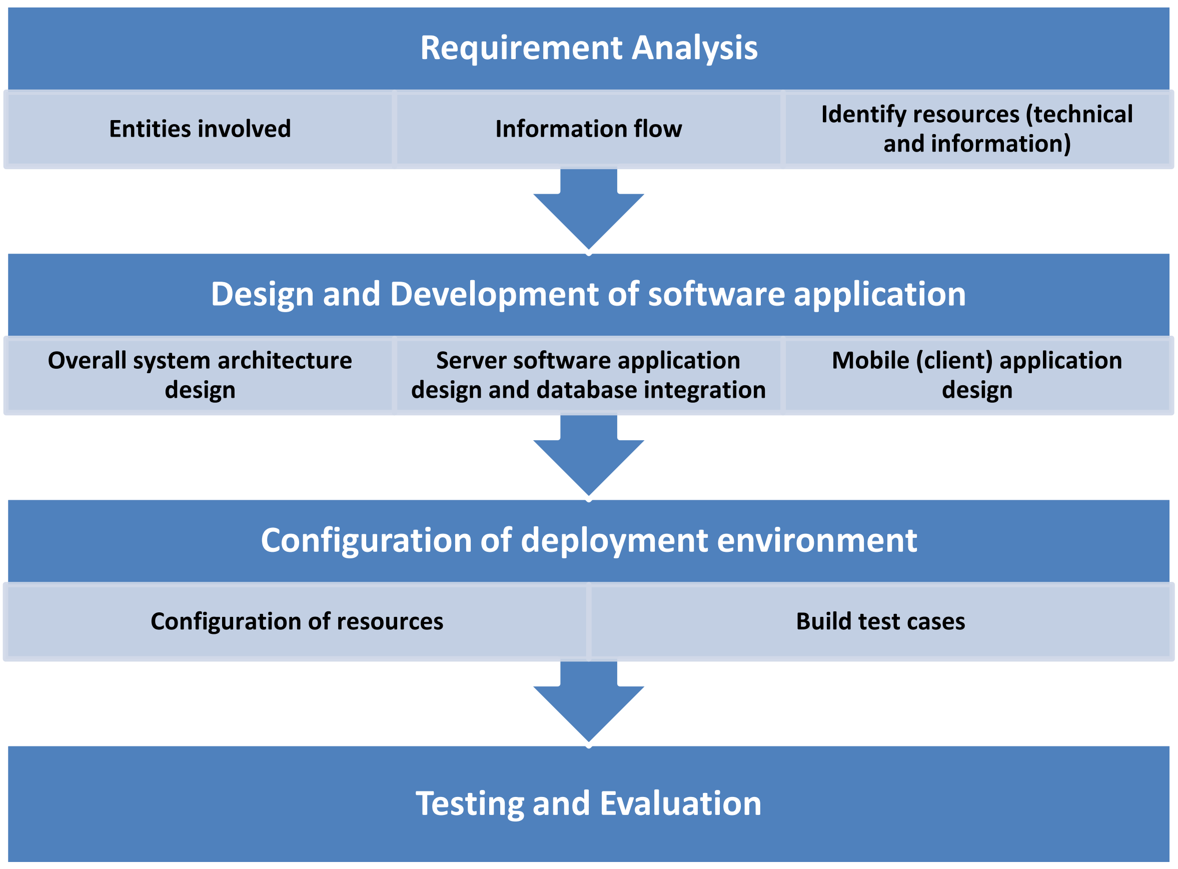

Requirement Analysis
Entities involved
Information flow
Identify resources (technical and information)
Design and Development of software application
Overall system architecture design
Server software application design and database integration
Mobile (client) application design
Configuration of deployment environment
Configuration of resources
Build test cases
Testing and Evaluation

Figure 3-1: Framework for Research Methodology

3.1. Requirement Analysis

This stage involves gathering information and performing an analysis to extract requirements. For this, it was important to understand how the disaster management authorities and organizations work, the scenarios they face during the post-disaster phase and identify the areas of possible improvement. It was required to know the kind of information they need and equipment they have for communication. A meeting was scheduled with The Deputy Commandant, NDRF at NDRF Headquarter, New Delhi, India. The agenda of the meeting was to discuss the basic practices during rescue operations by NDRF personnel. Since the research project cites Uttarakhand flash floods as an example scenario, it was the objective of discussion and explanation of processes. This meeting helped gather quite a lot of important information. Email and telephonic communication with other rescue workers at different seniority levels, was also carried out. The information extracted from the whole discussion has been assembled and discussed in coming sections.

3.1.1. How does NDRF work?

Once a disaster occurs, it calls for immediate, prompt and active response. NDRF is a dedicated response unit from the Government of India that is completely involved in disaster response, disaster risk reduction, mitigation and disaster prevention. Set up under the provisions of the Disaster Management Act, NDRF functions under the Ministry of Home Affairs, Government of India and supervised by NDMA. NDRF battalions are located at ten different locations across the country based on the vulnerability profile of specific regions in the country.

The Uttarakhand disaster owing to its magnitude and the number of people affected was given the National level disaster criteria. Once a disaster is categorized as National level disaster, the National Crisis Management Group comes into the picture. The central government plays an important role in such a case. For small scale disaster types, management is restricted to district and state level government and management committees if claimed to be of even small level. NDRF is called upon through either NDMA or MHA (Ministry of Home Affairs). NDRF can even respond by itself if it sees a situation of emergency. The NDRF personnel communicate with the central government or state government. At district level, they communicate with the district magistrate (DM)/District Collector (DC) and for state level – it is the Chief Secretary or Relief Commissioner. Either NDMA or NDRF directly communicates the requirements and the Relief Commissioner, Chief Secretary or District Magistrate sees to it. The relief work can be performed by rescue workers (NDRF themselves) but only after the rescue operations. The relief task can also be performed by the NGO's and relief organizations and the DM/DC takes this decision as per the situation. No separate teams are formed to carry out rescue and relief operations. Co-ordination of team members during the rescue and relief operations is controlled and supervised by a team officer from Govt. agencies and department concerned, as per SOP (Standard Operating Procedures) laid down by NDMA [2].

In case of a disaster, information related to ground reality plays an important role. There is a great deal of variety of information that is required. It is crucial to know the disaster affected locations, location of relief camps, equipment and amenities required and weather information. The people indirectly affected from the disaster need to know about their relatives and acquaintances unfortunate to have been caught in the fury of the disaster. The news channels and journalists take the onus of imparting all this information to the public. It so happened in Uttarakhand rescue operations that some sites were not accessible to journalists because of difficult weather and terrain. In such cases where journalists are not allowed or are not able to reach disaster sites, the rescue workers need to take over the responsibility of providing information as well.

3.1.2. How is the rescue operation conducted?

NDRF is the only disaster resilience agency in India that has adequate modern equipment to carry out rescue and relief operations in any kind of natural and man-made disaster. In response to a disaster, the following activities are conducted; either being on-site or off-site.

On-site operations

- Quick assessment of the situation in coordination with disaster management officials viz NDRF, NDMA representatives of State/Union Government authorities, local authorities, defense (Army/Air Force/Navy).Central Armed Police Forces and Police
- Mobilization of disaster management teams
- Arrangement of vehicles, helicopters, boats etc for movement of teams and equipment
- Establishment of communication network
- Conduct rescue and relief operation as per the requirement of the state authorities

Off-site operations

- Provision of medical help.
- Assisting state/local authorities in rehabilitation of the victims
- Adequate provisions of food, drinking water and other amenities
- Keeping constant contact with IMD authorities to react in the event of re-occurrence of disaster.

3.1.3. Information requirement and communication

To understand the communication scenario during rescue operations, several questions were posed regarding communication of information and equipment used. They are presented below in the form of questionnaire.

Q: How is the disaster management team organized? Who are the decision-makers?

A: In case of major disaster like Uttarakhand operation recently, National Crisis Management Teams take decision. The Government/NDMA/NDRF and the concerned State Governments are the decision-makers for the organization of disaster management teams to carry out the rescue and relief operations when a disaster occurs.

Q: Whom do the teams report to?

A: The teams report to the State government authorities like Relief Commissioner, Chief Secretary, and District Collector.

Q: Which teams respond first and who takes these decisions?

A: If it is beyond the capacity of State government, they seek NDMA/NDRF for sending disaster teams or otherwise. This decision is made by MHA or NDMA. Local police, volunteers and NGO's are also sent for disaster mitigation. The NDRF teams and personnel of CAPF's (Central Armed Police Forces) are pressed into service on occurrence of emergency.

Q: What is the information required before-hand and on the spot?

A: The place of occurrence, type of disaster, effect of devastation, number of casualties, weather conditions and conditions of roads and bridges and other access to the incident site, availability of amenities needs to be communicated before hand by the assessment team. On the spot information includes availability and requirement of equipment, operational preparedness, availability and requirement of medical aids and details of official's in-charge.

Q: What data needs to be sent to the decision makers?

A: The strength of teams/troops to be deployed, place of concentration of teams or Relief Camps, operational strategy, details of equipment, supporting elements and roadmap of operations to be carried out.

Q: Is there any existing use of geospatial data. Are there any portable devices used?

A: Yes, geospatial data in the form of paper maps or digital imagery is required. Mobile phones and tablets are used. But the usage is totally dependent on the availability of mobile towers and internet access which may fail in such situations.

Q: What are the limitations of the available devices? What features if implemented, would be helpful for rescue operations?

A: As stated earlier, the usage of such devices is totally dependent on the availability of the communication channel and internet access. But assuming that the communication infrastructure has been restored, there can be many possible features which would be helpful to the teams.

- Disaster alert feature and notification of potential hazards
- Evacuation routes as per availability of current maps of the region
- Splitting of the area into small zones would help in better visualization
- Current maps to be made available to the rescue teams
- Image geo-tagging
- Reporting of environmental factors, situation analysis of the disaster-struck area, and accessibility to the area.

Q: What kind of data analysis is required?

A: Weather situation, location and accessibility to the disaster struck region are the main data inputs required for analysis. The data analysis can be the impact of disaster in terms of people affected, infrastructure damage reported, relief camp locations and their proximity to necessary amenities. Weather information and prediction will help for planning further rescue and mitigation activities. The data analysis would be helpful in all spheres of disaster mitigation mechanism for positive evaluation of the situation prevailing in the site of disaster.

Q: How capable are the rescue workers to use such devices?

A: The rescue and relief teams are possible users of such devices. The rescue workers are well trained and conversant to operate such devices. Apart from this, adequate training is imparted to the teams in rescue and relief operations. They are being trained in chemical, biological, radio-active and nuclear (CBRN) emergencies, collapsed structural search and rescue (CSSR) operations, deep diving, flood rescue and heli- slithering.

Q: What if the GPRS/ Internet do not work?

A: Besides cell phones and internet connectivity, the teams are communicating with wireless communication equipment available with the responders. Wi-fi and other communication equipment are available with the teams to communicate in the event of any emergency situations. Establishment of INMARSAT sets is an alternative mode of communication. Communication through satellite phone is also used if available.

Q: How do the rescue and relief teams' co-ordinate?

A: A proper operational regime is formed for rescue and relief operations in the event of disasters. The teams are deployed for operations as per the 'Standing Operation Procedures' (SOP) issued by the Government of India. No separate teams are formed to carry out rescue and relief operations. Coordination of the team members during rescue and relief operations are controlled and supervised by a team officer comprising Government agencies and departments concerned as per the SOP's.

Source – Interview with Mr. R. K Srivastava, Deputy Commandant, NDRF

Vasant Kunj, New Delhi

26 Sept, 2013 [26]

Also the news reports and articles related to the incidents in Uttarakhand have been studied. With the help of this, a brief summary of the series of incidents and rescue operations in Uttarakhand, over a week, has been drafted.

Summary Report.

June 14, 2013 Uttarakhand, India

Torrential rainfall followed by floods and landslides affects the state of Uttarakhand severely. The early monsoons have brought misery in the life of the people in Uttarakhand, especially the Rudraprayag, Pauri, Uttarkashi, Chamoli and Tehri districts in the state. The current death toll is reported to be higher than 1000.

June 16, 2013 Uttarakhand, India

Heavy rainfall and cloudburst have accelerated the flash floods. 12 out of 13 districts in Uttarakhand have been affected. Rudraprayag, Chamoli, Uttarkashi and Pithoragarh are the worst affected districts. A team by Indian Red Cross has been sent to make an assessment of needs which would be followed up by rescue and relief work.

June 17, 2013 Uttarakhand, India

A huge reservoir above the land area of the Kedarnath temple burst, releasing huge volume of water. A cloudburst in the same area has worsened the situation and filled the temple and surrounding area with gushing flow of water and buried thousands of people. Rescue operations have started. Today the rainfall completes continuous 60 hours.

June 18, 2013 Uttarakhand, India

The NDRF team lands at Guptkashi (located between Kedarnath and Rudraprayag). A lot of infrastructural damage is reported and quick evacuation measures are required for people buried under structures. UAV named Netra along with two more hired UAV's is being used by NDRF to monitor images and ensure complete evacuation.

June 19, 2013 Uttarakhand, India

The state government has deployed helicopters to rescue people held up in different areas. District administration has released alerts and evacuated population living on the banks of the river. In terms of relief supply there is an immediate need of food, shelter, potable water and clothes.

June 21, 2013 Uttarakhand, India

The rescue operations are going on in various districts in Uttarakhand. Over 250 personnel of the NDRF have been deployed. The Indian Army has deployed 10,000 soldiers and 11 helicopters. The Indian Air Force has already airlifted 18424 people and also dropping/landing of relief material has begun.

Source [27] [28]

Requirement gathering consisted of reviewing research works, articles, journals, news reports and most importantly a personal interview with NDRF official. As per the interview conducted, summary of events and the literature review [11], the information has been analyzed and specific requirements have been extracted from it. The functionality in the application is carefully chosen for particular scenarios that have been drafted through careful evaluation of all the requirements. The Android application should possess the below functionality:

1. **Reporting system:** The application allows rescue workers to report incidents and other information when needed. Such reports can be about people rescued, resources required, information about disaster and an assessment of its impact. Two reports used for this functionality are First Incident Report and Summary Report.

2. **Emergency/Distress call:** The application allows immediate call to an emergency response office. Request can be about an airlift (evacuation or medical aid) or a need for more resources (rescue workers or materials).

3. **Disaster Alert:** Weather-related information is important to be prepared for any further chances of disaster. The application allows visualization of current weather and weather forecast updates.

4. **Geo-visualization System:** Map viewer to visualize the current user's location, nearby amenities, weather information and spatial extent of the disaster.

It became easier to understand who would be the target end-user of the application, who would need the data for analysis, what kind of data would be sent for analysis and the kind of analysis performed. The sub-parts of the requirement analysis stage are discussed in coming sections.

3.1.4. Entities involved

This application would best serve in the hands of a team officer, as he/she co-ordinates both relief and rescue operations. The data will be sent to Relief Commissioner, Chief Secretary or District Collector whom we will term as Data Analyst. The information that the **team officer** mainly needs is **location, weather updates or disaster warning, location of amenities**. The information that the **analyst** needs is **location, incident reports, resource requirement** as per the location and **visual images or video** to understand the impact of the disaster and communicate to news channels and journalists about the situation. Hereby the entities involved in the scenario are:

- The Team Officer
- The Data Analyst
- Android application
- Server application.

