International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 20, No. 1, 2024
© 2024 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

Suraksha AI: An Intelligent Women's Safety and Emergency Alert System Using Artificial Intelligence

[1]Bellala Indhu [2]Vemula Pawan Kalyan
[3]Thalakokkula Madhav [4]Indrasena Reddy
[5]Gade Venkata Vara Prasad, Associate Professor
Department of CSE (AI&ML) Narsimha Reddy Engineering College (Autonomous),
Hyderabad,TS -500100

***Abstract*—**Personal Personal safety apps are often not working when they are most critical due to issues with cloud-based alert pipelines where mobile data coverage fails. The paper introduces an offline-first Android emergency response application (ERA) developed using Java (SDK 21-32) without the need to have an active internet connection during the dispatching in case of distress. The architecture splits the user authentication and trusted-contact management into two locally stored SQLite databases (login.db & mylist.db) and provides Zero-Latency access to the device's emergency numbers. The safety dashboard checks if GPS data is available and gets latitude and longitude information from the GPS provider Android LocationManager during launch. A single press of the SOS button sends a direct cellular call to the emergency services as well as GPS-tagged SMS to all contacts that have been stored with SmsManager. A long press on the alarm control sends a high volume alarm sound for close threats using MediaPlayer and will not trigger if it gains unintended activation. In addition to defining the architecture of 6 activities and formalizing the latency, reliability and energy models, this work performs a static code audit of the java implementation and identifies concrete hardening measures for credential hashing, new location acquisition and (O(k)) SMS deduplication.

Key Words—Android, emergency response, GPS location, offline-first, SMS manager, SOS alert, SQLite, women safety.

I. INTRODUCTION

A public safety issue consistently exists for personal safety during commuting, late shift travel and moving through unknown urban or semi-urban environments, especially for women walking alone. If the user is in an acute threat situation, they may only have a few seconds to respond. Manual unlocking, dialing a helpline and typing a message, as well as describing nearby landmarks, creates a fatal cognitive and physical delay. Mobile emergency applications are designed to condense those steps into one very quick interaction to alert police and trusted family members.

Yet, while most of the world is having a smartphone revolution, many of today's safety applications are built on a fragile architecture premise: continuous packet data connectivity. In conditions with weak 4G/5G signals for instance in underground stations, elevators, rural highways or crowded public spaces, applications using Firebase Realtime Database, REST APIs or cloud-hosted map tiles slow down [3] or stop working [4]. Cellular voice and Short Message Service (SMS) channels are usually still in place and working, as well as on-device GPS receivers, which calculate coordinates passively from the satellites' broadcasts, in those very dead zones.

In order to eliminate data-network failure points from emergency dispatch, we suggest a Development of a native Android safety application written in Java 8 for Android 5.0 (API 21) to Android 12 (API 32) named as Stree Raksha. Stree Raksha has both user account and emergency contact databases which are stored locally, using two separate SQLite databases, and also coordinates GPS acquisition with the simultaneous broadcasting of an SMS via GSM, or the dialing of an emergency number, and provides a long press local audio "siren" for immediate physical deterrence.

International Journal of Engineering Research and Science & Technology

The key contributions of this work are as follows: (1) an offline-first, zero-cloud-dependency Android architecture for emergency response; (2) a dual-SQLite persistence design separating authentication from contact management; (3) analytical latency, reliability and energy models for multi-recipient SOS dispatch; and (4) a transparent code-level security and complexity audit outlining concrete production hardening steps.

II. LITERATURE SURVEY

There are three main design paradigms for smartphone-enabled women's safety: GSM/SMS-alert based, cloud based real-time tracking and wearable or sensor augmented. Early prototypes for Android were based on the idea of getting the user to do fewer steps in the event of distress. Yarrabothu and Thota developed Abhaya [1] an Android application that on a single tap, will obtain GPS coordinates, call the primary registered contact, and send regular SMS text updates containing a URL location every five minutes until the app is turned off. Chand et al. [2] created WoSApp with the ability to make an emergency call to the police through an on-screen panic button and shake the handset to send the location coordinates to pre-selected contacts.

