Rakshak: Post-Disaster Management through an Android Mobile Application

Ayush Roy[1], Devansh Mehta[2]
[1]*School of Computer Science and Engineering, VIT University*
[2]*School of Computer Science and Engineering, VIT University*

Abstract – In the aftermath of a natural or man-made disaster, the most important thing is rescue and reconstruction. Large-scale disasters cause significant damage to lives, livelihood, and infrastructure. The breakdown of existing infrastructure is particularly detrimental since it also hampers rescue efforts and emergency services. From an administrative standpoint, effective logistical management and allocation of resources, as well as getting real-time reports, is highly imperative.

Key Words: Disaster Management, Application, GIS, SOS

1. INTRODUCTION

The idea behind this solution is to create a comprehensive mobile application that incorporates the good parts of existing solutions while using some more robust technologies to counter the limitations [1]. The app serves a two-fold purpose. First, it provides assistance to users by alerting them of incoming disasters and providing support in the event of one. Secondly, and more importantly, it provides real-time data to disaster management agencies, enabling them to make decisions on the go.

1.1 Ideation

The app requires users to upload their information beforehand. In the event of a disaster, the app will ask them to upload their real-time status using an interactive interface. The app also sends out their location and uses the data entered by users to judge the severity and identify hotspots. The app also has an SOS feature that sends out a notification to the necessary emergency services, emits a loud alarm, and uses the phone's camera flash to act as a beacon.

On the administrative dashboard, hotspots are marked on a map [2] and colour-coded based on severity. This enables the authorities to make the right decisions based on the real-time data.

1.2 Motivation for the Idea

The motivation for this idea came from reviewing the disaster response plans and methodologies currently in place. Even with the advancements in Data Science, the authorities still follow rigid protocols, which may or may not be the most efficient response to the situation. Our aim was to create a solution that enables a more dynamic management response.

2. EXISTING SYSTEM AND LIMITATIONS

The existing solution of a post-disaster management system works specifically for an extensible and usable platform for the basic coordination functions required in responding to disaster [3]. It includes the features of the addition or modification of events by the developer team. The affected people can send their responses (i.e., modes of recovery needed and urgency) to the recovery team via SMS or Email. This system is useful, but it lacks location accuracy and provides users with the control to update any sudden disasters on the app and quick response to the same. The existing system provides a map view of events, a list view of events, and information on what to do in case of an emergency. This lacks the prediction part of any disaster and upcoming signs of any disaster.

The system is composed of three basic parts:

1. main (server) app, which professionals use
2. web services, which offer data to the public
3. mobile apps, which can be freely downloaded and used by citizens for them to stay informed of important events

2.1 Existing App Example (TN SMART)

The existing system (TN SMART) is an Android app that gives weather alerts, but it has some disadvantages:

1. Requires user to provide detailed information for sign up, which in turn requires the user to waste precious time to fill the form before they can access any feature

2. This app is focused mainly on providing weather and disaster forecasts. But most of the time, disasters cannot be predicted beforehand

3. To access the map in this app, users must navigate to a link in their web browser. Most of the time, a user does not want unnecessary information like cloud cover, etc.

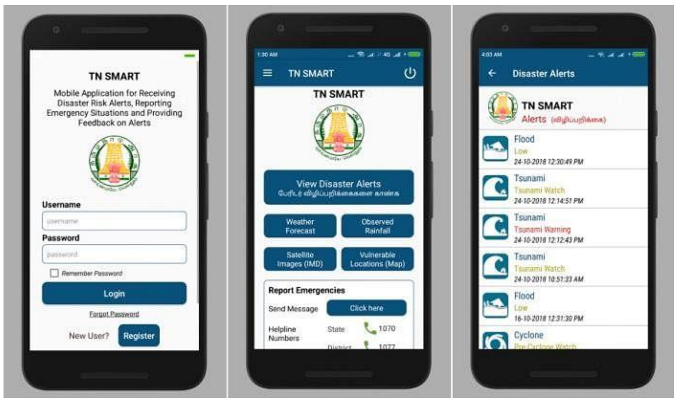


Fig – 1: *TN SMART App Interface*

3. PROPOSED SYSTEM

The proposed design for the system includes three essential models. These modules provide the basis of the application. The backend resource is currently dependent on existing open-source APIs [4].

a. <u>Login Module</u>

The user can log in with their mobile phone number. An OTP will be sent and received on their phone. It will be automatically verified. Once it is verified, the user is taken to the main screen.