3.1.5. The Information flow

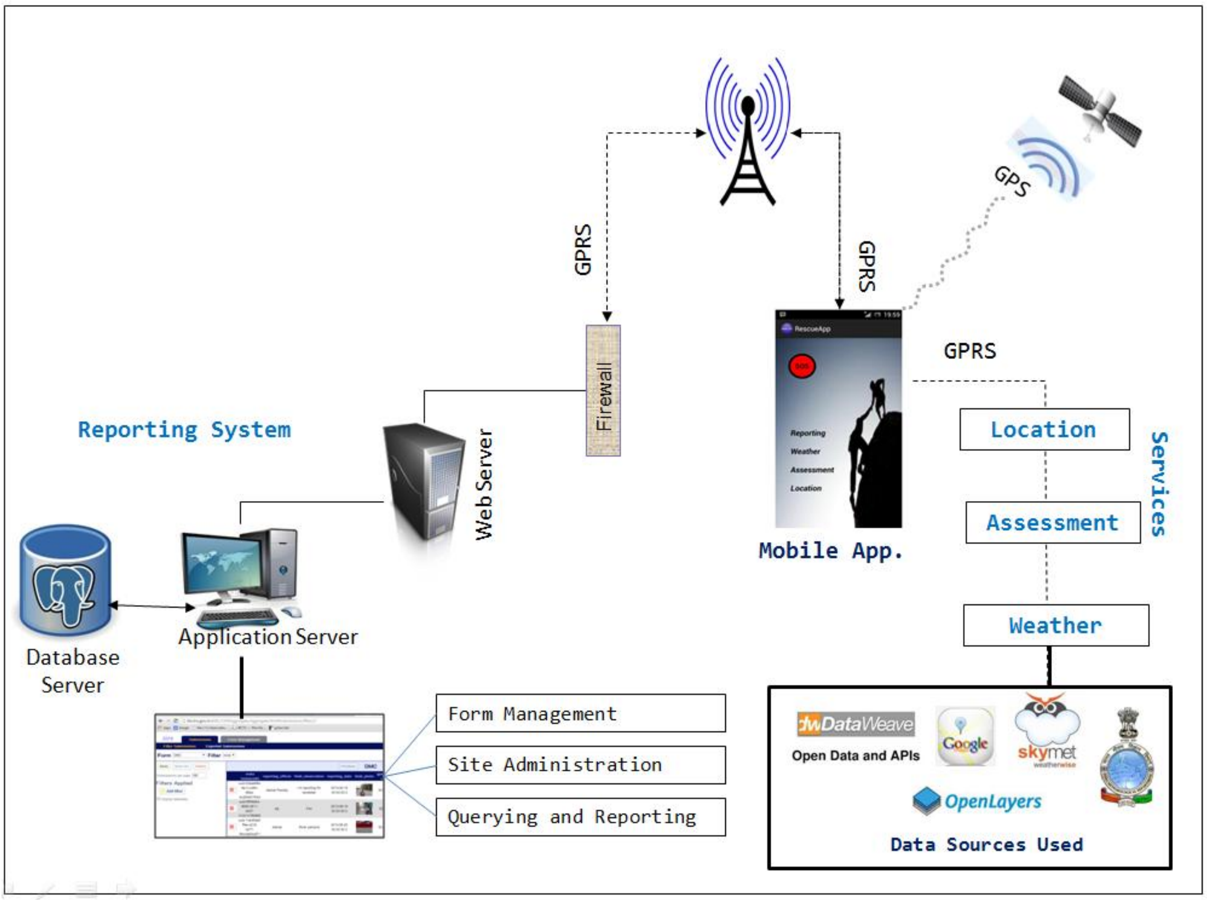


Figure 3-2: Information flow

Figure 3.2 displays the information flow for the application. It shows the communication between the mobile application (client) and the server application. The mobile application needs GPS signals for location services, network services for making calls and sending text messages, and internet access to use maps and weather services. For the reporting system, the mobile app uses GPRS to connect to the web server. The web server, after performing an authentication check enables access to the application server. This application server is also integrated with a database server (PostgreSQL) in this study. Depending on different authentication roles, the user has different access rights at the application server. Site Administration deals with managing the application server and needs admin access. Form management deals with submitting, accessing and deleting report forms access, and this may or may not be given to the end user. Downloading of these report forms is one time effort which can be done online at field or at the base camp/office location itself where network connectivity with server is available. Once data is received at the server from the mobile application, it is updated into the integrated database. Querying and Reporting access allows user to apply SQL queries on the database and generate specific reports for planning and decision- making.

The other functions of Emergency/Distress Call, Disaster Alerts and Geo-visualization are achievable through SOS (commonly used description for distress signal), weather, and location and assessment services. The data sources used are displayed in the information flow diagram. For these services too, the mobile application uses GPS to retrieve and display location and GPRS to connect to the various data sources used.

3.1.6. Resources required

The technical resources required are GPRS and GPS-enabled Android Phone and a computer system to be set up as server. The information resources have been discussed in Section 3.1.3.

3.2. Design and development of software application

This topic is explained in detail in the next chapter but a brief overview about each stage is presented here.

3.2.1. Overall system architecture design

The system architecture design is such that it involves two major entities: the mobile application and server application. The mobile application should be able to send and retrieve data from the server.

3.2.2. Server software application design and database integration

The server application needs to be installed and integrated with database. Once the data is received by the server, it should also be used to update the database. The mobile application should have an authentication to connect to the server. The database connectivity should be managed at the server end only.

3.2.3. Mobile (client) application design

The mobile application is the most important component of this research project. It exhibits the functionality discussed in Chapter 1. Also we tried to address as many as possible requirements that were retrieved from the disaster management officials by their experience during various rescue operations.

3.3. Configuration of deployment environment

The deployment of the application needs some resources to be configured. Most importantly an Android Smartphone with GPRS (preferred 3G), Android OS version 4.0.1 and later versions, inbuilt GPS, total storage of 14 MB(internal storage 9.85MB and external storage 4MB). It can consume additional storage as per usage.. The server and database should be up and working 24/7 to accommodate information retrieval from mobile application and vice versa.

3.3.1. Configuration of resources

The following resources should be configured to fully deploy and test the application

- Android phone
- Network /Internet connectivity
- GPS reception
- Server application
- Database server

3.3.2. Build test cases

The below criteria is devised to test the mobile application. It checks the application performance by testing its accuracy and time consumption under various networks. It checks the application in different field conditions. The memory and battery consumption and lastly usability testing to check the application from user's point of view is also important.

- **Software Testing**
  - Location Accuracy
  - Time consumed
    - GPS
    - Internet (Map loading and information display)
  - Battery consumed
  - Memory consumed
  - Network Access
    - 3G/2G/Wi-Fi
- **Field Testing**
  - Dense forest
  - Urban area
  - Cloud cover
  - Humidity
  - Hilly Terrain
- **Usability Testing** (User Interface and Usefulness)
  - Field Workers
  - Random Users

3.3.3. Testing and evaluation

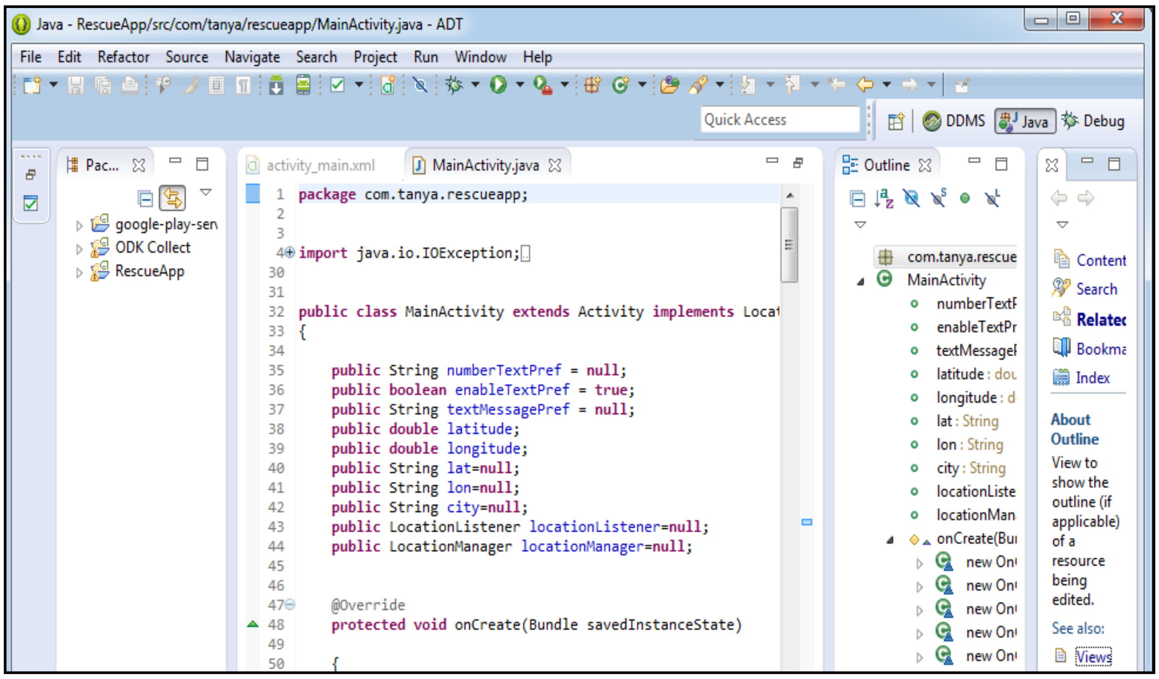
The test cases are formulated taking into account possible factors that may affect the app's utility. Humidity and cloud cover conditions were present in Dehradun itself. Urban area testing was done at Indian Institute of Remote Sensing, Dehradun campus and nearby areas. The Hilly terrain testing has been performed at Mussoorie (about 35 Km from Dehradun and at an average altitude of 6170 ft). The dense forest testing has been performed at Rajaji National Park. It is located between Haridwar and Dehradun and consists of tropical and subtropical moist broadleaf forests. The test results are discussed in Chapter on Results and Discussions.

4. DESIGN , IMPLEMENTATION AND TESTING

Android is Google's mobile operating system and is based on Linux operating system. Android offers its Software Development Kit (SDK) for developing and designing Android apps. For this research project standard Android development tools were used for application development and standard practices suggested in Android development documentation were studied. The integrated development environment (IDE) used in this study is Eclipse (Juno) and the programming language used for software development and customization is Java. For this JRE (Java Run Time Environment) and Java Development Kit (JDK) was installed as well in computer system. For integrating HTML and JavaScript code within the application, PhoneGap was used. The installations required for the application development are as mentioned below. To understand Android programming fundamentals and development tutorials , the website [29] was referred.

4.1. Steps for installations

1. Eclipse (IDE) for Android SDK was downloaded and installed referring to [30]. The ADT (Android Development Tools) package version installed was 22.0.1 and Eclipse version 3.3.2



```
package com.tanya.rescueapp;


import java.io.IOException;


public class MainActivity extends Activity implements Locat
{

    public String numberTextPref = null;
    public boolean enableTextPref = true;
    public String textMessagePref = null;
    public double latitude;
    public double longitude;
    public String lat=null;
    public String lon=null;
    public String city=null;
    public LocationListener locationListener=null;
    public LocationManager locationManager=null;


    @Override
    protected void onCreate(Bundle savedInstanceState)

    {
```

Figure 4-1: Eclipse (IDE) used for application development

2. Oracle JRE version 7 and JDK version 7.0.400.43 was downloaded and installed referring to [31].

3. PhoneGap 2.9.0 was installed from GithHub [32].

For the installation procedure the source [33] was referred.

The functionality that the software application provides is the Reporting System, an Emergency/Distress Call, Disaster alerts and Geo-visualization. The detailed explanation of each feature is presented in the coming sections.

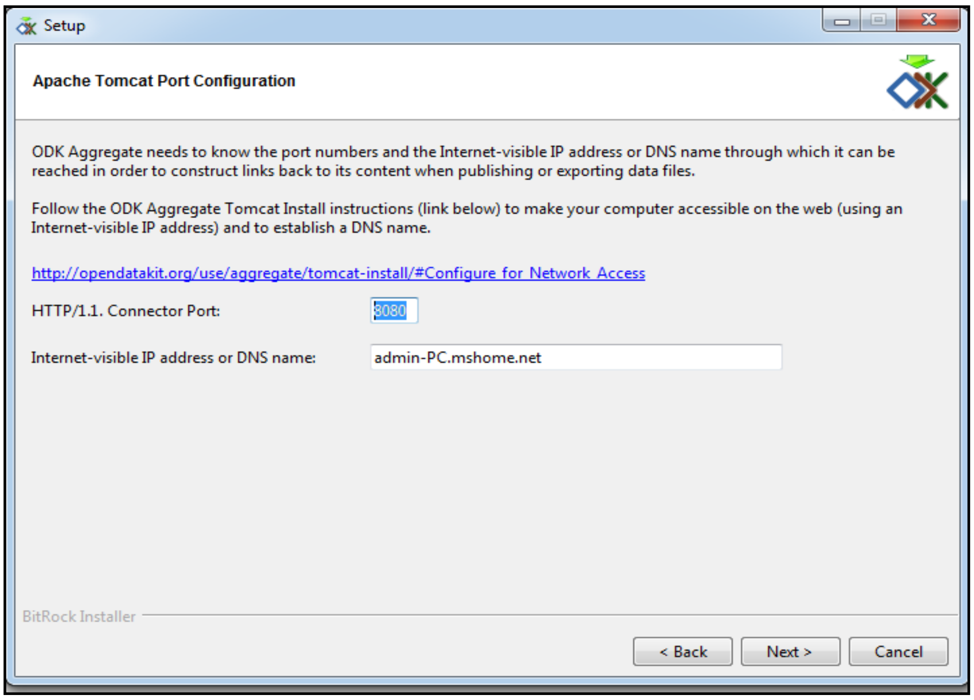
4.2. Reporting System

Reporting ground reality to central system say control room is an important aspect of disaster management. It can be used while the rescue operations are in progress and majorly once the relief operations begin. For the team officer, it is crucial to report incidents and details that are important for analysts. These details include the incident site, exact location coordinates, images, video recordings, number of people rescued, injured and died. The analyst can make assessment of the disaster impact both regarding life and property. The research work enables reporting of this information through an interface which allows user to fill in data. It is a good practice to have template report forms ready to be used at the time of disaster.

ODK (Open Data Kit)[34] is an open source, set of tools that are designed to be customized and used for field data collection. ODK Collect is one of the tools available as an Android application and used for field data collection. It allows user to download predefined forms from the server and fill the data that needs to be reported. ODK Aggregate allows deploying a server with data base integration. These two tools have been used in the study.

4.2.1. Creating Server

- Apache Tomcat Server version 6.0 was downloaded and installed by following instructions from [35] and set the port number for local host as shown in Figure 4.1.



Apache Tomcat Port Configuration

ODK Aggregate needs to know the port numbers and the Internet-visible IP address or DNS name through which it can be reached in order to construct links back to its content when publishing or exporting data files.

Follow the ODK Aggregate Tomcat Install instructions (link below) to make your computer accessible on the web (using an Internet-visible IP address) and to establish a DNS name.

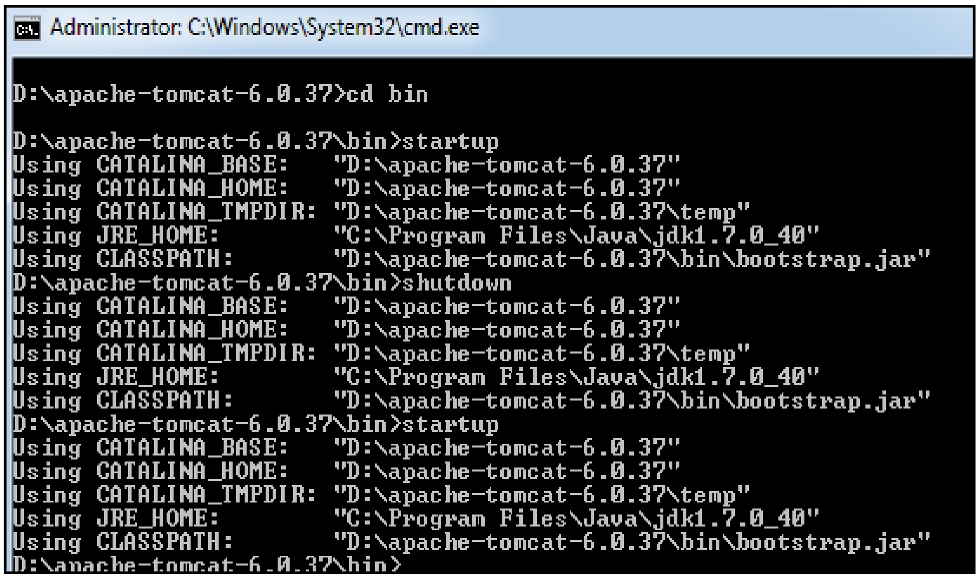
http://opendatakit.org/use/aggregate/tomcat-install/#Configure_for_Network_Access

HTTP/1.1. Connector Port: 8080

Internet-visible IP address or DNS name: admin-PC.mshome.net

Figure 4-2: Apache Tomcat configuration

- Server configuration through command prompt was performed as shown in Figure 4.2.