The second wave of systems focused on external hardware or multiple sensor activation. Mahajan et al. [5] and Monisha et al. [6] combined wearable microcomputers and GSM/GPS devices with smartphone applications to alert a victim without requiring hand access to their phones. The use of a dedicated wearable device reduces user compliance down the road, while hardware triggers enhance discretion. Recently, Aggarwal et al. [7] scanned Android safety methods such as geofencing, voice commands and accelerometer shake detection, and reported that false positive due to normal phone motions are still a major usability hurdle.

For continuous monitoring, cloud-first architectures have also become popular. Prashanth et al. [3] connected the Firebase real-time databases with real-time GPS streaming while Jothilakshmi et al. [4] suggested a cross-platform system built with React Native and Firebase that provides a one-click SOS button with the option of audio recording and a web-based admin portal. But cloud-based apps silently fail when MO data service is turned off or throttled. Stree Raksha does this by ensuring that all critical paths for dispatch communication lookups, GPS reading, SMS transmission, voice calling and siren playback are strictly on-device, implemented via native Android APIs [8]–[11]. This is summarized in Table 1.

TABLE I

Comparison of Existing Women's Safety Systems and Stree Raksha

System / Study	Trigger & Alert	Network / DB	Key Limitation
Abhaya [1]	1-tap call + 5-min SMS loop	Local + GSM/GPS	No local audio siren deterrent
WoSApp [2]	Shake / panic police call	Local + GSM/GPS	Requires police tie-up; no siren
FEMME [6]	Wearable button + GSM/GPS	Hardware + GSM	Extra wearable cost & charging
Prashanth [3]	SOS + live GPS stream	Firebase (Online)	Fails without mobile data
Jothilakshmi [4]	1-tap SOS + audio + web	React Native + Cloud	Requires active internet
Stree Raksha	1-tap Call+SMS; long-press siren	Dual SQLite (Offline)	Needs password hashing & Fused GPS

III. PROPOSED SYSTEM ARCHITECTURE

A. Six-Activity Navigation and Interface Design

The implementation of Stree Raksha is done on Java 8 with AndroidX AppCompat (1.4.1), material components (1.6.0) and Constraint Layout (2.1.3). There are six activities to coordinate the user cycle in AndroidManifest.xml. MainActivity is the one that launches the app, showing a 3000ms splash screen using Handler.Use the postDelayed() method before navigating to LoginActivity and LoginActivity2 for account sign-up and sign-in. After successful authentication, the user will be directed to MainActivity2, the main emergency dashboard (Fig. 1). Restoring legibility in context of stress, its layout is based on three buttons of action: ADD CONTACTS (launching the Register activity), EMERGENCY!! and EMERGENCY!! (both of which launch the Emergency activity). Voice and SMS alerts (dispatched) and EMERGENCY ALARM (to set off the siren).

International Journal of Engineering Research and Science & Technology

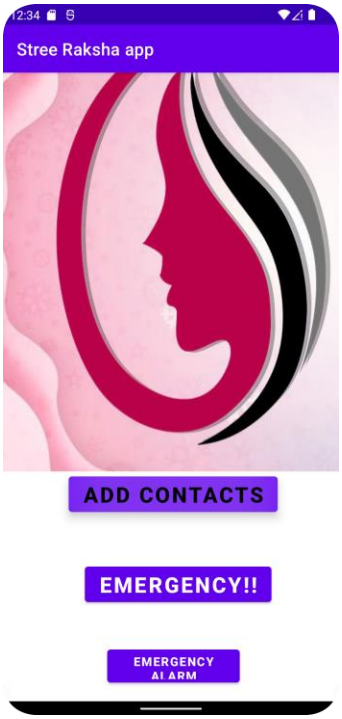

Fig. 1. Stree Raksha primary emergency dashboard (MainActivity2) showing ADD CONTACTS, EMERGENCY!!, and EMERGENCY ALARM controls.