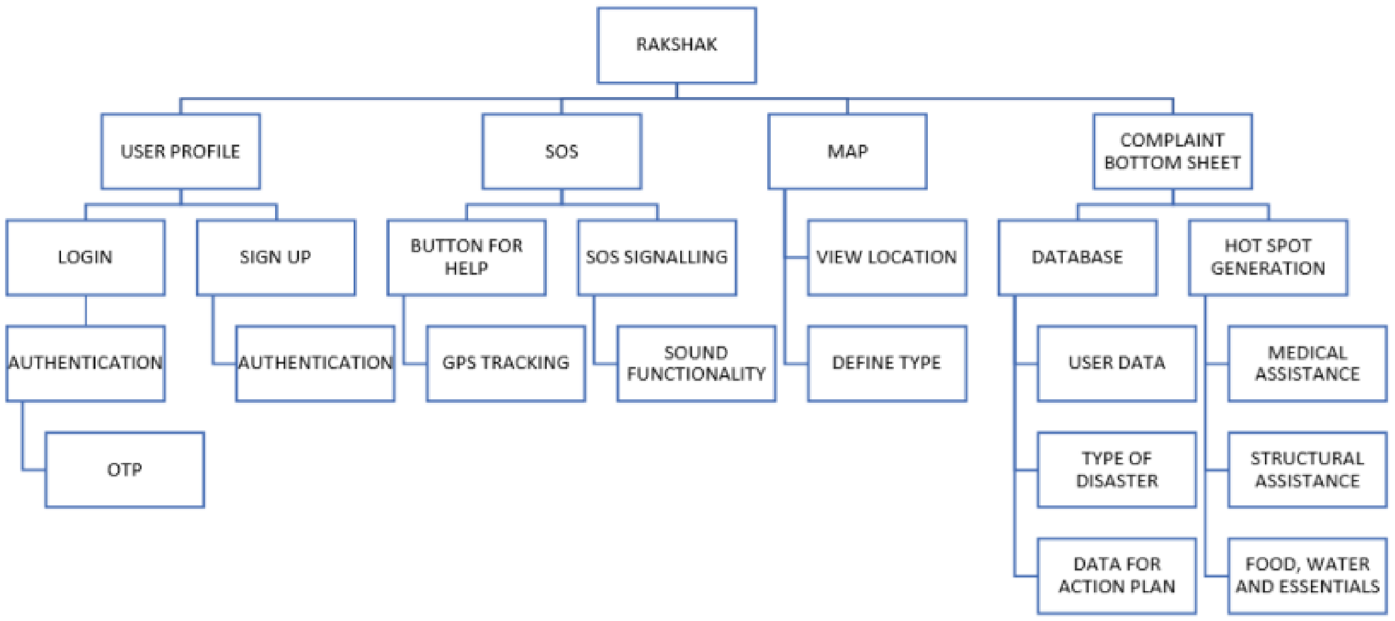
b. <u>Map View Module</u>

The user is initially shown a world map view with all the reported disasters. Each disaster type is colour-coded, i.e., every disaster is shown with a marker of a different colour. On tap, the marker displays the disaster type it represents. Also, instead of displaying many markers when the map is zoomed out, the markers are clustered and shown as a heatmap. Whenever a new disaster is reported, the map pans and zooms to the new location. There is a '+' button to report a new disaster.