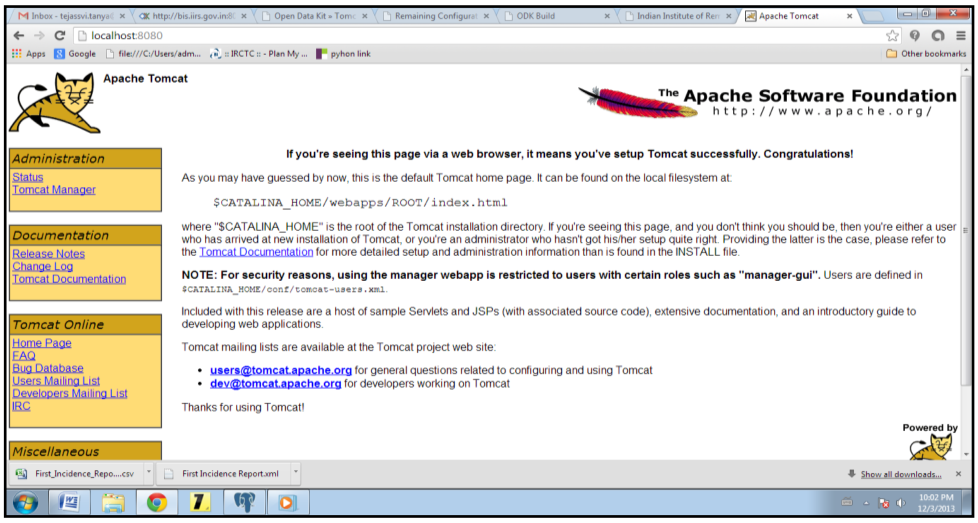
```
Administrator: C:\Windows\System32\cmd.exe

D:\apache-tomcat-6.0.37>cd bin

D:\apache-tomcat-6.0.37\bin>startup
Using CATALINA_BASE:   "D:\apache-tomcat-6.0.37"
Using CATALINA_HOME:   "D:\apache-tomcat-6.0.37"
Using CATALINA_TMPDIR: "D:\apache-tomcat-6.0.37\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.7.0_40"
Using CLASSPATH:       "D:\apache-tomcat-6.0.37\bin\bootstrap.jar"
D:\apache-tomcat-6.0.37\bin>shutdown
Using CATALINA_BASE:   "D:\apache-tomcat-6.0.37"
Using CATALINA_HOME:   "D:\apache-tomcat-6.0.37"
Using CATALINA_TMPDIR: "D:\apache-tomcat-6.0.37\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.7.0_40"
Using CLASSPATH:       "D:\apache-tomcat-6.0.37\bin\bootstrap.jar"
D:\apache-tomcat-6.0.37\bin>startup
Using CATALINA_BASE:   "D:\apache-tomcat-6.0.37"
Using CATALINA_HOME:   "D:\apache-tomcat-6.0.37"
Using CATALINA_TMPDIR: "D:\apache-tomcat-6.0.37\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.7.0_40"
Using CLASSPATH:       "D:\apache-tomcat-6.0.37\bin\bootstrap.jar"
D:\apache-tomcat-6.0.37\bin>
```

Figure 4-3: Server start-up

- Server was configured and checked by opening local host in browser with defined port number as shown in Figure 4.3.



If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

```
$CATALINA_HOME/webapps/ROOT/index.html
```

where "$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Documentation for more detailed setup and administration information than is found in the INSTALL file.

NOTE: For security reasons, using the manager webapp is restricted to users with certain roles such as "manager-gui". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- **users@tomcat.apache.org** for general questions related to configuring and using Tomcat
- **dev@tomcat.apache.org** for developers working on Tomcat

Thanks for using Tomcat!

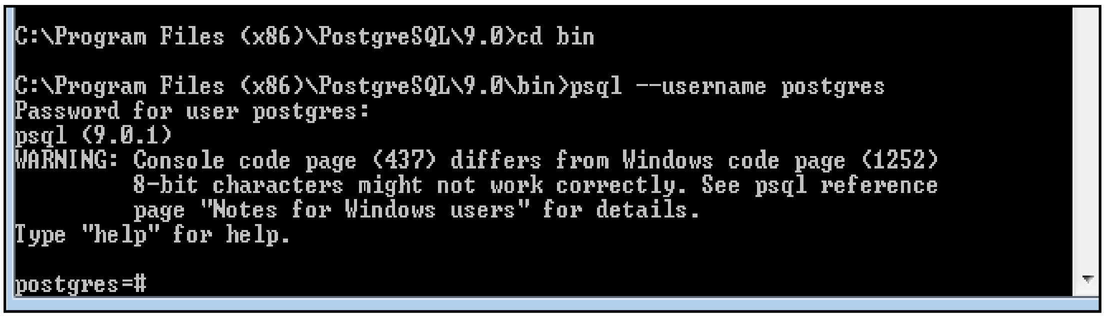
Figure 4-4: Checking installed Server

4.2.2. Integration with database

- PostGreSql was installed and configured. The version used for this project was PostGreSql 9.0 and pgAdmin III 1.16
- Windows ODK Aggregate Installer was downloaded from [36] and installed.
- While installing Aggregate it asked for postgres details and details of the new database to be created. For the purpose of the project the details are as below

Database name: RescueApp
User: rescue
Password: rescue

Once installed, a readme document opened up in browser with steps to be followed and executed on Postgres SQL shell (psql) commandline client as shown in Figure 4.4.



```
C:\Program Files (x86)\PostgreSQL\9.0>cd bin

C:\Program Files (x86)\PostgreSQL\9.0\bin>psql --username postgres
Password for user postgres:
psql (9.0.1)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=#
```

Figure 4-5: Configuring PostgreSql

- This created an ODKAggregate.war file which was deployed to Apache Tomcat web server. The IIRS server installed used the url location [37].A Google account for verification was required for first time configuration of ODKAggregate and administration of server. After successful login through Google account, an admin login was created for management and administration of local server.

4.2.3. Building report forms

Reporting forms are XML documents created at server. The XML Form was designed using XML editors. ODK supports various data types for form creation like text, numeric, date, time, location, media (image, audio and video) and selection (list, group). The forms can also be created using ODK Build application available online [38]. The form once designed were downloaded as xml and uploaded to Aggregate Server.
The report forms created for this research project are First Incident Report (FIR) and Summary Report. The FIR means incident reporting by rescue workers as and when they first visit and incident site. As soon as the rescue workers arrive at the incident site, they need to communicate information to the analysts to help plan the disaster management operations. Summary report is a more detailed report which includes details on the disaster impact and relief supplies required.
Table 4.1 shows the First Incident Report template form. The Summary Report consists of form elements of FIR with additional list of resource requirements like relief material, food packets, medical kits and water bottles.

Table 4.1: FIR Form Elements

Form Element	Description	Data Type
Reporting Person	Name of the reporting person	Text
Incident Site	Name of the incident site	Text
Date/Time	Date and time of reporting	System generated
Location	Exact location on map	GPS co-ordinates
Image	Image of the incident site	Media file
Video	Video of the incident site	Media file
People Rescued	Number of people rescued	Number
People died	Number of deaths	Number
People with casualties	Number of people with injuries or serious medical condition	Number
Other information	Any relevant information	Text

4.2.4. Collecting data and uploading to server

- ODK Collect source code was downloaded and imported in Package explorer in Eclipse. This was enabled as library by selecting IsLibrary from Android properties.[32]
- From RescueApp project properties, ODKCollect was added as library. This merged ODK Collect with RescueApp Project. ODK Collect was customized and used for the Reporting feature of the RescueApp along with RescueApp's other functionality.
- The server address was embedded in the mobile app itself to make it more user-friendly.

4.3. Emergency/ Distress Call

During disaster management, on the spot communication is important. Sometimes an emergency call needs to be made to communicate a request and also the location of the caller is important. The Emergency/distress call feature in RescueApp is in the form of an SOS button. By using SOS button, a call is made to a pre-saved number which can be re-defined. Simultaneously an SMS is sent to the same number. This SMS includes a message asking for help at the user's location. Fundamental techniques used for this function are:

4.3.1. Retrieving location in form of latitude and longitude

The internal GPS of the device has been utilized to retrieve the current location of the user. GPS gets signals through a constellation of 24 satellites. Through coding the application realized the retrieval of the location of the device in the form of latitude and longitude. These values were converted to a text format and displayed on the screen along with accuracy of the coordinates. The accuracy aspect was extremely important while using the application for Reporting purposes. To use location services in an application, an instance of the Location Services class i.e Location Client was used. Firstly Location Manager and Location Listener were instantiated.

```
private LocationManager locationManager=null;
private LocationListener locationListener=null;

locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000, 10,locationListener);
```

These classes executed the onLocationChanged method that returned a location object. This object contains location information.

```
/*----------Listener class to get coordinates ------------- */
public class MyLocationListener implements LocationListener
{
    @Override
    public void onLocationChanged(Location loc)
    {
            latitude=String.valueOf(loc.getLatitude());
            longitude=String.valueOf(loc.getLongitude());
            accuracy=String.valueOf(loc.getAccuracy());


    }


  }
```

4.3.2. Using Geocoder to retrieve address from latitude and longitude

Reverse Geocoding i.e. the process of retrieving the address from latitude and longitude values. This process was used in the application. The application was programmed to fetch the latitude and longitude values and then retrieve the address corresponding to those values to be displayed to the user.

As shown below, the Geocoder retrieved a list of addresses for the parameters: latitude and longitude. This list was converted to type string and address was retrieved for the location.

```
Geocoder geocoder = new Geocoder(getBaseContext(), Locale.getDefault());

 // Changed the Max results to 10--to check
  try
  {
      List<Address> list = geocoder.getFromLocation(latitude,longitude,10);

      if(list != null & list.size() > 0)
      {
          Log.v("tanya", list.toString());
          Address address=list.get(0);
          //Retrieving city from address
          city=address.getLocality()+ "," + address.getSubLocality();
```

Once the SOS button is clicked, it performs the below two actions. It places a call to a saved number and simultaneously sends an SMS with location details.

```
 //Makes a call to the saved number
 startActivity(new Intent(Intent.ACTION_CALL, Uri.parse("tel:"+ numberTextPref)));
//Initiate SmsManager to send SMS
SmsManager sms = SmsManager.getDefault();

 sms.sendTextMessage(numberTextPref, null, "HELP! I am at "·
+strReturnedAddress.toString()+". My co-ordinates are "+lat+", "+lon, null, null);
```

4.4. Disaster Alert

At the time of disaster, once the rescue and relief activities have begun, it is important to know current weather and weather alerts. Disaster alerts for the project have been chosen to be weather-related. The Weather tab on the main screen depicts disaster alert functionality. It shows the weather details for user's current location. Fundamental technique used for this functionality is:

4.4.1. Retrieving xml data from RSS feeds

Firstly a connection is created to the URL through HTTP

```
url = new URL("http://www.skymetweather.com/content/stories/weather-news-and-analysis/feed/");
HttpURLConnection con=(HttpURLConnection)url.openConnection();
con.setRequestMethod("GET");
con.setDoInput(true);
con.connect();
InputStream is=con.getInputStream();
```

Parser was created to read the XML data. This data is output to an array and retrieved as string

```
//PRINT INPUTSTREAM IN CONSOLE
BufferedReader br= new BufferedReader(new InputStreamReader(is));
StringBuilder sb = new StringBuilder();
String line;
while ((line = br.readLine()) != null) {
sb.append(line);
```

Various data sources in the form of API's are used. For weather information an API key is used to retrieve weather information from [39]. Geocoder is again used to retrieve city or locality from user's current location and the result is passed as a parameter to the weather URL. It then retrieves the weather details for user's location. For weather feeds another URL is used. It can be retrieved from [40].

Satellite imagery for weather visualization is Kalpana-1 of INSAT series of satellites of ISRO. The image resolution of Kalpana-1 satellite image is 8km. It uses VHR Sensor, Mercator projection and a cloud cover visible channel. The image has been retrieved from [35]. Google Maps API with weather layer are also used to show local and nearby weather conditions.

```
<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=true&libraries=weather">

var weatherLayer = new google.maps.weather.WeatherLayer({

    temperatureUnits: google.maps.weather.TemperatureUnit.CELCIUS

});
```

4.5. Geo-visualization System

Geo-visualization in simple terms means to form a mental image of some information and associating it with earth location. This function cannot be depicted on its own and needs some aspect which uses it for displaying information. Hence geo-visualization has been used for displaying current location and weather information. The assessment function also uses geo-visualization techniques to display data.

Fundamental techniques used for this function are:

4.5.1. Using open layers as an HTML JavaScript feature

```
map = new OpenLayers.Map({
    div: 'map',
    projection: 'EPSG:900913',
    numZoomLevels: 18,
    controls: [
        new OpenLayers.Control.TouchNavigation({
            dragPanOptions: {
                enableKinetic: true
            }
        }),
        new OpenLayers.Control.Zoom(),
        toolbar
    ],
    layers: [osm, vector],
    center: new OpenLayers.LonLat(0, 0),
    zoom: 1,
    theme: null
});
```

4.5.2. Retrieving location and nearby hospitals on Google Maps

This code retrieves the latitude and longitude of the user's location and loads Google map for the location by passing it as a parameter. It then gives a list from which the user can choose radius. It searches for hospitals within that radius and displays to the user. Geo-visualization is also depicted by weather function as explained above. It is possible to locate and display various amenities near user's current location using Google maps API. For the sake of demonstration, hospital has been used. A Google places documentation [41] lists out the various possible types of places that can be searched for, using the API.

```
 url = new URL("https://maps.googleapis.com/maps/api/place/nearbysearch/json?location="

+String.valueOf(myLocation.getLatitude())+","+String.valueOf(myLocation.getLongitude())+

"&radius="+radius+"&types=hospital&sensor=true&key=AIzaSyBky6v3qgMwaoo-Jf2WF4pH09kXGF8aROY");
```

4.6. Testing and evaluation of the application

The testing and evaluation of the software application with major importance to accuracy and time consumed is shown in Table 4.1 while Table 4.2 shows time consumed by various type of information to load and display on screen. This has been checked for in presence of 3G, 2G, Wi-Fi networks and also when there is no network access. The testing has been performed using Sony Xperia neo L with operating system Android v4.0.4 (Ice Cream Sandwich). It has an internal memory of 1 GB and 512 MB of RAM. The dimensions of the device are 121 x 61.1 x 12.2 mm (4.76 x 2.41 x 0.48 in). The app can be used on other versions of Android as well. This can be managed by making changes to the minimum and target Sdk version in code. The current min Sdk Version is 14 and target Sdk version is 19.

```
<uses-sdk

    android:minSdkVersion="14"

    android:targetSdkVersion="19" />
```

The total memory storage used by the app is around 14 MB. Internal storage memory consumed by RescueApp is 9.84 MB and external storage of 4 MB. The reporting feature needs storage to store report forms. This can exceed when reports are filled up and saved to be sent later. This is because the report forms allow uploading images and video recordings of the incident site, and these media files occupy storage on SD Card.