B. Dual-SQLite Persistence Layer

Rather than mixing user credentials and emergency contact records in a single schema, Stree Raksha isolates them into two dedicated SQLiteOpenHelper classes [11]. DBHelper manages login.db (version 1) containing table users(username TEXT PRIMARY KEY, password TEXT). It exposes insertData(username, password) via ContentValues, duplicate checking via checkusername(username), and login validation via checkusernamepassword(username, password). Independently, DatabaseHandler manages mylist.db (version 1) containing table mylist_data(ID INTEGER PRIMARY KEY AUTOINCREMENT, ITEM1 TEXT). In the Register activity, users add trusted phone numbers through addData(), view stored entries in a ListView populated via getListContents(), and remove outdated numbers.

TABLE II

Software Stack, Permissions, and Dual-SQLite Schema

Layer / Module	Technical Specification
Target Platform	Android Java 8; minSdk 21, compile/targetSdk 32
UI Framework	AppCompat 1.4.1, Material 1.6.0, ConstraintLayout
Location APIs	LocationManager + Play Services Location 19.0.1
Auth DB (login.db)	DBHelper: users(username PK, password)
Contacts DB (mylist.db)	DatabaseHandler: mylist_data(ID PK, ITEM1)
Audio Asset	MediaPlayer with bundled res/raw/alarm.mp3
SMS & Telephony	SmsManager.getDefault(), Intent.ACTION_CALL
Manifest Permissions	CALL_PHONE, SEND/READ/RECEIVE_SMS, LOCATION

C. Permission Governance and GPS Acquisition

AndroidManifest.xml includes nine permissions for telephony (CALL_PHONE, READ_PHONE_STATE), messaging (SEND_SMS, RECEIVE_SMS, READ_SMS), location (ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, ACCESS_BACKGROUND_LOCATION), and INTERNET. On MainActivity2 start, it asks for permission to send SMS at runtime using ActivityCompat.get the permissions with requestPermissions() and verify with LocationManager.isProviderEnabled(GPS_PROVIDER) [9]. If GPS is not enabled, onGPS() displays an AlertDialog that cannot be canceled, and asks the user to go to the Settings.ACTION_LOCATION_SOURCE_SETTINGS. Otherwise, startTrack() checks to see if they have granted fine or coarse location and queries for the last known location (GPS_PROVIDER), storing the latitude x and longitude y as strings for later use in the Sos.

D. Multi Channel SOS Dispatch & Long Press Siren

When you loadData(), you open a SQLite Cursor on mylist.db, which is triggered by tapping EMERGENCY!!.When you tap EMERGENCY!!, you trigger a call to loadData(), which calls for an opening of a SQLite Cursor on the file mylist.db. When there are no contacts saved, a Toast alert will show to the user that they need to save numbers first. If any contacts exist, call() will verify permission for the call, requesting it dynamically if required, on Android 6.0+ / API 23+ and send an Intent(Intent.ACTION_CALL) to emergency helpline tel:1000. The service sendsSms() formats the payload 'I NEED HELP LATITUDE: x LONGITUDE: y' and will dispatch it using the SmsManager.getDefault().sendTextMessage() [10]. EMERGENCY ALARM does not add an

OnLongClickListener since there is no need to do so in crowded displays as there is a MediaPlayer playback of R.raw.alarm as soon as the button is held.

IV. The mathematical and algorithmic formulation.

A. Algorithmic Workflow

Algorithm 1 is the runtime execution of MainActivity2 from dashboard initialization to multi-channel SOS dispatch:

2) Create LocationManager and MediaPlayer(R.raw.alarm).

2) Ask for SEND_SMS, READ_SMS and FINE/COARSE_LOCATION permissions.

3) Ask user for coordinates (x, y) using onGPS(); if disabled read the coordinates from the cache using startTrack().

4) If you press EMERGENCY!!, you will query Cursor C from mylist.db, if C.count>0, you will execute ACTION_CALL(tel:1000) and send a broadcast SMS(x, y) to all the contacts in the list.

5) When EMERGENCY ALARM is pressed and held for > τ_long (Δt_press ≥ τ_long): MediaPlayer siren is started.