c. <u>Alerts Module</u>

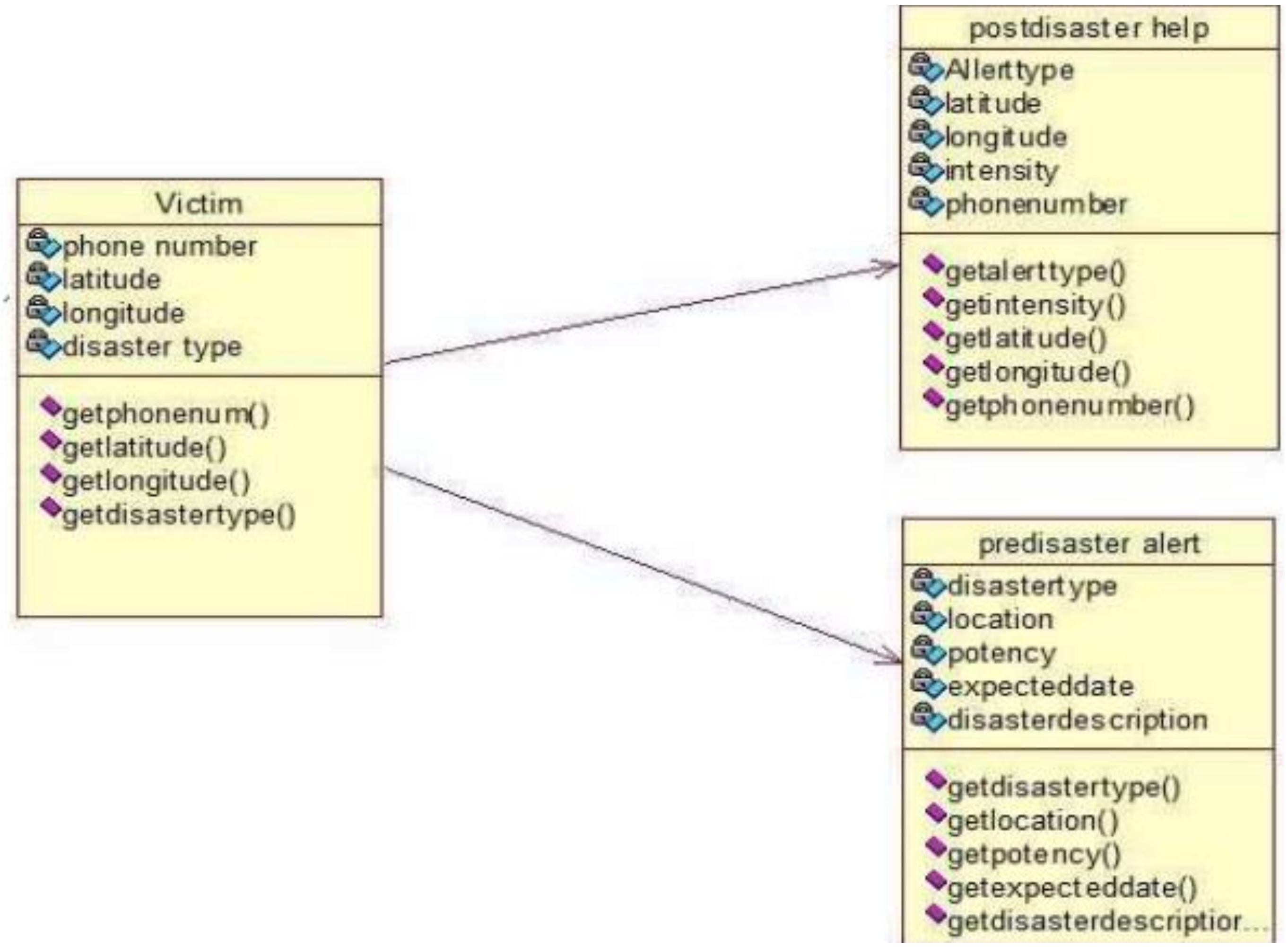
On pressing the '+' button in the map view, a bottom sheet is opened, which contains some pre-set disaster types and also a 'other' button, which, when pressed, gives the user the ability to submit a disaster type which is not already present. There is a field where users can enter the intensity of the disaster. After filling out these two things, the user's location is taken automatically, the map is panned to the current location, and the marker of the current disaster is added.