Table 4.2: Testing and Evaluation of RescueApp

	Location Accuracy		Time Taken		Battery
Test Conditions	**GPS**	**Network**	**GPS**	**Internet (Maps loading, Information Display)**	
NETWORK					
3G	Up to 5m	Up to 30m	10 seconds	8-10 seconds	Increased consumption when used with 3G
2G	Up to 5m	Up to 39 m	18 seconds	10-40 seconds	Normal Battery use
Wi-Fi	Up to 5m	Up to 22 m	4 seconds	3-7 seconds	Lesser Battery consumption
No Signal	Up to 6m	Nil	26 seconds	10 seconds	Lesser Battery consumption
FIELD TEST					
Dense Forest	Less Accuracy and sometimes no signal		20-30 seconds		Battery use independent of field test conditions
Urban Area	5m -29m		10-20 seconds		-do-
Cloud Cover	Up to 5m		10 seconds		-do-
Humidity	Up to 5m		10 seconds		-do-
Hilly Terrain	5m-30m		10-20 seconds		-do-

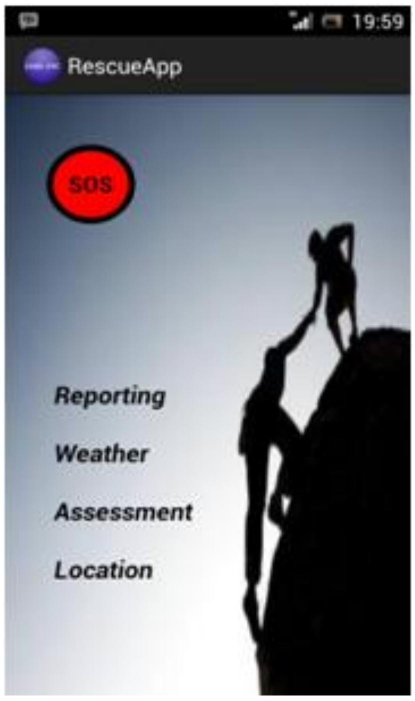
Table 4.3: Time variation with network coverage

	3G	2G	Wi-Fi	No Signal
Weather Information	8 seconds	10 seconds	5 seconds	Nil
Weather feeds	10 seconds	20 seconds	4 seconds	Nil
Weather on Maps	8 seconds	20 seconds	7 seconds	Nil
Satellite Image	7 seconds	30 seconds	5 seconds	Nil
Location on Maps	5-10 seconds	4-10 seconds	5 seconds	10 seconds

5. RESULTS AND DISCUSSION

The research involved understanding requirements by personnel involved in rescue operations and subsequently, development of application. The App has been named as **RescueApp** as it addresses the communication requirements of the rescue workers for disaster management. The disaster management rescue operations begin as soon as the rescue teams are mobilised. Evacuation of people and moving them to relief camps or safety zones is given prime importance. Quick medical help is provided to those in need. Communication and information retrieval comes next. This is where the application RescueApp will prove useful. The team officer who coordinates for both rescue and relief operations is the target user of this mobile (client) application. The information collected by team officer is sent to the Relief Commissioner, Chief Secretary or District Collector. These officials might further involve a third person as the analyst. That analyst is the target end user for the server application.

The main screen of RescueApp shows some features in the form of buttons that represent the proposed functionality. Each button click takes the user to a separate screen with options to choose from.



SOS feature is used for Emergency Call

Reporting feature is for field data collection and reporting

Weather feature is for retrieving weather information

Assessment allows user to point locations and mark areas on the map to make an assessment about the disaster.

Location feature is to know, view and send current location as SMS

Figure 5-1: RescueApp

Figure 5.1 shows the main screen of RescueApp. Each of these buttons depict the features of the App. The App uses a logo to represent it in the mobile device. The app makes use of GPS, network/ internet access to retrieve and send information. The functionality as retrieved from requirements in Section 3.1.3 has a feature catering to its function in the application. Each of these features is explained in the coming sections along with Case Scenarios.

5.1. SOS Button – Emergency/ Distress Call

Case Scenario: Suppose the user is in need of urgent medical help. This needs to be communicated as soon as possible with the dedicated helpline numbers. Such numbers are made available at the time of disaster. Emergency Call allows the user to quickly place a call without wasting time in searching the contacts list and dialling some number. The user can request for help. Figure 5.3 shows the Emergency Call feature. By the time the user ends the call, an SMS will have been sent to the emergency number also.

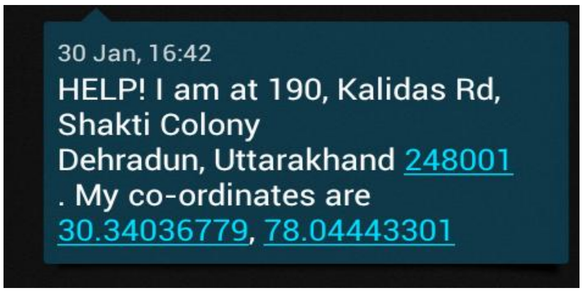

30 Jan, 16:42
HELP! I am at 190, Kalidas Rd, Shakti Colony
Dehradun, Uttarakhand 248001
. My co-ordinates are
30.34036779, 78.04443301

Figure 5-2: SMS with user's location

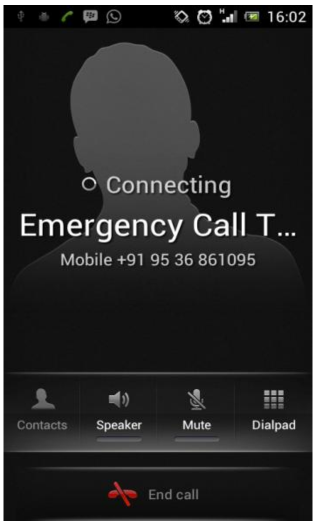
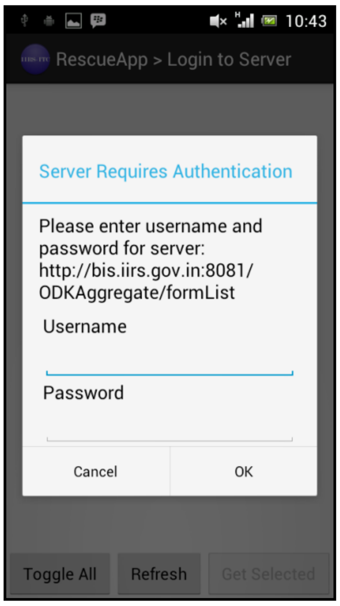


Figure 5-3: Emergency Call

The exact location of the caller is important to know. In an emergency situation, the user cannot dictate full address or landmark and chances are that the user might not even know his/her exact location. Geo-location sent in the form GPS co-ordinates can help pin point the location on map and know the address and nearby landmarks. If address is retrieved and sent in SMS, it is an added advantage and will help save time. Figure 5.2 shows the contents of the SMS that is sent by use of the SOS button. The SMS consists of a message, address and location co-ordinates. It is sometimes possible that Geocoder does not return an address corresponding to the location; in such case only the location coordinates are sent.

5.2. Reporting

Case Scenario: Suppose the rescue team officer needs to communicate First Incident Report (explained in Section 4.2.3). By applying the reporting button, the app displays a screen with a menu layout (as shown in Figure 5.5). The user needs to login to server to download the report templates (as shown in Figure 5.4). Only authorized user can download the report template and the login will be valid till the session time expires (120 seconds). This implies that the user can download the report templates once for all, while the network access is available. The process of filling out reports does not require cell network or internet connectivity. It uses the mobile device's own resources like device memory, GPS and camera.


10:43
RescueApp > Login to Server

Server Requires Authentication

Please enter username and password for server: http://bis.iirs.gov.in:8081/ODKAggregate/formList

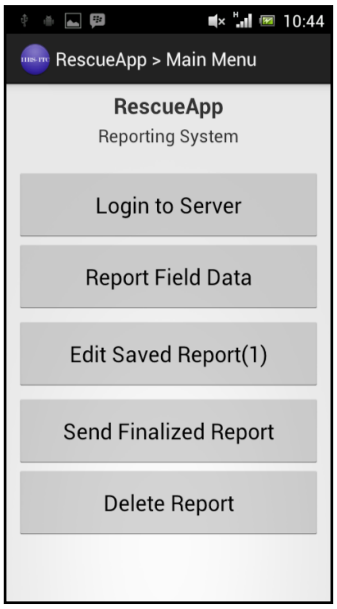
Username

Password

Cancel OK

Toggle All Refresh Get Selected

Figure 5-4: Authentication to Server

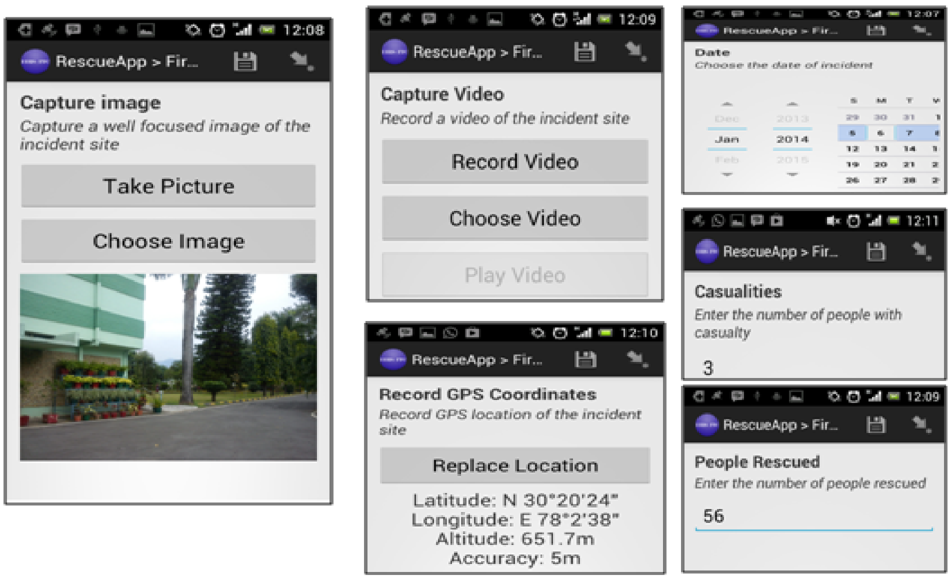

10:44
RescueApp > Main Menu

RescueApp
Reporting System

Login to Server
Report Field Data
Edit Saved Report(1)
Send Finalized Report
Delete Report

Figure 5-5: Reporting Menu

The Reporting feature allows user to upload images, location co-ordinate, video, textual and numeric information, date and time of reporting. Figure 5.6 shows the report elements as per First Incident Report.


RescueApp > Fir...

Capture image
Capture a well focused image of the incident site
Take Picture
Choose Image

Capture Video
Record a video of the incident site
Record Video
Choose Video
Play Video

Record GPS Coordinates
Record GPS location of the incident site
Replace Location
Latitude: N 30°20'24"
Longitude: E 78°2'38"
Altitude: 651.7m
Accuracy: 5m

Date
Choose the date of incident
Jan 2014

Casualities
Enter the number of people with casualty
3

People Rescued
Enter the number of people rescued
56

Figure 5-6: Reporting options

The data once collected can be saved in Report. The user can create many such reports and upload them to the server. Once uploaded, the analyst can view and analyze this data from the database server. The data reported from mobile application to server system is in the form of an XML document. The data is submitted through an HTTP request to server and the server application parses the XML content to database with proper validation of the data types. The conversion of XML to SQL allows the analyst to query data, perform analysis and create reports on this submitted information. Spatial analysis can be performed on this information for example one can determine the shortest route to nearest airport. Visualization of this submitted information will enable the analyst to view the location on map along with the tagged image, video and other information. Figure 5.7 shows a satellite view with the point location. Once clicked on this point, it displays the geo-tagged information recorded for this location as in Figure 5.8. The reported information is updated into PostgreSql database as shown in Figure 5.9.

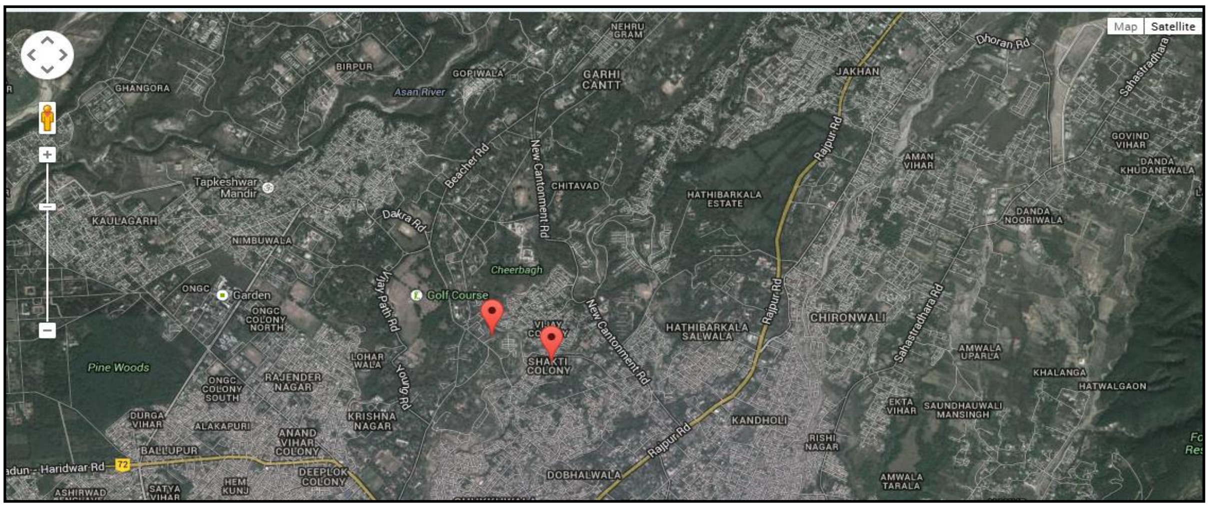
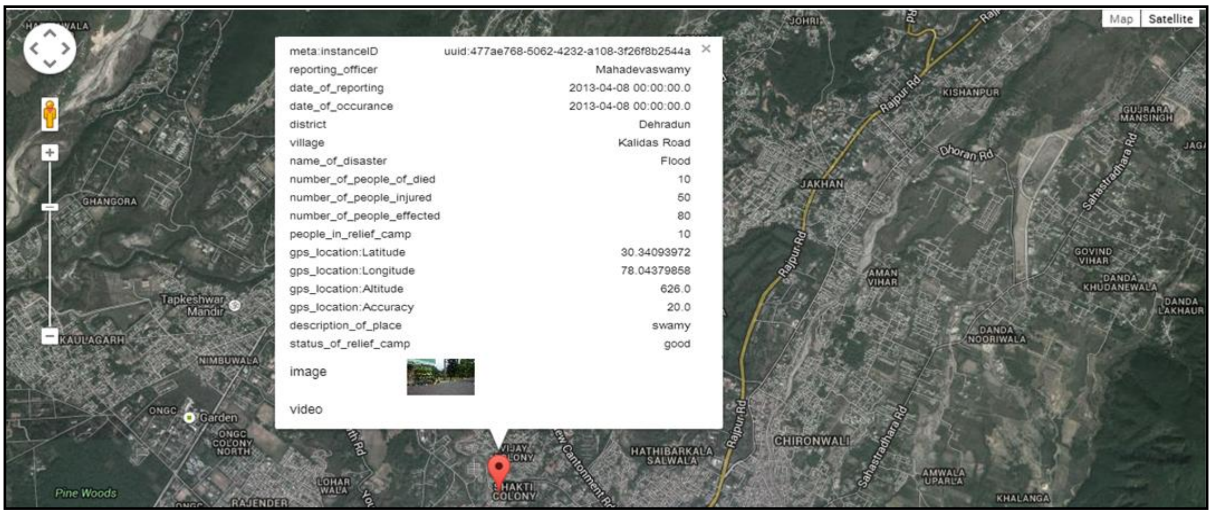
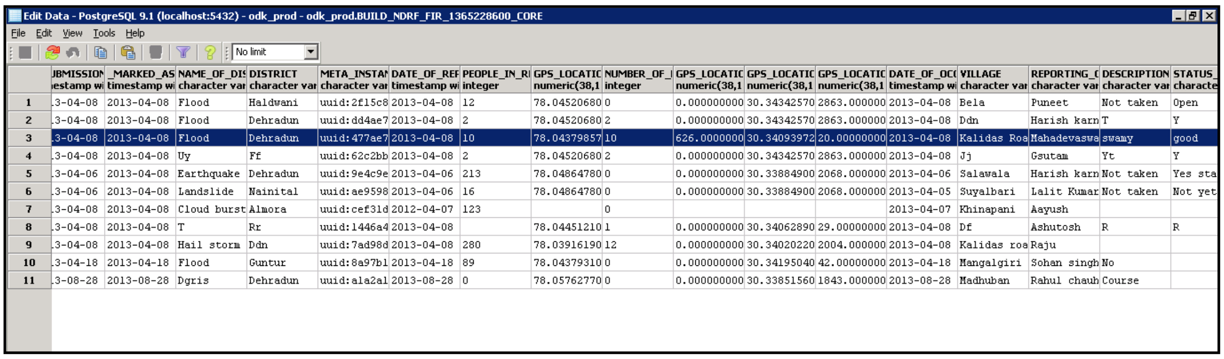