B. Analytical Models

1) Authentication and Contact Query Complexity: Sign-in in DBHelper uses O(log N_u) SQLite B-tree time to look up the primary-key of an account on N_u registered accounts; SOS dispatch scans all k saved emergency contacts in mylist.db:

$$\text{Auth}(u, p) = I((u, p) \in \text{Dusers}) \quad (1)$$

2) End-to-End SOS Latency and Delivery Reliability: Total response latency T_SOS includes GPS acquisition T_GPS, SQLite cursor read T_DB, queuing SMS in sequence across k contacts, and voice call intent launch T_call. If the probability of delivering an SMS to recipient i is p_SMS(i), then the probability R_alert that at least one of the contacts gets the alert is:

$$TSOS = TGPS + TDB + \Sigma i=1..k\ TSMS(i) + Tcall \quad (2)$$

$$Ralert = 1 - \Pi i=1..k\ (1 - pSMS(i)) \quad (3)$$

The current implementation of MainActivity2.java has loadData() making a call to sendSms() within the cursor loop, and sendSms() iterating over all k records, resulting in quadratic message complexity C_curr(k). The existing sendSms()'s linear dispatch can be overcome by refactoring it to traverse the ArrayList a single time, therefore reducing its complexity to linear C_opt(k):

$$Ccurr(k) = k2 \rightarrow Copt(k) = k \quad (4)$$

4) Long-Press Siren Guard and Geodesic Position Error: The hold time Δt_press must be greater than the Android long press timeout τ_long (default is 500 ms). The Haversine distance between two GPS fixes (φ_1, λ_1) and (φ_2, λ_2) on earth radius R_E is as follows and the error for GPS is limited to R_95 for 95% probability:

$$\text{Siren} = I(\Delta tpress \geq \tau long) \quad (5)$$

$$d = 2RE \cdot \arcsin(\sqrt{(\sin2(\Delta\varphi/2) + \cos \varphi1 \cos \varphi2 \sin2(\Delta\lambda/2))}) \quad (6)$$

$$R95 \approx 1.73 \cdot \sigma GPS \quad (7)$$

5) Energy Consumption per SOS Event: Total battery energy E_SOS consumed during an emergency activation includes GPS fix power, cellular radio transmission and speaker amplification.

This results in the following equation for ESOS:This leads to the following expression for ESOS: $ESOS = PGPS \cdot TGPS + PGSM \cdot (Tcall + k \cdot TSMS) + Paudio \cdot Tsiren$ (8)

The security audit and verification process is conducted at the V level (code level).

A. Module-wise Functional Verification

All eight Java classes and six XML layouts are statically inspected for complete offline execution for user registration, Contact CRUD, GPS provider prompting, dialing the cellular, formatting the SMS, and playing back the alarm in the MediaPlayer. The behavior of each application module is summarized in Table IIII

TABLE III

Module-Level Functional Verification Summary

Module / Class	Verified Responsibility	Offline Status
MainActivity	3,000 ms splash & LoginActivity launch	Fully Offline
LoginActivity2 & DBHelper	SQLite user auth via login.db	Fully Offline
Register & DatabaseHandler	Add/delete/view contacts in mylist.db	Fully Offline
MainActivity2 (GPS)	GPS check & coordinate caching	Satellite Only
MainActivity2 (SOS)	ACTION_CALL(1000) + SmsManager alert	GSM Only
MainActivity2 (Alarm)	Long-press MediaPlayer R.raw.alarm	Fully Offline

B. Security and Reliability Hardening Findings

The prototype is successful in showing the concept of offline-first in action, but the inspection of the code shows five solid engineering gaps that need to be addressed before it goes to the public Play Store (Table IV). First, DBHelper keeps passwords as plain text strings in login.db; this can be done with salted hash of PBKDF2 or even SHA-256 to secure local credentials on breached gadgets. Second, MainActivity2 uses getLastKnownLocation(GPS_PROVIDER) which returns null if no other application has previously cached a last GPS fix. Using the existing FusedLocationProviderClient (Play Services Location 19.0.1) [8] and having location callbacks enabled ensures fresh coordinates.

