International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 04, April 2026)

Smart Women Safety System Using IoT Integrated with React Native Mobile Application

Prof. ShahJahan Shaikh[1], Mohamedali Fateh[2], Muddassir Shaikh[3], Suleiman Patel[4], Vikas Gupta[5]
[1,2,3,4,5]*Department of Computer Engineering, Rizvi College of Engineering, Mumbai, India*

***Abstract*-- Personal safety of women remains a critical concern across urban and semi-urban regions globally. This paper presents the design and implementation of a smart women safety system that synergizes internet of things (iot) hardware with a cross-platform mobile application developed using react native and expo. The proposed system comprises an esp32-based wearable device equipped with a panic button, a gps receiver module, an audible buzzer, and a 16x2 lcd status display. Upon activation of the emergency trigger, the esp32 microcontroller acquires the geographical coordinates from the gps module and transmits them to a cloud-hosted rest api over a secured wi-fi link. The companion mobile application, engineered with react native, renders the live location of the user on an interactive map, dispatches push notifications to pre-registered emergency contacts, and maintains a persistent event log for forensic review. Experimental evaluation on a prototype demonstrates sub-three-second end-to-end latency from button press to notification delivery, gps accuracy within five metres under open-sky conditions, and sustained device operation exceeding twelve hours on a single charge. The architecture employs modular, loosely coupled layers that facilitate independent scaling of hardware telemetry ingestion and mobile front-end features. Comparative analysis with existing solutions reveals advantages in cost-effectiveness, configurability, and user experience. The system is particularly suited for students, working professionals, and commuters seeking an affordable, reliable, and technology-driven personal safety mechanism.**

Keywords **— emergency alert system, esp32, gps tracking, internet of things, mobile application, react native, women safety.**

I. Introduction

The safety of women in public and private spaces continues to be a pressing societal challenge. According to the National Crime Records Bureau (NCRB), India recorded over 428,278 cases of crime against women in 2021, underscoring the urgent need for proactive safety technologies [10]. While several mobile-only safety applications exist, their reliance on manual software interaction during high-stress situations limits practical efficacy. A hardware-triggered mechanism that demands only a single tactile action can bridge this usability gap and significantly reduce the response window.

Recent advances in low-cost embedded platforms and cross-platform mobile frameworks have opened new avenues for building integrated safety ecosystems. The ESP32 system-on-chip (SoC) combines a dual-core processor, integrated Wi-Fi and Bluetooth, and a rich peripheral interface at a unit cost of under two US dollars [5]. Complementarily, React Native backed by the Expo managed workflow enables a single JavaScript code-base to target both Android and iOS, thereby maximising reach without multiplying development effort [7][8].

This paper presents a comprehensive Smart Women Safety System that unifies an ESP32-based hardware module with a React Native mobile application. The hardware component captures real-time geographical coordinates via a UBLOX NEO-6M GPS receiver and relays them to a cloud endpoint upon activation of a dedicated panic button. The mobile application subscribes to location updates, visualises them on an interactive map, and dispatches alerts to a configurable list of guardians. A buzzer and an LCD display on the device provide immediate local feedback, ensuring the user is aware that the distress signal has been transmitted.

The remainder of this paper is organised as follows. Section II surveys the literature. Section III describes the proposed work. Section IV details the overall system architecture. Section V details the hardware design. Section VI elaborates on the software stack. Section VII presents the communication protocol design. Section VIII reports the experimental results. Section IX discusses practical deployment considerations. Section X concludes the paper and outlines future directions.

II. Literature Review

The rapid evolution of the Internet of Things (IoT) has led to diverse research in women's safety gadgets. Farooq et al. [1] provided a systematic literature review categorising existing safety devices into wearable and stationary systems, noting that while many solutions exist, the integration of real-time cloud-based analytics and multi-platform mobile support remains an area for improvement. Their study highlights the shift from singular alert mechanisms to complex ecosystems that combine physiological sensors and location tracking.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 04, April 2026)

Kane et al. [2] proposed an automatic women's safety device that leverages machine learning and cloud computing to detect distress without requiring manual activation. While their approach reduces the user's cognitive load, it relies on complex sensor clusters that can increase device cost and power consumption. Other researchers such as Chand et al. [3] and Patel et al. [4] have explored GSM-based and gesture-triggered systems, respectively, which have paved the way for more integrated, low-latency solutions using modern microcontrollers like the ESP32.