Figure 5-7: Satellite view with point location



meta:instanceID	uuid:477ae768-5062-4232-a108-3f26f8b2544a
reporting_officer	Mahadevaswamy
date_of_reporting	2013-04-08 00:00:00.0
date_of_occurance	2013-04-08 00:00:00.0
district	Dehradun
village	Kalidas Road
name_of_disaster	Flood
number_of_people_of_died	10
number_of_people_injured	50
number_of_people_effected	80
people_in_relief_camp	10
gps_location:Latitude	30.34093972
gps_location:Longitude	78.04379858
gps_location:Altitude	626.0
gps_location:Accuracy	20.0
description_of_place	swamy
status_of_relief_camp	good
image	
video	

Figure 5-8: Reported information on click event



Edit Data - PostgreSQL 9.1 (localhost:5432) - odk_prod - odk_prod.BUILD_NDRF_FIR_1365228600_CORE

File Edit View Tools Help

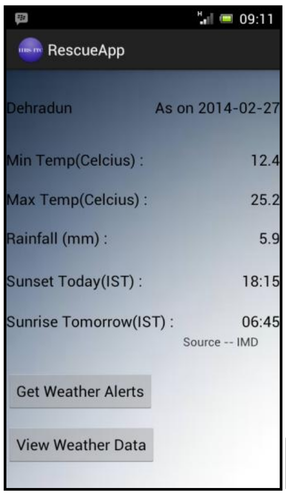
No limit

	JBMISSION iestamp wi	_MARKED_AS timestamp wi	NAME_OF_DIS character var	DISTRICT character var	META_INSTAN character var	DATE_OF_REF timestamp wi	PEOPLE_IN_RI integer	GPS_LOCATIC numeric(38,1	NUMBER_OF_I integer	GPS_LOCATIC numeric(38,1	GPS_LOCATIC numeric(38,1	GPS_LOCATIC numeric(38,1	DATE_OF_OC(timestamp wi	VILLAGE character var	REPORTING_(character var	DESCRIPTION character var	STATUS_ characte
1	.3-04-08	2013-04-08	Flood	Haldwani	uuid:2f15c8	2013-04-08	12	78.04520680	0	0.000000000	30.34342570	2863.000000	2013-04-08	Bela	Puneet	Not taken	Open
2	.3-04-08	2013-04-08	Flood	Dehradun	uuid:dd4ae7	2013-04-08	2	78.04520680	2	0.000000000	30.34342570	2863.000000	2013-04-08	Ddn	Harish karn	T	Y
3	.3-04-08	2013-04-08	Flood	Dehradun	uuid:477ae7	2013-04-08	10	78.04379857	10	626.0000000	30.34093972	20.00000000	2013-04-08	Kalidas Roa	Mahadevaswa	swamy	good
4	.3-04-08	2013-04-08	Uy	Ff	uuid:62c2bb	2013-04-08	2	78.04520680	2	0.000000000	30.34342570	2863.000000	2013-04-08	Jj	Gsutam	Yt	Y
5	.3-04-06	2013-04-08	Earthquake	Dehradun	uuid:9e4c9e	2013-04-06	213	78.04864780	0	0.000000000	30.33884900	2068.000000	2013-04-06	Salawala	Harish karn	Not taken	Yes sta
6	.3-04-06	2013-04-08	Landslide	Nainital	uuid:ae9598	2013-04-06	16	78.04864780	0	0.000000000	30.33884900	2068.000000	2013-04-05	Suyalbari	Lalit Kumar	Not taken	Not yet
7	.3-04-08	2013-04-08	Cloud burst	Almora	uuid:cef31d	2012-04-07	123		0				2013-04-07	Khinapani	Aayush		
8	.3-04-08	2013-04-08	T	Rr	uuid:1446a4	2013-04-08		78.04451210	1	0.000000000	30.34062890	29.00000000	2013-04-08	Df	Ashutosh	R	R
9	.3-04-08	2013-04-08	Hail storm	Ddn	uuid:7ad98d	2013-04-08	280	78.03916190	12	0.000000000	30.34020220	2004.000000	2013-04-08	Kalidas roa	Raju		
10	.3-04-18	2013-04-18	Flood	Guntur	uuid:8a97b1	2013-04-18	89	78.04379310	0	0.000000000	30.34195040	42.00000000	2013-04-18	Mangalgiri	Sohan singh	No	
11	.3-08-28	2013-08-28	Dgris	Dehradun	uuid:a1a2a1	2013-08-28	0	78.05762770	0	0.000000000	30.33851560	1843.000000	2013-08-28	Madhuban	Rahul chauh	Course	

Figure 5-9: Reported information updated in database

5.3. Weather –Disaster Alerts and Geo-visualization

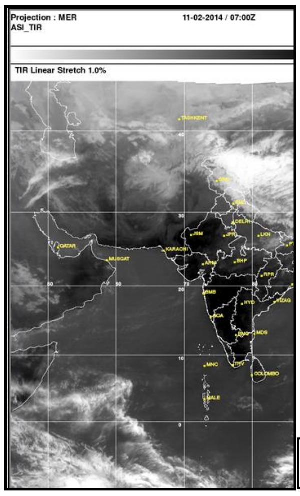
Case Scenario: Suppose the team officer needs to take some decisions based on the weather conditions. For example the relief camp is set up at a particular location and needs to be moved to another location because the weather predicts extreme conditions. By using the Weather button, the app shows a screen with weather information of the user's current location as shown in Figure 5.10 (a). This can take a few seconds to load since it first retrieves location and then fetches the weather information for that particular location. It displays the information in text format. It shows minimum and maximum temperature, rainfall if recorded, sunset and sunrise timings.


09:11
RescueApp

Dehradun	As on 2014-02-27
Min Temp(Celcius) :	12.4
Max Temp(Celcius) :	25.2
Rainfall (mm) :	5.9
Sunset Today(IST) :	18:15
Sunrise Tomorrow(IST) :	06:45

Source -- IMD

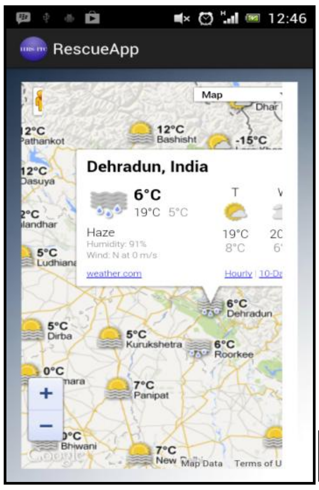
a



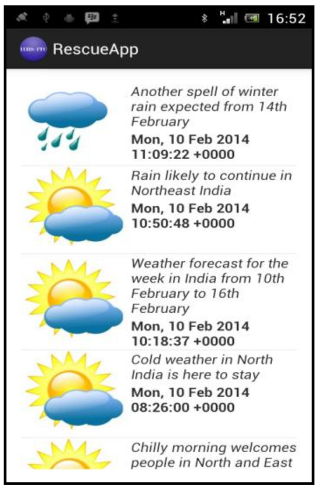
b

Figure 5-10: (a) Weather data for user's location; (b) Satellite Imagery for weather

Other options include visualization of weather data on map or satellite imagery. Weather data can be using a satellite cloud cover image (as shown in Figure 5.10(b)) and local level through weather layer overlaid on Google maps (as shown in Figure 5.11(a)). It also gives a three day forecast. The Figure 5.11(b) shows weather feeds that are updated daily. These weather feeds are themselves analytical reports.



a


16:52
RescueApp

Another spell of winter rain expected from 14th February
Mon, 10 Feb 2014 11:09:22 +0000

Rain likely to continue in Northeast India
Mon, 10 Feb 2014 10:50:48 +0000

Weather forecast for the week in India from 10th February to 16th February
Mon, 10 Feb 2014 10:18:37 +0000

Cold weather in North India is here to stay
Mon, 10 Feb 2014 08:26:00 +0000

Chilly morning welcomes people in North and East

b

Figure 5-11: (a) Map view for weather; (b) Weather Feeds

5.4. Assessment – Geo-visualization

Case Scenario: Suppose the team officer wants to visualize the extent of disaster impact or point out affected locations on map. The Assessment button shows a screen that loads Open Street Maps along with tools shown on left upper corner of the screen. With these tools the user can zoom in and mark areas and point out locations on the map. Open layers are in the form JavaScript libraries. The JavaScript source for this functionality is [42] . This information will be updated in database and the analyst can use this information for planning and analysis. Figure 5.12 (a) shows marking an unshaded region on map, (b) shows shaded region on map while (c) shows point locations that can be marked on the map.

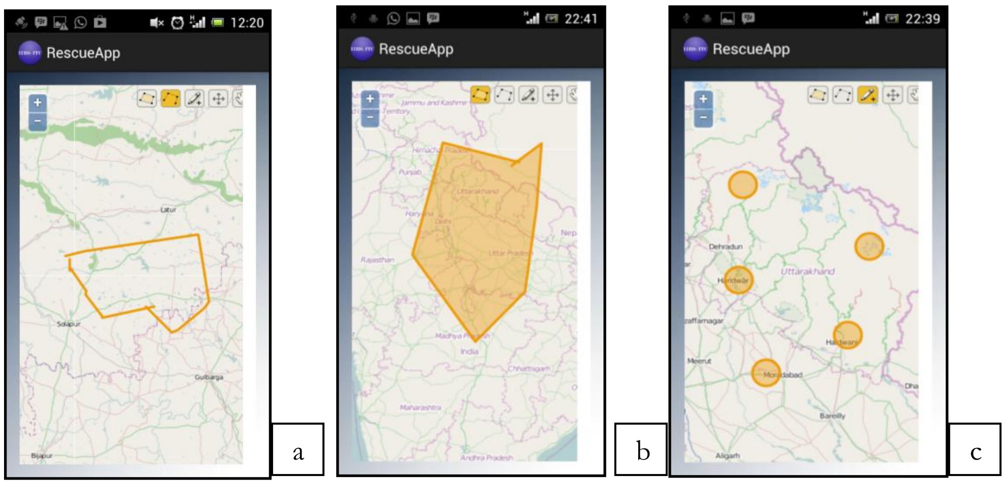
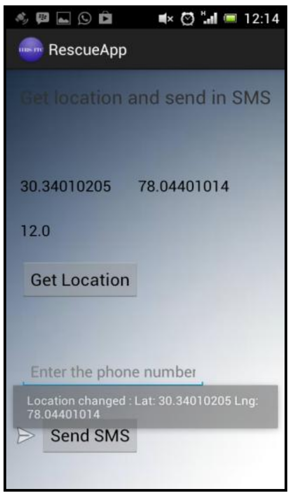


Figure 5-12: Assessment functionality (a) Unshaded area ; (b) Shaded area ; (c) Point locations

5.5. Location – Geo-visualization

Case Scenario: Suppose the team officer wants to know his/her current location and send that information to a number other than the emergency number possibly another rescue worker. Also the officer may want to know nearby amenities within a given distance from his/her current location.



12:14

RescueApp

Get location and send in SMS

30.34010205 78.04401014

12.0

Get Location

Enter the phone number

Location changed : Lat: 30.34010205 Lng: 78.04401014

Send SMS

Figure 5-13: Retrieve location and send in SMS

The Location feature provides two options. One is to view and send location information as in Figure 5.13. This allows the user to retrieve location coordinates and accuracy through the on-board GPS. This information can be sent as an SMS to a number that user can provide. The other option is to view location on map. This location shows up as a marker in a Google map window as shown in Figure 5.14. The user can choose a radius value (in meters) and the map will show nearby hospitals within the selected radius. The hospitals are shown as plus sign marker and provide information about the hospital when tapped on the icon. The application can be programmed to show other amenities as well but for the sake of demonstration, hospitals have been used.

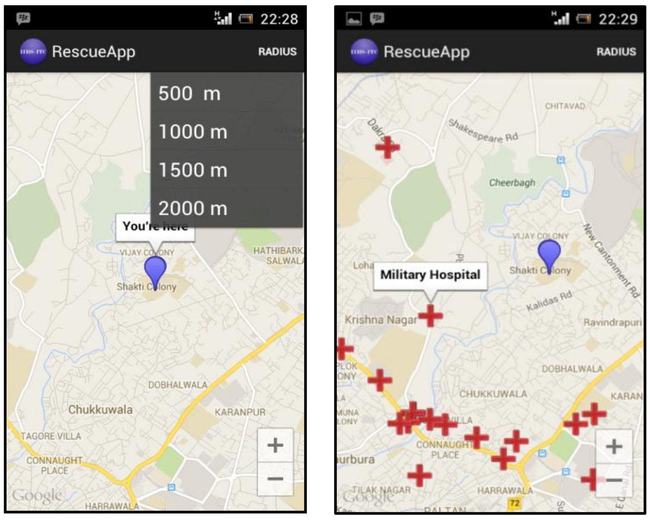


Figure 5-14: User's location and nearby hospitals

5.6. Testing Results

The testing results as displayed in Chapter 4 are discussed here. After performing the network testing, it was found that GPS exhibited same accuracy in presence and absence of Network but the time consumed to display GPS results varied. Using Wi-Fi, reduced GPS loading time since it itself gave an accuracy of up to 22 meters. Maps loading and information display was best with Wi-Fi and 3G network. The 2G network comparatively took more time to display maps and satellite imagery. As proven theoretically, cloud cover and humidity had no influence on GPS signals but dense forest greatly affected it. It was almost at the verge of getting no signal for a long time. Hilly terrain and urban area affected GPS signals in some areas as experienced while walking or travelling besides a trail of high mountains, standing near the entrance of a building and between clusters of buildings.

Network connectivity was affected in hilly areas and alternative available network had to be manually chosen. In absence of network, GPS gave the same accuracy but it took longer time tp display results as compared to the time taken in the presence of network. Maps could be loaded and displayed in absence of network as well. This was because the maps were used earlier and thereby stored in cache memory. They showed up only for the location they were earlier used.

The app consumed minimal memory and battery. While the App was running, the battery consumed was about 2% of the total hours remaining for use. The total memory used by the app was around 14 MB. Internal storage memory consumed by RescueApp was 9.84 MB and external storage of 4 MB.

For usability testing, the application was distributed to users in separate locations and asked for screenshots of the application. A small introduction about each feature helped the users understand and use the application with ease. The application gave weather and location details for different user locations. It was checked on different Android versions (4.0.4, 4.1.2, 4.3) as well. It gave some package parsing issues when checked with the latest Android version 4.4 but after updating the Android 4.4 operating system to the latest version, it worked well.