3.1 Break Down Structure

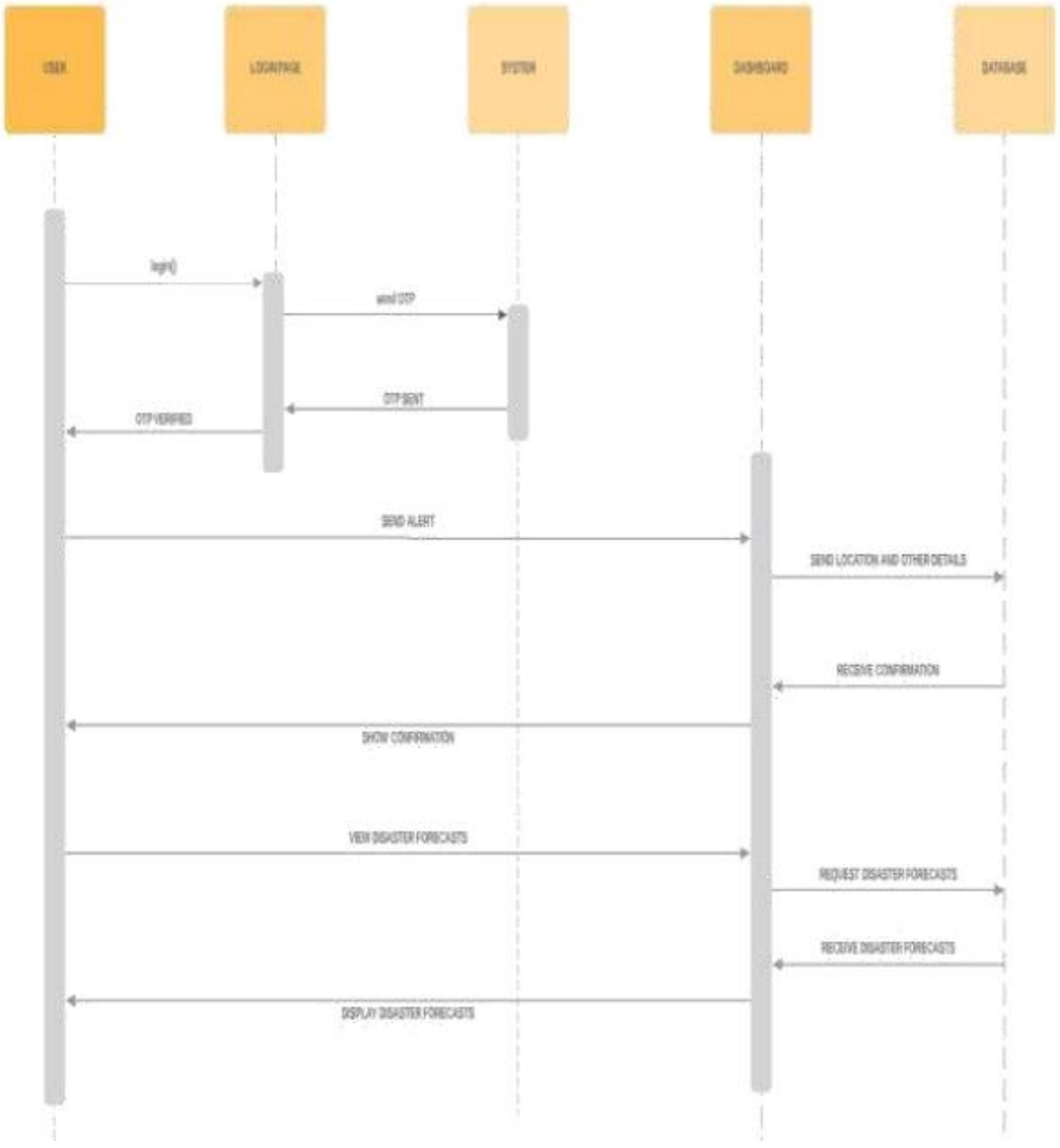


3.2 System Design

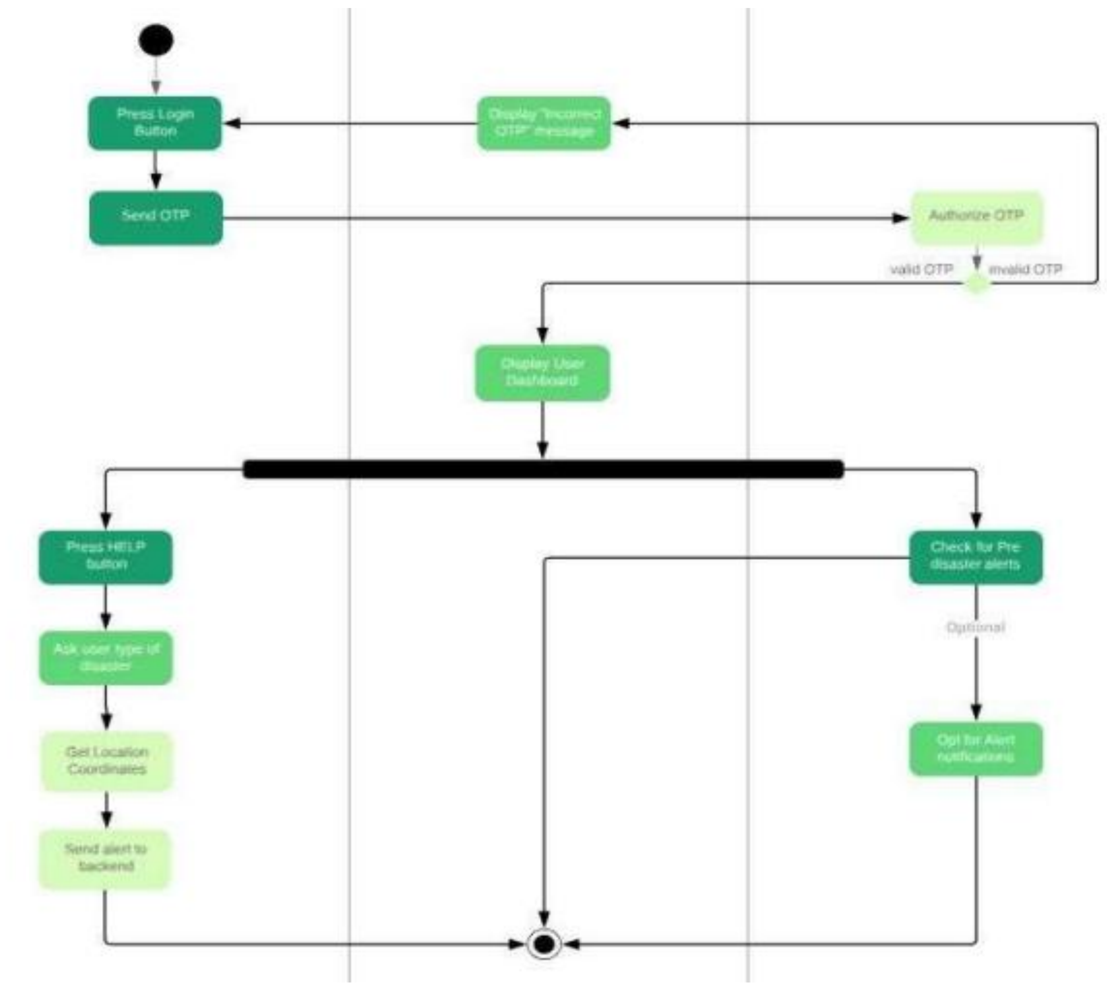
3.2.1 Use-case Diagram



3.2.2 Sequence Diagram



3.2.3 Activity Diagram



3.2.4 Tech Stack

1. Backend: APIs written in NodeJs, Firebase Services
2. Maps: Google Map SDK
3. Frontend: Android using Kotlin and iOS using SwiftUI
4. Load balancing: Nginx

The conscious decision to go for separate iOS and Android apps rather than having a common hybrid app has been made in order to keep the app size as minimal as possible so as to support every kind of user device. Kotlin for Android is the latest stack being used with backward compatibility. And SwiftUI is the latest one for iOS.

NodeJS is the industry standard; hence, services are built using that. We've also integrated Firebase Services for other Platforms tasks like performing Authentication [5].

4. RESULTS AND CONCLUSION

The following snippets show the application interface as well as the functionality. Currently, most of the app's functions require an active net connection. However, future enhancements would include the creation of a Bluetooth mesh network which will work even in the absence of network connectivity. Another enhancement would be to develop a custom backend for the app which uses real-time data on trained ML models to indicate the appropriate actions, thus eliminating the middlemen and streamlining the whole process.