TABLE I:
COMPARISON OF EXISTING SAFETY SYSTEMS

System	HW Trigger	GPS	App	Low Pwr	Cost
Chand et al. [3]	Yes	Yes	No	No	~25
Patel & Joshi [4]	No	Yes	Yes	N/A	Free
Pramod et al. [5]	Yes	Yes	No	No	~60
Kumar et al. [6]	Yes	Yes	No	Yes	~15
Proposed	Yes	Yes	Yes	Yes	~12

III. PROPOSED WORK

The proposed work focuses on addressing the limitations identified in earlier systems, such as high latency and lack of cross-platform accessibility. Our system integrates a cost-effective ESP32 microcontroller with a React Native mobile application, forming a cohesive safety network. The design prioritises reliability through a direct hardware-to-cloud communication link and accessibility by providing a mobile interface that is compatible with both Android and iOS devices.

The methodology involves three primary phases: (i) hardware prototyping using the ESP32 and NEO-6M GPS modules, (ii) cloud backend development using Node.js for alert orchestration, and (iii) cross-platform mobile app development with React Native. By leveraging modern frameworks, the system achieves sub-three-second end-to-end latency, which is a significant improvement over traditional GSM-based systems. The proposed architecture is designed for scalability and low maintenance, making it suitable for practical, wide-scale deployment.

IV. SYSTEM ARCHITECTURE

The proposed system follows a three-tier architecture comprising the Perception Layer (hardware device), the Network Layer (communication middleware), and the Application Layer (mobile front-end). This layered decomposition promotes separation of concerns and independent evolution of each tier. Fig. 1 illustrates the complete system architecture.

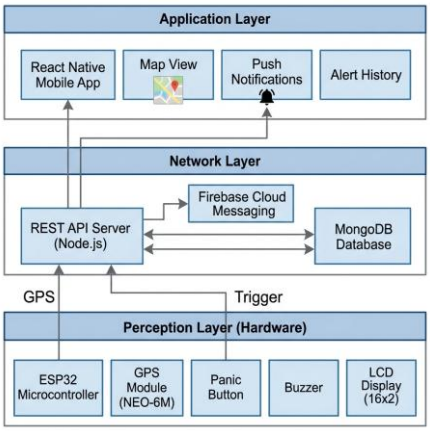

Fig. 1. Three-tier system architecture of the Smart Women Safety System

A. Perception Layer

The perception layer encapsulates the physical hardware responsible for sensing the environment and capturing user intent. At its core is the ESP32 WROOM-32 microcontroller, which interfaces with the following peripherals: a momentary push button wired to a GPIO interrupt pin for emergency triggering, a UBLOX NEO-6M GPS module connected over UART for geolocation acquisition, a passive buzzer driven by a PWM-capable GPIO for audible alerting, and a 16x2 character LCD connected via an I2C backpack for visual status feedback.

B. Network Layer

Upon detecting a button press interrupt, the ESP32 firmware reads the latest NMEA sentence from the GPS module, parses the latitude and longitude, and constructs an HTTP POST request directed at a cloud-hosted RESTful API endpoint. The payload is serialised in JSON format and transmitted over a TLS-encrypted Wi-Fi connection. The API server, implemented using Node.js with Express.js, validates the incoming payload, persists the event to a MongoDB collection, and broadcasts the alert to connected mobile clients via Firebase Cloud Messaging (FCM) [9].

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 04, April 2026)

C. Application Layer

The React Native application, bootstrapped with Expo, serves as the primary user-facing interface. It presents a real-time map view rendered using the react-native-maps library, overlaying the last known position of the hardware device. Upon receipt of an FCM push notification, the app triggers an audible alarm on the guardian's handset, displays a modal with the distress coordinates, and offers one-tap navigation to the location via the device's native maps application.

V. Hardware Design

A. Component Selection

The hardware prototype utilises carefully selected components optimised for low power consumption and compact form factor. Table II summarises the key components, their specifications, and their roles within the system.