6. CONCLUSION AND RECOMMENDATIONS

The primary objective of the research was to build an Android application abiding by information and communication requirements envisaged by discussions with the disaster management teams. The interview conducted with an official involved in rescue and relief activities of disaster management helped gather useful information. The functionality proposed for the mobile application addresses the elicited requirements. Each function has a feature in the application dedicated to it. Also utmost care has been given to make the application as user-friendly as possible. Textual detail has been kept minimal on the screen but still each feature has been explained well. The application involves various data sources to make weather data and predictions available. Many Android apps exist for disaster management but RescueApp combines various features into one.

6.1. The Research questions

1. ***What is the essential functionality required in the mobile application?***

   Review of rescue operations, discussions and the interview conducted helped understand the scenarios and frame requirements. These requirements further helped to decide the functionality in the application. The essential functionality planned and developed in the mobile application is:

   - Reporting system
   - Emergency/Distress call
   - Disaster Alert
   - Geo-visualization system

2. ***Is the spatial functionality achievable through a simple interface?***

   Yes, the spatial functionality is achievable through a simple interface. The application offers all the proposed functionality through a simple and user-friendly interface. Effective display of maps and images is possible on a small handheld device.

3. ***What can be the criteria for application evaluation?***

   The criteria for application evaluation are categorized into four types. Software testing, Network Access testing, Field testing and Usability testing. Software testing checks the use of resources by the application. The network access testing checks the mobile application use in all available forms of network and also in absence of network. Field testing involved testing and evaluating the application in different kinds of situations. Lastly usability testing was performed by giving the app to random users and users found the interface easy to use and understand.

4. ***How to make the application serviceable in event of network failure?***

The application makes use of the in-built features in the mobile device. The camera and GPS are functional in absence of network and internet access. The Reporting function shows that the reports once downloaded from the server can be used for field data collection even in absence of network and the data can be sent once the network access resumes. Location information can be retrieved in absence of network since it needs only GPS that works independent of network access. Calling, Sending SMS, Retrieving Maps and Images for displays are features dependent on network access. However in some cases alternate means of network can be set up like satellite phone or connectivity through Wi-Fi/Bluetooth in computer device.

6.2. Recommendations

1. As of now the application depicts client-server architecture but only as a part of the Reporting system feature. This architecture can be extended to other functions as well. Also more features can be added through which the application user receives information from the analyst and vice-versa. Weather and location updates will also be sent from the application to the server. In case the analyst needs to inform the user in the field about any weather hazard related to current location, it will be possible.

2. The Assessment feature can be elaborated. The marked points and area on the map can be converted into WKT (Well known text) format and sent to the server. Also for better assessment, a tablet is recommended.

3. The location feature on map currently shows only hospitals near the user's location. This can be extended to show other amenities also.

4. The Emergency/ Distress call places a call to a pre-saved local number. But this can be changed to an existing emergency number like 911 or 100 which are usual helpline numbers made available at the time of disaster.

5. As of now, only weather has been considered for Disaster Alerts. But disaster alerts can be proximity alerts (if the user is in or near a danger zone).

6. The weather feeds as of now are retrieved from a source that receives updates from IMD in form of a batch daily at 4 PM. This needs to use an alternative source that provides more current data. Accurate weather API's are available for commercial purposes. Such API's can be used when the application is deployed by disaster management teams. Also GeoRSS feeds if available will make it easier to visualize weather conditions.

7. The maps can be prepared to be stored offline and used even in absence of network access.

REFERENCES

1. Coppola, D.P., *Introduction to international disaster management*. 2006: Butterworth-Heinemann.

2. National Disaster Management Authority, G.o.I., *National Disaster Management Guidelines- National Disaster Management Information and Communication System (NDMICS)*. 2012, NDMA Bhawan: National Disaster Management Authority, Govt of India.

3. Bhanumurthy, V., et al., *Emergency Management–A Geospatial Approach*. the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Beijing, 2008.

4. Gupta, A.K., S. Singh, and S.S. Nair, *Hydrometeorological Hazards in Uttarakhand India, Himalaya-Forensic Assessment of 2013 Flash Flood Disaster: Need of Integrated Planning for Sustainable Development*. International Journals of Geography and Environment Sciences, 2013. **Vol. 1(1)**.

5. Thayyen, R.J., et al., *Study of cloudburst and flash floods around Leh, India, during August 4–6, 2010*. Natural Hazards, 2013. **65**(3): pp. 2175-2204.

6. Manfré, L.A., et al., *An analysis of geospatial technologies for risk and natural disaster management*. ISPRS International Journal of Geo-Information, 2012. **1**(2):pp. 166-185.

7. Weng, Y.-H., F.-S. Sun, and J.D. Grigsby, *GeoTools: An android phone application in geology*. Computers & Geosciences, 2012. **44**(0): pp. 24-30.

8. Shu, X., Z. Du, and R. Chen. *Research on mobile location service design based on Android*. in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on*. 2009. IEEE.

9. Doner, F. and T. Yomralioglu, *Examination and comparison of mobile GIS technology for real time geo-data acquisition in the field*. Survey Review, 2008. **40**(309): pp. 221-234.

10. Kohli, S., et al., *Sahana Eden Essential Guide, Google Summer of Code Documentation Summit, Mountain View, USA, 18-20 October 2011, publisher: Lulu Enterprises*. 2011, Inc.

11. Karnatak, H.C., et al., *Spatial mashup technology and real time data integration in geo-web application using open source GIS–a case study for disaster management*. Geocarto International, 2012. **27**(6): pp. 499-514.

12. Kevany, M.J., *Geo-information for disaster management: lessons from 9/11*, in *Geo-information for disaster management*. 2005, Springer. pp. 443-464.

13. Tsai, M.-K. and N.-J. Yau, *Improving information access for emergency response in disasters*. Natural Hazards, 2013: pp. 1-12.

14. Jiang, B. and X. Yao, *Location-based services and GIS in perspective*. Computers, Environment and Urban Systems, 2006. **30**(6): pp. 712-725.

15. Singhal, M. and A. Shukla, *Implementation of Location based Services in Android using GPS and Web Services*. IJCSI International Journal of Computer Science Issues, 2012. **9**(1).

16. Tsou, M.-H. and C.-H. Sun, *Mobile GIServices applied to disaster management*. Dynamic and Mobile GIS: Investigating Changes in Space and Time, 2006.

17. Richter, S. and M. Hammitzsch. *Development of an Android App for notification and reporting of natural disaster such as earthquakes and tsunamis*. in *EGU General Assembly Conference Abstracts*. 2013.

18. Gómez, D., et al., *A Review on Mobile Applications for Citizen Emergency Management*, in *Highlights on Practical Applications of Agents and Multi-Agent Systems*. 2013, Springer. pp. 190-201.

19. Palmer, N., et al. *Raven: Using Smartphones For Collaborative Disaster Data Collection*. in *Under Submission To The Intl. Workshop on Information Systems for Crisis Response and Management (ISCRAM 2012)*. 2012.

20. Asif, M., et al., *A Web-based Disaster Management-Mitigation Framework Using Information and Communication Technologies and Open Source Software*. JU Journal of Information Technology, 2012. **Vol. 1**.

21. Fajardo, J.T.B. and C.M. Oppus, *A mobile disaster management system using the android technology*. WSEAS Transactions on Communications, 2010. **9**(6): pp. 343-353.

22. Gorbil, G. and E. Gelenbe. *Disruption Tolerant Communications for Large Scale Emergency Evacuation*. in *Proceedings of the 11th IEEE International Conference on Pervasive Computing and Communication, San Diego, CA, USA*. 2013.

23. Shao, Z., et al. *A Rapid and Reliable Disaster Emergency Mobile Communication System via Aerial Ad Hoc BS Networks*. in *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*. 2011. IEEE.

24. Eleiche, M.A. and B. Markus, *Network Analysis Methods for Mobile GIS*. 2011, University of West Hungary.

25. Farnham, S., E. Pedersen, and R. Kirkpatrick. *Observation of Katrina/Rita Groove deployment: Addressing social and communication challenges of ephemeral groups*. in *Proceedings of the 3rd International ISCRAM Conference*. 2006.

26. Srivastava, M.R.K., *Specific requirements of rescue teams*, M.T. Tejassvi, Editor. 2013.

27. Officer, C.R., *Daily Flood Situation Report*. 2013, Ministry of Home Affairs (Disaster Management Division).

28. India, S. *Flood Incident in Uttarakhand*. Available from: www.sphereindia.org.in [Accessed 16/09/2013]

29. Project, T.A.O.S. *Android Development*.Available from: http://developer.android.com/training/index.html.[Accessed 18/09/2013]

30. Project, A.O.S. *Get the Android SDK*. Available from: http://developer.android.com/sdk/index.html. [Accessed 18/09/2013]

31. Oracle. *Java Download*.Available from: http://www.java.com/.

[Accessed 18/09/2013]

32. Github. *Download Phonegap*. Available from: https://github.com/phonegap/phonegap/archive/master.zip. [Accessed 09/10/2013]

33. V.Retsios, *Installation of the Android Development Kit for developing a PhoneGap application*. 2013, ITC, Enschede, The Netherlands: 2nd SEMA Hackathon and Software Training, 26-27 October 2013, Dar es Salaam, Tanzania.

34. Borriello, G., *Open data kit: creating an open source community for mobile data collection*, in *Proceedings of the 3rd ACM international workshop on MobiArch*. 2011, ACM: Bethesda, Maryland, USA. p. 1-2.

35. Kit, O.D. *Tomcat Install*. Available from: http://opendatakit.org/use/aggregate/tomcat-install/. [Accessed 23/11/2013]

36. Kit, O.D. *Downloads*. Available from: http://opendatakit.org/downloads/.[Accessed 28/11/2013]

37. Kit, O.D. *ODKAggregate Server*. Available from: http://bis.iirs.gov.in:8080/ODKAggregate/. [Accessed 28/11/2013]

38. Kit, O.D. *ODK Build*. Available from: http://build.opendatakit.org/.[Accessed 29/11/2013]

39. DataWeave.in. *Weather api*. Available from: http://api.dataweave.in/v1/indian_weather/findByCity/?api_key=c9a17a31ac8c6aebf6edf52374fe9e7ca956b264&city=+city. [Accessed 24/01/2014]

40. skymetweather.com. *Weather feeds and analysis*. Available from: http://www.skymetweather.com/content/stories/weather-news-and-analysis/feed/. [Accessed 23/01/2014];

41. GoogleDevelopers. *Google Places API*. 2013; Available from: https://developers.google.com/places/documentation/supported_types.[Accessed 25/01/2013]

42. OpenLayers. *OpenLayers development examples for mobile*.Available from: http://openlayers.org/dev/OpenLayers.mobile.js. [Accessed 23/11/2013]

APPENDIX

RescueApp:Java Code

MainActivity.java

```
package com.tanya.rescueapp;

import java.io.IOException;
import java.util.List;
import java.util.Locale;
import org.odk.collect.android.activities.SplashScreenActivity;
import com.tanya.rescueapp.R;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.provider.Settings;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity implements LocationListener
{
	public String numberTextPref = null;
	public boolean enableTextPref = true;
	public String textMessagePref = null;
	public double latitude;
	public double longitude;
	public String lat=null;
	public String lon=null;
	public String city=null;
	public LocationListener locationListener=null;
	public LocationManager locationManager=null;
	@Override
	protected void onCreate(Bundle savedInstanceState)
	{
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		// A pop up message to request enabling of GPS
```

```
    //Toast.makeText(getBaseContext(),"Please enable
    GPS",Toast.LENGTH_SHORT).show();
        //Declare Shared Preferences for emergency call
        SharedPreferences  prefs =
        PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        numberTextPref = prefs.getString("numberfield", "+919634082137");
        enableTextPref = prefs.getBoolean("enabletextmessage", true);
        textMessagePref  =  prefs.getString("textmessagefield",  "I  need  Help  at"+"Latitude:
        "+lat+"Longitude: "+lon+city);
        //Declare Button variables
        Button weather1=(Button) findViewById(R.id.weather1);
        Button assessment1= (Button)findViewById(R.id.assessment1);
        Button reporting1= (Button)findViewById(R.id.reporting1);
        Button location1= (Button)findViewById(R.id.location1);
        Button SOS= (Button) findViewById(R.id.SOS);

//Create on click event for buttons

// Weather Info
weather1.setOnClickListener (new View.OnClickListener()
{
        public void onClick(View v)
        {
                  //Opening a web page through button click in another window
                //Intent browserIntent= new Intent
                (Intent.ACTION_VIEW,Uri.parse("http://www.imd.gov.in/section/nhac/dyna
                mic/nhacsatimg.htm"));
                //startActivity(browserIntent);
                Intent nextScreen = new Intent(MainActivity.this, WeatherActivity.class);
                startActivity(nextScreen);
        }
});

// Assessment
assessment1.setOnClickListener (new View.OnClickListener()
{
        public void onClick(View v)
        {
                Intent nextScreen = new Intent(MainActivity.this, SecondActivity.class);
                 startActivity(nextScreen);

        }
});

//Reporting Data
reporting1.setOnClickListener (new View.OnClickListener()
{
        public void onClick(View v)
        {

                Intent intent= new Intent(MainActivity.this, SplashScreenActivity.class);
                startActivity(intent);
        }
});
```

```
        //Get Location
    location1.setOnClickListener (new View.OnClickListener()
        {
                public void onClick(View v)
                {
                        Intent nextScreen = new Intent(MainActivity.this, FourthActivity.class);
                        startActivity(nextScreen);
                }
        });

        //Emergency Call
                SOS.setOnClickListener (new View.OnClickListener()
                {
                        public void onClick(View v)
                        {
                                Toast.makeText(MainActivity.this, "Your location will be sent as an
                                SMS", Toast.LENGTH_SHORT).show();
                                dialEmergency();
                        }
                });
        }

        @Override
        public void onResume()
        {
                SharedPreferences prefs =
                PreferenceManager.getDefaultSharedPreferences(getBaseContext());
                numberTextPref = prefs.getString("numberfield", "+919634082137");
                enableTextPref = prefs.getBoolean("enabletextmessage", true);
                textMessagePref = prefs.getString("textmessagefield", "I need Help at Latitude: "+lat+"
                Longitude: "+lon);
                super.onResume();
        }

protected void launchCall()
        {
                PackageManager pm= getPackageManager();
                Intent intent = pm.getLaunchIntentForPackage("org.odk.collect.android");
                startActivity(intent);
                try
                {
                        startActivity(intent);
                }
                catch (Exception e)
                {
                        // TODO: handle exception
                        e.printStackTrace();
                }
        }
        public void dialEmergency()
        {
                //Declare Location variables
                //Call, finally
                //Makes a call to the saved number
                startActivity(new Intent(Intent.ACTION_CALL, Uri.parse("tel:"+ numberTextPref)));
```

```
            locationManager = (LocationManager)
            getSystemService(Context.LOCATION_SERVICE);
            locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, ,this);
    }

    // Menu section
    private static final int MENU_SETTINGS = Menu.FIRST;
    private static final int MENU_EXIT = Menu.FIRST + 1;

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
            super.onCreateOptionsMenu(menu);
            menu.add(0, MENU_SETTINGS, MENU_SETTINGS, "Settings");
            menu.add(0, MENU_EXIT, MENU_EXIT, "Exit");
            return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
            super.onOptionsItemSelected(item);
            switch (item.getItemId())
            {
            case MENU_SETTINGS:
                    Intent myIntent= new
                    Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                    startActivity(myIntent);
                    break;
            case MENU_EXIT:
                    Intent exit_intent = new Intent(Intent.ACTION_MAIN);
                    exit_intent.addCategory(Intent.CATEGORY_HOME);
                    exit_intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                    startActivity(exit_intent);
            }
            return true;
    }

  @Override
  public void onLocationChanged(Location loc)
  {
            latitude=loc.getLatitude();
            longitude=loc.getLongitude();
            lat=String.valueOf(loc.getLatitude());
            lon=String.valueOf(loc.getLongitude());
            locationManager.removeUpdates(this);

            //Use Geocoder to retrieve location

             Geocoder geocoder = new Geocoder(getBaseContext(), Locale.getDefault());

              // Changed the Max results to 10--to check
              try
              {
                      List<Address> list = geocoder.getFromLocation(latitude,longitude,10);
```

```
 if(list != null & list.size() > 0)
{
            Log.v("tanya", list.toString());
            Address address=list.get(0);
             //Retrieving city from address
            city=address.getLocality()+ "," + address.getSubLocality();
try
{
            SmsManager sms = SmsManager.getDefault();
            StringBuilder strReturnedAddress = new StringBuilder();
            for(int i=0; i<address.getMaxAddressLineIndex(); i++)
                        {
                        strReturnedAddress.append(address.getAddressLine(i)).append("\n");
                         }
 sms.sendTextMessage(numberTextPref, null, "HELP! I am at "+strReturnedAddress.toString()+". My
co-ordinates are "+lat+", "+lon, null, null);
  }
catch (Exception e)
{
            e.printStackTrace();
}
}
else
            {
            Toast.makeText(getBaseContext(),"No Address returned! Sending Co-ordinates via
            SMS!",Toast.LENGTH_SHORT).show();
            SmsManager sms = SmsManager.getDefault();
            sms.sendTextMessage(numberTextPref, null, "HELP! I am at these co-ordinates. "+lat+", "+lon,
null, null);
             }
}
catch (IOException e)
{
       // TODO Auto-generated catch block
            e.printStackTrace();
            Log.v("tanya", "error: "+e.getMessage());
            Toast.makeText(getBaseContext(),"Cannot get Address!",Toast.LENGTH_SHORT).show();
}       }

        @Override
        public void onProviderDisabled(String provider)
        {
           // TODO Auto-generated method stub
        }
        @Override
        public void onProviderEnabled(String provider)
        {
            // TODO Auto-generated method stub
        }
        @Override
        public void onStatusChanged(String provider, int status, Bundle extras)
        {
           // TODO Auto-generated method stub
        }
}// End of Class
```