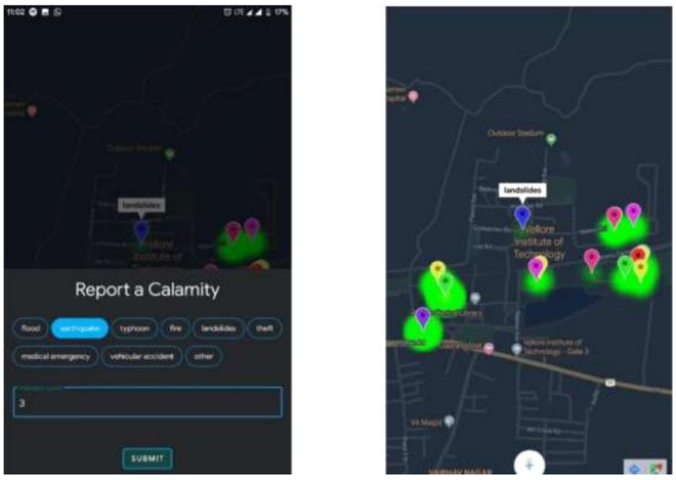


__Fig – 2:__ Rakshak Map Dashboard with Hotspot

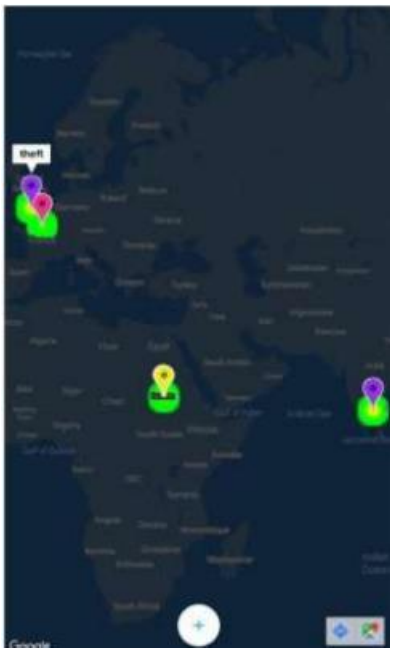


__Fig – 3:__ Global Map Sample

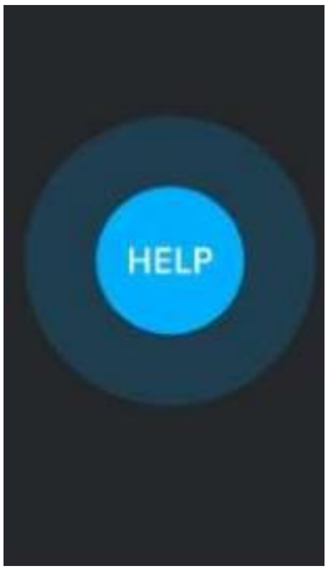


Fig – 4: *SOS button snippet*

REFERENCES

1. Das, Pulak. "Disaster management in India: policy review and institutional structure." *Asiapac Journal of Socical Sciences* 4 (2012): 37-52.
2. Mishra, Pramod K. "Maps and disaster management." *Economic and Political Weekly* (2002): 4676-4677.
3. Souza, Flávio, and Ibrahim Kushchu. "Mobile disaster management system applications–current overview and future potential." *Proceedings EURO mGOV* (2005): 455-466.
4. Sharma, Sujeet Kumar, Santosh K. Misra, and Jang Bahadur Singh. "The role of GIS-enabled mobile applications in disaster management: A case analysis of cyclone Gaja in India." *International Journal of Information Management* 51 (2020): 102030.
5. Khatun, Shanta, Fahim Hossain Saiki, and Milon Biswas. "SecureIT using Firebase, Google map and Node. Js." *arXiv preprint arXiv:2004.05880* (2020).
6. Bruce, K.B., Cardelli, L., Pierce, B.C.: Comparing Object Encodings. In: Abadi, M., Ito, T. (eds.): Theoretical Aspects of Computer Software. Lecture Notes in Computer Science, Vol. 1281. Springer-Verlag, Berlin Heidelberg New York (1997) 415–438
7. van Leeuwen, J. (ed.): Computer Science Today. Recent Trends and Developments. Lecture Notes in Computer Science, Vol. 1000. Springer-Verlag, Berlin Heidelberg New York (1995)