Third, refactor loadData() and sendSms() to pass the contact list only once, thereby avoiding the duplicate SMS bug ($O(k^2)$) and avoiding carrier throttling. Fourthly, PendingIntent delivery callbacks in sendTextMessage() will tell you if each SMS message was delivered. Last but not least, an explicit Stop Alarm switch and Emergency override of device media volume makes the local siren audible even if it was muted by the phone.

TABLE IV

Code-Level Audit and Production Hardening Roadmap

Target File	Observed Issue	Recommended Fix
DBHelper.java	Plaintext passwords in login.db	Salted PBKDF2/SHA-256 hashing
MainActivity2 (GPS)	getLastKnownLocation() may be null	Active FusedLocationProviderClient
MainActivity2 (SMS)	Nested loop causes k^2 SMS sends	Single-pass ArrayList + PendingIntent
MainActivity2 (Alarm)	No stop button or volume override	Add Stop toggle & AudioManager max
MainActivity2 (Call)	Hardcoded helpline tel:1000	Configurable regional SOS (e.g. 112)

VI. CONCLUSION AND FUTURE SCOPE

Stree Raksha's offline-first Android emergency response system features dual persistence in SQLite, single-tap (cellular) calling and GPS-based SMS broadcasting, and a long-press local siren. The system eliminates the need for mobile data connectivity on the critical distress path, ensuring operation even in areas with poor connectivity where cloud-based safety apps are unable to work. Salted credential hashing, FusedLocationProviderClient active tracking, SMS deduplication with delivery receipts (when available), accelerometer shake-to-SOS triggers, and optional cloud telemetry sync when packet data exists will be added in the future..

REFERENCES

[1] R. S. Yarrabothu and B. Thota, "Abhaya: An Android app for the safety of women," in Proc. Annu. IEEE India Conf. (INDICON), Dec. 2015, pp. 1–4.

[2] D. Chand, S. Nayak, K. S. Bhat, S. Parikh, Y. Singh, and A. A. Kamath, "A mobile application for women's safety: WoSApp," in Proc. IEEE Region 10 Conf. (TENCON), Nov. 2015, pp. 1–5.

[3] D. S. Prashanth, G. Patel, and B. Bharathi, "Research and development of a mobile based women safety application with real-time database and data-stream network," in Proc. Int. Conf. Circuit, Power Comput. Technol. (ICCPCT), Apr. 2017, pp. 1–5.

[4] P. Jothilakshmi, M. S. Rishikesh, and M. Y. Srikanth, "Cross-platform women safety mobile application with real-time alert system and GPS tracking," in Proc. Int. Conf. Electron. Syst. Intell. Comput. (ICESIC), 2026.

[5] M. Mahajan, K. Reddy, and M. Rajput, "Design and implementation of a rescue system for safety of women," in Proc. Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET), Mar. 2016, pp. 1955–1959.

[6] D. G. Monisha, M. Monisha, G. Pavithra, and R. Subhashini, "Women safety device and application—FEMME," Indian J. Sci. Technol., vol. 9, no. 10, pp. 1–6, Mar. 2016.

[7] D. Aggarwal, K. Banerjee, R. Jain, S. Agrawal, S. Mittal, and V. Bhatt, "An insight into Android applications for safety of women: Techniques and applications," in Proc. IEEE Delhi Section Conf. (DELCON), Feb. 2022, pp. 1–6.

[8] Google Developers, "FusedLocationProviderClient: Google Play services location API." Accessed: Nov. 2026. [Online]. Available: https://developers.google.com/location-context/fused-location-provider

[9] Android Developers, "LocationManager API reference." Accessed: Nov. 2026. [Online]. Available: https://developer.android.com/reference/android/location/LocationManager

[10] Android Developers, "SmsManager API reference." Accessed: Nov. 2026. [Online]. Available: https://developer.android.com/reference/android/telephony/SmsManager

[11] Android Developers, "Save data using SQLite." Accessed: Nov. 2026. [Online]. Available: https://developer.android.com/training/data-storage/sqlite

[12] Android Developers, "MediaPlayer overview." Accessed: Nov. 2026. [Online]. Available: https://developer.android.com/media/platform/mediaplayer