SplashScreenActivity.java

```
package org.odk.collect.android.activities;

import org.odk.collect.android.R;
import org.odk.collect.android.application.Collect;
import org.odk.collect.android.preferences.PreferencesActivity;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.widget.ImageView;
import android.widget.LinearLayout;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

public class SplashScreenActivity extends Activity
{

    private static final int mSplashTimeout = 2000; // milliseconds
    private static final boolean EXIT = true;
    private int mImageMaxWidth;
    private AlertDialog mAlertDialog;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        // must be at the beginning of any activity that can be called from an external intent
        try {
            Collect.createODKDirs();
        } catch (RuntimeException e) {
            createErrorDialog(e.getMessage(), EXIT);
            return;
        }
        mImageMaxWidth = getWindowManager().getDefaultDisplay().getWidth();
        // this splash screen should be a blank slate
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.splash_screen);

        // get the shared preferences object
        SharedPreferences mSharedPreferences = PreferenceManager.getDefaultSharedPreferences(this);
        Editor editor = mSharedPreferences.edit();
        // get the package info object with version number
        PackageInfo packageInfo = null;
        try {
            packageInfo =
                getPackageManager().getPackageInfo(getPackageName(),
PackageManager.GET_META_DATA);
        } catch (NameNotFoundException e)
        {
            e.printStackTrace();
        }

        boolean firstRun = mSharedPreferences.getBoolean(PreferencesActivity.KEY_FIRST_RUN, true);
        boolean showSplash =
            mSharedPreferences.getBoolean(PreferencesActivity.KEY_SHOW_SPLASH, false);
        String splashPath =
            mSharedPreferences.getString(PreferencesActivity.KEY_SPLASH_PATH,
                getString(R.string.default_splash_path));

        // if you've increased version code, then update the version number and set firstRun to true
             if (mSharedPreferences.getLong(PreferencesActivity.KEY_LAST_VERSION, 0) <
            packageInfo.versionCode) {
            editor.putLong(PreferencesActivity.KEY_LAST_VERSION, packageInfo.versionCode);
            editor.commit();
            firstRun = true;
        }
        // do all the first run things
        if (firstRun || showSplash) {
            editor.putBoolean(PreferencesActivity.KEY_FIRST_RUN, false);
            editor.commit();
            startSplashScreen(splashPath);
        } else {
            endSplashScreen();
        }
    }
    private void endSplashScreen()
    {

        // launch new activity and close splash screen
        startActivity(new Intent(SplashScreenActivity.this, MainMenuActivity.class));
        finish();
    }
    // decodes image and scales it to reduce memory consumption
    private Bitmap decodeFile(File f) {
        Bitmap b = null;
        try {
            // Decode image size
            BitmapFactory.Options o = new BitmapFactory.Options();
            o.inJustDecodeBounds = true;
            FileInputStream fis = new FileInputStream(f);
            BitmapFactory.decodeStream(fis, null, o);
            try {
                fis.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
            e.printStackTrace();
        }
        int scale = 1;
        if (o.outHeight > mImageMaxWidth || o.outWidth > mImageMaxWidth) {
            scale =
                (int) Math.pow(
                    2,
                    (int) Math.round(Math.log(mImageMaxWidth
                            / (double) Math.max(o.outHeight, o.outWidth))
                            / Math.log(0.5)));
        }

        // Decode with inSampleSize
        BitmapFactory.Options o2 = new BitmapFactory.Options();
        o2.inSampleSize = scale;
        fis = new FileInputStream(f);
        b = BitmapFactory.decodeStream(fis, null, o2);
        try {
            fis.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    } catch (FileNotFoundException e) {
    }
    return b;
}
private void startSplashScreen(String path) {

    // add items to the splash screen here. makes things less distracting.
    ImageView iv = (ImageView) findViewById(R.id.splash);
    LinearLayout ll = (LinearLayout) findViewById(R.id.splash_default);
    File f = new File(path);
    if (f.exists()) {
        iv.setImageBitmap(decodeFile(f));
        ll.setVisibility(View.GONE);
        iv.setVisibility(View.VISIBLE);
    }

    // create a thread that counts up to the timeout
    Thread t = new Thread() {
        int count = 0;
        @Override
        public void run() {
            try {
                super.run();
                while (count < mSplashTimeout) {
                    sleep(100);
                    count += 100;
                }
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                endSplashScreen();
            }
            }
        };
        t.start();
    }
            private void createErrorDialog(String errorMsg, final boolean shouldExit) {
        Log.v("ERRORCHECK", errorMsg);
            Collect.getInstance().getActivityLogger().logAction(this, "createErrorDialog", "show");
        mAlertDialog = new AlertDialog.Builder(this).create();
        mAlertDialog.setIcon(android.R.drawable.ic_dialog_info);
        mAlertDialog.setMessage(errorMsg);
        DialogInterface.OnClickListener errorListener = new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int i) {
                switch (i) {
                    case DialogInterface.BUTTON_POSITIVE:
                            Collect.getInstance().getActivityLogger().logAction(this, "createErrorDialog", "OK");
                        if (shouldExit) {
                            finish();
                        }
                        break;
                }
            }
        };
        mAlertDialog.setCancelable(false);
        mAlertDialog.setButton(getString(R.string.ok), errorListener);
        mAlertDialog.show();
    }

    @Override
    protected void onStart() {
        super.onStart();
                Collect.getInstance().getActivityLogger().logOnStart(this);
    }

    @Override
    protected void onStop() {
                Collect.getInstance().getActivityLogger().logOnStop(this);
        super.onStop();
    }

}
```

WeatherActivity.java

```
package com.tanya.rescueapp;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.List;
import java.util.Locale;
import org.json.JSONArray;
import org.json.JSONObject;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import com.tanya.rescueapp.R;

public class WeatherActivity extends Activity implements LocationListener
{

    TextView source;
    TextView textViewMax;
    TextView textViewMin;
    TextView textViewRain;
    TextView textViewSunrise;
    TextView textViewSunset;
    Button alerts;
    Button view;
    JSONArray data;
   double latitude;
   double longitude;
   String lat;
   String lon;
  public LocationManager locationManager;
  String city;
  public String response,minTemp,maxTemp,rainfall,date,sunrise,sunset;

 @Override
                public void onCreate(Bundle savedInstanceState)
                        {
                    super.onCreate(savedInstanceState);
                    setContentView(R.layout.weatheractivity);

                    locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
                    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);

                  //Initialize buttons
                    alerts =(Button) findViewById(R.id.button1alert);
                    view =(Button) findViewById(R.id.button2view);
                  //Initialize Text View
                    source =(TextView) findViewById(R.id.txtSource);
                    textViewMax=(TextView) findViewById(R.id.textViewMax);
                textViewMin=(TextView) findViewById(R.id.textViewMin);
                textViewRain=(TextView) findViewById(R.id.textViewRain);
                textViewSunrise=(TextView) findViewById(R.id.textViewsunrise);
                textViewSunset=(TextView) findViewById(R.id.textViewsunset);

        // Calling Weather Alerts on button click
              alerts.setOnClickListener (new View.OnClickListener()
                    {
                    public void onClick(View v)
                    {
                            //Getting weather feeds
                            Intent nextScreen = new Intent(WeatherActivity.this,ReadXMLActivity.class);
                        startActivity(nextScreen);
                    }
            });
             // Calling Weather Geovisualiztion on button click
    view.setOnClickListener (new View.OnClickListener()
            {
                    public void onClick(View v)
                    {
                            // View Weather Data
                            Intent nextScreen = new Intent(WeatherActivity.this,ViewWeather.class);
                        startActivity(nextScreen);
                    }
            });
}
            private class BackgroundCall extends AsyncTask<Void, Void, Void>
              {
                            @Override
                            protected Void doInBackground(Void... params)
                    {
                                    // TODO Auto-generated method stub

                                    URL url;
                                    try {
url = new
URL("http://api.dataweave.in/v1/indian_weather/findByCity/?api_key=c9a17a31ac8c6aebf6edf52374fe
9e7ca956b264&city="+city);
HttpURLConnection con=(HttpURLConnection)url.openConnection();
con.setRequestMethod("GET");
con.setDoInput(true);
con.connect();
//con.connect();
InputStream is=con.getInputStream();

//PRINT INPUTSTREAM IN CONSOLE
BufferedReader br = new BufferedReader(new InputStreamReader(is));
StringBuilder sb = new StringBuilder();
String line;
while ((line = br.readLine()) != null)
{
sb.append(line);
}
response=sb.toString();
System.out.println(response);
```

```
br.close();
 JSONObject jsonObj = new JSONObject(response);
data = jsonObj.getJSONArray("data");
if(data.length()>0)
{
JSONObject tempObj=data.getJSONObject(0);
minTemp=tempObj.getJSONArray("Minimum Temp").getString(0);
maxTemp=tempObj.getJSONArray("Maximum Temp").getString(0);
date=tempObj.getString("date");
rainfall=tempObj.getJSONArray("24 Hours Rainfall").getString(0);
sunrise=tempObj.getJSONArray("Tommorows Sunrise").getString(0);
sunset=tempObj.getJSONArray("Todays Sunset").getString(0);
}
}
catch (Exception e)
{
Log.i("WeatherError", e.getMessage());
}
return null;
}
@Override
protected void onPostExecute(Void result)
{
		// TODO Auto-generated method stub
		super.onPostExecute(result);
		//Log.v("ERRORCHECK", artifact);
		//((TextView)findViewById(R.id.textView1)).setText(artifact);
if(data.length()>0)
{
 ((TextView)findViewById(R.id.txtMax)).setText(maxTemp);
 ((TextView)findViewById(R.id.txtMin)).setText(minTemp);
 ((TextView)findViewById(R.id.txtDate)).setText("As on "+date);
 ((TextView)findViewById(R.id.txtCityName)).setText(city);
 ((TextView)findViewById(R.id.txtsunset)).setText(sunset);
 ((TextView)findViewById(R.id.txtsunrise)).setText(sunrise);
if(rainfall.equalsIgnoreCase("nil")){
(TextView)findViewById(R.id.txtRain)).setText("No rainfall");}
else
{
		 ((TextView)findViewById(R.id.txtRain)).setText(rainfall);
}
else
{
AlertDialog.Builder builder=new AlertDialog.Builder(WeatherActivity.this);
builder.setTitle("Attention!");
builder.setMessage("No data found for this location.");
builder.setNegativeButton("OK",new DialogInterface.OnClickListener()
{
@Override
public void onClick(DialogInterface arg0, int arg1)
{
// TODO Auto-generated method stub
}
});
builder.show();
```

```
}
}
}
 public void onLocationChanged(Location loc)
     {
                    latitude=loc.getLatitude();
                    longitude=loc.getLongitude();
                    lat=String.valueOf(loc.getLatitude());
                    lon=String.valueOf(loc.getLongitude());
                    locationManager.removeUpdates(this);
                    //Use Geocoder to retrieve location
                     Geocoder geocoder = new Geocoder(getBaseContext(), Locale.getDefault());

                     // Changed the Max results to 10--to check
                      try
                      {
                              List<Address> list = geocoder.getFromLocation(latitude,longitude,10);

                              if(list != null & list.size() > 0)
                              {
                                        Log.v("tanya", list.toString());
                                        Address address=list.get(0);
                                        //Retrieving city from address
                                        city=address.getLocality();
                                        if(city==null)
                                                  city=address.getSubLocality();
                     // Toast.makeText(getApplicationContext(),city,Toast.LENGTH_SHORT).show();
                                       new BackgroundCall().execute();
                              }

                      }
                      catch (IOException e)
                      {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                              Log.v("tanya", "error: "+e.getMessage());
          Toast.makeText(getBaseContext(),"Cannot get Address!",Toast.LENGTH_SHORT).show();
                      }
     }

     public void onProviderDisabled(String provider)
     {
        // TODO Auto-generated method stub
     }

     public void onProviderEnabled(String provider) {
        // TODO Auto-generated method stub
     }

     public void onStatusChanged(String provider, int status, Bundle extras)
     {
        // TODO Auto-generated method stub
     }

}
```

SecondActivity.java

```
package com.tanya.rescueapp;

import com.tanya.rescueapp.R;
import android.os.Bundle;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.webkit.WebSettings;
import android.webkit.WebView;

public class SecondActivity extends Activity
{
        @SuppressLint("SetJavaScriptEnabled")
        @Override
        protected void onCreate(Bundle savedInstanceState)
        {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.secondactivity);

                // Declare webView
                final WebView map= (WebView) findViewById(R.id.webView1);
                WebSettings webSettings = map.getSettings();
                webSettings.setJavaScriptEnabled(true);

                map.loadUrl("file:///android_asset/www/openlayers.html");
}
}
```

FourthActivity.java

```
package com.tanya.rescueapp;

import com.tanya.rescueapp.R;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.View;
import android.widget.Button;

public class FourthActivity extends Activity
{

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

                //Declare Button variable
                 Button get_location =(Button) findViewById(R.id.get_location);
                 Button send_location =(Button) findViewById(R.id.send_location);

                // On click event for Get Location
                get_location.setOnClickListener (new View.OnClickListener()
                {
                        public void onClick(View v)
                        {
                        // Current data fetch from google maps
                                Intent nextScreen = new Intent(FourthActivity.this,MapActivity.class);
                        startActivity(nextScreen);
                        }
                });

        // On click event for Send Location
        send_location.setOnClickListener(new View.OnClickListener()
        {
                        public void onClick(View v)
                        {
                Intent nextScreen = new Intent(FourthActivity.this, CheckCode.class);
                     startActivity(nextScreen);

                        }
        });
}
}
```

Layout Files XML Code

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="@drawable/help"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:textAlignment="viewEnd"
  tools:context=".MainActivity" >
  <Button
    android:id="@+id/reporting1"
    style="android:buttonStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@android:color/transparent"
    android:layout_alignParentLeft="true"
    android:layout_centerVertical="true"
    android:layout_marginLeft="22dp"
    android:text="Reporting"
    android:textAlignment="viewStart"
    android:textSize="20dp"
    android:textStyle="bold|italic" />
  <Button
    android:id="@+id/weather1"
    style="android:buttonStyle"
    android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/reporting1"
        android:layout_below="@+id/reporting1"
        android:background="@android:color/transparent"
        android:text="Weather"
        android:textAlignment="viewStart"
        android:textSize="20dp"
        android:textStyle="bold|italic" />
    <Button
        android:id="@+id/assessment1"
        style="android:buttonStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/location1"
        android:layout_below="@+id/weather1"
        android:background="@android:color/transparent"
        android:text="Assessment"
        android:textAlignment="viewStart"
        android:textSize="20dp"
        android:textStyle="bold|italic" />
    <Button
        android:id="@+id/location1"
        style="android:buttonStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/weather1"
        android:layout_below="@+id/assessment1"
        android:background="@android:color/transparent"
        android:text="Location"
        android:textAlignment="inherit"
        android:textSize="20dp"
        android:textStyle="bold|italic" />
    <Button
        android:id="@+id/SOS"
        android:layout_width="70dp"
        android:layout_height="65dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="25dp"
        android:background="@drawable/round_button"
        android:clickable="true"
        android:text="SOS"
        android:textAlignment="center"
        android:textStyle="bold" />

</RelativeLayout>
```

<u>WeatherActivity.xml</u>

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:background="@drawable/helpbck"
    android:orientation="vertical"
    android:weightSum="1" >
```

```
<TextView
   android:id="@+id/txtCityName"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignBaseline="@+id/txtDate"
   android:layout_alignBottom="@+id/txtDate"
   android:layout_alignLeft="@+id/textViewMin"
   android:text=""
   android:textAppearance="?android:attr/textAppearanceMedium" />

<Button
   android:id="@+id/button2view"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignLeft="@+id/button1alert"
   android:layout_alignParentBottom="true"
   android:layout_marginBottom="28dp"
   android:text="View Weather Data" />

<TextView
   android:id="@+id/txtMin"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignBaseline="@+id/textViewMin"
   android:layout_alignBottom="@+id/textViewMin"
   android:layout_alignLeft="@+id/txtMax"
   android:text=""
   android:textAppearance="?android:attr/textAppearanceMedium" />

<Button
   android:id="@+id/button1alert"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_above="@+id/button2view"
   android:layout_alignLeft="@+id/textViewsunset"
   android:layout_marginBottom="15dp"
   android:text="Get Weather Alerts" />

<TextView
   android:id="@+id/textViewsunrise"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_below="@+id/textViewsunset"
   android:layout_marginTop="23dp"
   android:text="Sunrise Tomorrow(IST):"
   android:textAppearance="?android:attr/textAppearanceMedium" />

<TextView
   android:id="@+id/textViewsunset"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentLeft="true"
   android:layout_below="@+id/textViewRain"
   android:layout_marginTop="26dp"
```

```
        android:text="Sunset Today(IST):"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/textViewRain"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textViewMax"
        android:layout_marginTop="21dp"
        android:text="Rainfall (mm):"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/textViewMax"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textViewMin"
        android:layout_marginTop="22dp"
        android:text="Max Temp(Celcius):"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/textViewMin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtCityName"
        android:layout_marginTop="37dp"
        android:text="Min Temp(Celcius):"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/txtDate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="25dp"
        android:layout_marginTop="32dp"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/txtsunrise"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/textViewsunrise"
        android:layout_alignBottom="@+id/textViewsunrise"
        android:layout_alignParentRight="true"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
```

```
        android:id="@+id/txtSource"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/button1alert"
        android:layout_alignRight="@+id/txtDate"
        android:layout_marginBottom="25dp"
        android:text="Source -- IMD" />

    <TextView
        android:id="@+id/txtsunset"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/textViewsunrise"
        android:layout_alignLeft="@+id/txtsunrise"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/txtRain"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/textViewsunset"
        android:layout_alignRight="@+id/txtsunset"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/txtMax"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/textViewMax"
        android:layout_alignBottom="@+id/textViewMax"
        android:layout_alignRight="@+id/txtRain"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceMedium" />

</RelativeLayout>
```

RescueApp Manifest File

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tanya.rescueapp"
    android:versionCode="1"
    android:versionName="1.0" >


<uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="19" />

    <uses-feature android:name="android.hardware.location" android:required="false" />
    <uses-feature android:name="android.hardware.location.network" android:required="false" />
    <uses-feature android:name="android.hardware.location.gps" android:required="false" />
    <uses-feature android:name="android.hardware.telephony" android:required="false" />
    <uses-feature android:name="android.hardware.wifi" android:required="false" />
    <uses-feature android:glEsVersion="0x00020000"  android:required="true"/>

    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.USE_CREDENTIALS" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>

     <permission
         android:name="org.opendatakit.tables.permission.MAPS_RECEIVE"
         android:protectionLevel="signature" />

    <uses-permission android:name="org.opendatakit.tables.permission.MAPS_RECEIVE" />
    <uses-permission  android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"
/>
      <application
       android:allowBackup="true"
       android:name="org.odk.collect.android.application.Collect"
       android:icon="@drawable/ic_rescueicon"
       android:label="@string/app_name"
       android:theme="@style/AppTheme"
       android:logo="@animator/staranimation">
       <provider
         android:exported="true"
         android:name="org.odk.collect.android.provider.FormsProvider"
         android:authorities="org.odk.collect.android.provider.odk.forms" />
           <provider
           android:exported="true"
           android:name="org.odk.collect.android.provider.InstanceProvider"
           android:authorities="org.odk.collect.android.provider.odk.instances" />

       <activity
           android:name="org.odk.collect.android.activities.FormEntryActivity"
           android:configChanges="orientation"
           android:label="@string/app_name"
           android:windowSoftInputMode="adjustResize" >
           <intent-filter>
               <action android:name="android.intent.action.VIEW" />
               <action android:name="android.intent.action.EDIT" />
               <category android:name="android.intent.category.DEFAULT" />
               <data android:mimeType="vnd.android.cursor.item/vnd.odk.form" />
               <data android:mimeType="vnd.android.cursor.item/vnd.odk.instance" />
           </intent-filter>
       </activity>
       <activity
```

```
   android:name="org.odk.collect.android.activities.DrawActivity"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.InstanceChooserList"
   android:label="@string/app_name" />
<intent-filter>
   <action android:name="android.intent.action.VIEW" />
   <action android:name="android.intent.action.EDIT" />
   <category android:name="android.intent.category.DEFAULT" />
   <data android:mimeType="vnd.android.cursor.dir/vnd.odk.instance" />
</intent-filter>
<activity
   android:name="org.odk.collect.android.activities.InstanceChooserTabs"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.FormChooserList"
   android:label="@string/app_name" >
   <intent-filter>
      <action android:name="android.intent.action.VIEW" />
      <action android:name="android.intent.action.EDIT" />
      <action android:name="android.intent.action.PICK" />
      <category android:name="android.intent.category.DEFAULT" />
      <data android:mimeType="vnd.android.cursor.dir/vnd.odk.form" />
   </intent-filter>
</activity>
<activity
   android:name="org.odk.collect.android.activities.FormManagerList"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.FormDownloadList"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.DataManagerList"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.FileManagerTabs"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.InstanceUploaderList"
   android:label="@string/app_name">
        <intent-filter>
           <action android:name="android.intent.action.VIEW" />
           <action android:name="android.intent.action.EDIT" />

           <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
</activity>
<activity
   android:name="org.odk.collect.android.activities.InstanceUploaderActivity"
   android:label="@string/app_name" />
 <activity
   android:name="org.odk.collect.android.activities.MainMenuActivity"
   android:configChanges="orientation"
   android:label="@string/app_name" >
</activity>
```

```
<activity
   android:name="org.odk.collect.android.preferences.PreferencesActivity"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.preferences.AdminPreferencesActivity"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.FormHierarchyActivity"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.GeoPointActivity"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.GeoPointMapActivity"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.GeoPointMapActivitySdk7"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.BearingActivity"
   android:label="@string/app_name" />
<activity
   android:name="org.odk.collect.android.activities.SplashScreenActivity"
   android:theme="@android:style/Theme.Dialog" >
</activity>
<!-- Enable Shortcuts for Command Actions -->
<activity
   android:name="org.odk.collect.android.activities.AndroidShortcuts"
   android:label="ODK Form"
   android:theme="@android:style/Theme.Translucent.NoTitleBar" >
   <intent-filter>
      <action android:name="android.intent.action.CREATE_SHORTCUT" />

      <category android:name="android.intent.category.DEFAULT" />
   </intent-filter>
</activity>
 <receiver
   android:name="org.odk.collect.android.receivers.NetworkReceiver"
   android:enabled="true" >
   <intent-filter>
      <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
   </intent-filter>
   <intent-filter>
      <action android:name="org.odk.collect.android.FormSaved" />
   </intent-filter>
</receiver>

<activity
   android:name="com.tanya.rescueapp.MainActivity"
   android:screenOrientation="portrait"
   android:label="@string/app_name" >
   <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
```

```
</activity>
     <activity
      android:name="com.tanya.rescueapp.SecondActivity"
       android:screenOrientation="portrait"
      android:label="@string/app_name" >
    </activity>
      <activity
            android:name="com.tanya.rescueapp.ThirdActivity"
            android:screenOrientation="portrait"
            android:label="@string/app_name" >
      </activity>
     <activity
            android:name="com.tanya.rescueapp.FourthActivity"
            android:screenOrientation="portrait"
            android:label="@string/app_name" >
  </activity>
     <activity
         android:name="com.tanya.rescueapp.Feeds"
         android:screenOrientation="portrait"
         android:label="@string/app_name" >
     </activity>
      <activity
   android:name="com.tanya.rescueapp.ReadXMLActivity"
   android:screenOrientation="portrait"
   android:label="@string/app_name" >
     </activity>
      <activity
   android:name="com.tanya.rescueapp.RSSAdapter"
   android:screenOrientation="portrait"
   android:label="@string/app_name" >
     </activity>
        <activity
   android:name="com.tanya.rescueapp.RSSFeedClass"
   android:screenOrientation="portrait"
   android:label="@string/app_name" >
     </activity>
    <activity
   android:name="com.tanya.rescueapp.GetWeather"
   android:screenOrientation="portrait"
   android:label="@string/app_name" >
   </activity>
    <activity
   android:name="com.tanya.rescueapp.ViewWeather"
   android:screenOrientation="portrait"
   android:label="@string/app_name" >
   </activity>
      <activity
   android:name="com.tanya.rescueapp.GetCity"
   android:screenOrientation="portrait"
   android:label="@string/app_name" >
     </activity>
      <activity
            android:name="com.tanya.rescueapp.SendSms"
            android:screenOrientation="portrait"
            android:label="@string/app_name" >
        </activity>
            <activity
                android:name="com.tanya.rescueapp.CheckCode"
                android:screenOrientation="portrait"
                android:label="@string/app_name" >
          </activity>
           <activity
                android:name="com.tanya.rescueapp.GetLocation"
                android:screenOrientation="portrait"
                android:label="@string/app_name" >
         </activity>
           <activity
        android:name="com.tanya.rescueapp.Weather"
        android:screenOrientation="portrait"
        android:label="@string/app_name" >
         </activity>
      <activity
        android:name="com.tanya.rescueapp.WeatherHTTPClient"
        android:screenOrientation="portrait"
        android:label="@string/app_name" >
        </activity>
      <activity
        android:name="com.tanya.rescueapp.JSONWeatherParser"
        android:screenOrientation="portrait"
        android:label="@string/app_name" >
      </activity>
        <activity
        android:name="com.tanya.rescueapp.Emergency"
        android:screenOrientation="portrait"
        android:label="@string/app_name" >
      </activity>
        <activity
        android:name="com.tanya.rescueapp.ShowWeather"
        android:screenOrientation="portrait"
        android:label="@string/app_name" >
      </activity>
       <activity
                android:name="com.tanya.rescueapp.WeatherActivity"
                android:screenOrientation="portrait"
                android:label="@string/app_name" >
          </activity>
        <meta-data
                        android:name="com.google.android.maps.v2.API_KEY"
                        android:value="AIzaSyCVQIPkijBsJKdbHtuMJ3Icm1ibIFVwAA8"/>

      <meta-data
   android:name="com.google.android.gms.version"
   android:value="@integer/google_play_services_version" />
       <activity android:name="MapActivity"
          android:screenOrientation="portrait"
          android:label="@string/app_name" >
        </activity>
   </application>
</manifest>
```

First Incident Report form

```
<h:html xmlns="http://www.w3.org/2002/xforms"
xmlns:h="http://www.w3.org/1999/xhtml"
xmlns:ev="http://www.w3.org/2001/xml-events"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:jr="http://openrosa.org/javarosa">
  <h:head>
    <h:title>First Incident Report</h:title>
    <model>
      <instance>
        <data id="build_First-Incident-Report_1389677284">
          <meta>
            <instanceID/>
          </meta>
          <reportingofficer/>
          <date/>
           <time/>
          <location/>
          <image/>
          <video/>
          <resources/>
           <information/>
        </data>
      </instance>
      <itext>
        <translation lang="eng">
          <text id="/data/reportingofficer:label">
            <value>Reporting Officer</value>
          </text>
          <text id="/data/date:label">
            <value>Date of Occurence</value>
          </text>
          <text id="/data/date:hint">
            <value></value>
          </text>
           <text id="/data/Time:label">
            <value>Time</value>
          </text>
          <text id="/data/Time:hint">
            <value>Choose the current time</value>
          </text>
          <text id="/data/location:label">
            <value>Location</value>
          </text>
          <text id="/data/image:label">
            <value>Capture Image</value>
          </text>
          <text id="/data/video:label">
            <value>Record Video</value>
          </text>
          <text id="/data/resources:label">
            <value>Resources Required</value>
          </text>
          <text id="/data/resources:option0">
            <value>Medical Help</value>
          </text>
          <text id="/data/resources:option1">
            <value>Blankets</value>
          </text>
          <text id="/data/resources:option2">
            <value>Food and water supplies</value>
          </text>
        </translation>
      </itext>
      <bind nodeset="/data/meta/instanceID" type="string"
readonly="true()" calculate="concat('uuid:', uuid())"/>
      <bind nodeset="/data/reportingofficer" type="string"
required="true()"/>
      <bind nodeset="/data/date" type="date" required="true()"/>
     <bind nodeset="/data/Time" type="time" jr:preload="timestamp"
jr:preloadParams="start"/>
      <bind nodeset="/data/location" type="geopoint"
required="true()"/>
      <bind nodeset="/data/image" type="binary" required="true()"/>
      <bind nodeset="/data/video" type="binary"/>
      <bind nodeset="/data/resources" type="select1"/>
    </model>
  </h:head>
  <h:body>
    <input ref="/data/reportingofficer">
      <label ref="jr:itext('/data/reportingofficer:label')"/>
    </input>
    <input ref="/data/date">
      <label ref="jr:itext('/data/date:label')"/>
      <hint ref="jr:itext('/data/date:hint')"/>
    </input>
     <input ref="/data/Time">
      <label ref="jr:itext('/data/Time:label')"/>
      <hint ref="jr:itext('/data/Time:hint')"/>
    </input>
    <input ref="/data/location">
      <label ref="jr:itext('/data/location:label')"/>
    </input>
    <upload ref="/data/image" mediatype="image/*">
      <label ref="jr:itext('/data/image:label')"/>
    </upload>
    <upload ref="/data/video" mediatype="video/*">
      <label ref="jr:itext('/data/video:label')"/>
    </upload>
    <select1 ref="/data/resources">
      <label ref="jr:itext('/data/resources:label')"/>
      <item>
        <label ref="jr:itext('/data/resources:option0')"/>
        <value/>
      </item>
      <item>
        <label ref="jr:itext('/data/resources:option1')"/>
        <value/>
      </item>
      <item>
        <label ref="jr:itext('/data/resources:option2')"/>
        <value/>
      </item>
    </select1>
  </h:body>
</h:html>
```