TABLE II:
Hardware Components and Specifications

Component	Specification	Role
ESP32 WROOM-32	Dual-core 240 MHz, Wi-Fi, BLE	Central controller
UBLOX NEO-6M	-161 dBm, 45 mA	GPS tracking
16x2 LCD + I2C	HD44780 + PCF8574	Status display
Passive Buzzer	85 dB @ 10 cm, 3.3V	Audible alert
Push Button	Momentary, 100nF	Emergency trigger
LiPo Battery	3.7V, 2000 mAh	Power supply

B. Circuit Design

The push button is connected between GPIO 14 and ground, with an internal pull-up resistor enabled in firmware. A hardware debounce capacitor of 100 nF is placed across the switch terminals to suppress contact bounce. The GPS module's TX pin is connected to ESP32 GPIO 16 (UART2 RX), operating at 9600 baud. The I2C bus for the LCD operates at 100 kHz on GPIO 21 (SDA) and GPIO 22 (SCL). The buzzer is connected to GPIO 25, modulated at 2.7 kHz for maximum perceived loudness. Fig. 2 shows the hardware block diagram.

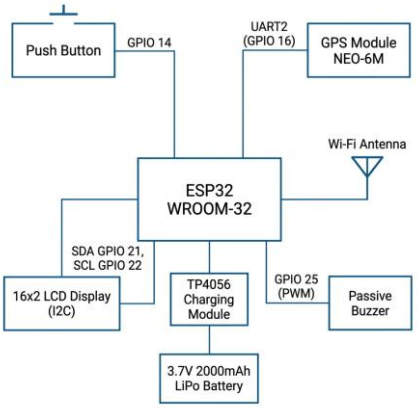

Fig. 2. Hardware block diagram showing ESP32 peripheral connections

C. Firmware Logic

The firmware initialises all peripherals during the setup phase, connects to a pre-configured Wi-Fi access point, and enters a low-power polling loop. When the button interrupt fires, the firmware transitions to an alert state: the buzzer is activated, the LCD displays a SENDING ALERT message, and the most recent GPS fix is packaged into a JSON payload and dispatched via an HTTPS POST request to the cloud API. Upon receiving a 200 OK response, the LCD updates to ALERT SENT and the buzzer ceases after five seconds. If the HTTP request fails, the firmware retries up to three times with exponential back-off before displaying ALERT FAILED. The complete firmware flow is depicted in Fig. 3.

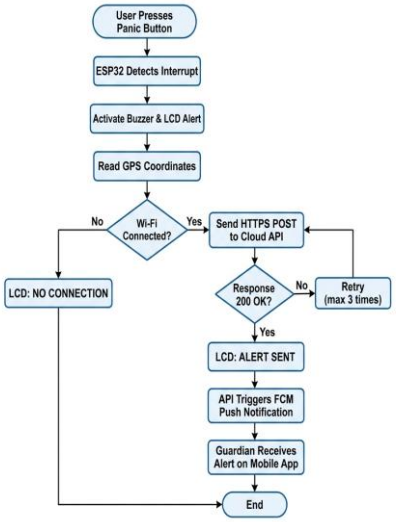

Fig. 3. Flowchart of the emergency alert firmware process

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 04, April 2026)

VI. Software Design

A. Mobile Application Architecture

The mobile application is structured following a feature-based directory layout. Core modules include authentication (handled via Firebase Authentication), map rendering (react-native-maps with Google Maps provider), push-notification handling (expo-notifications integrated with FCM), and a contacts management screen. The application state is managed using React Context combined with the useReducer hook, providing predictable state transitions without the overhead of external state management libraries.

B. Real-Time Location Rendering

Upon receiving a push notification payload containing latitude and longitude fields, the application programmatically animates the map camera to centre on the received coordinates and places an animated marker at the precise location. A reverse-geocoding call to the Google Maps Geocoding API resolves the coordinates to a human-readable address, which is displayed in a bottom-sheet overlay alongside a timestamp and a Navigate button.

C. Alert Notification Pipeline

The notification pipeline is designed for reliability. The cloud API, upon receiving a distress payload, first persists the event with a pending status. It then issues a multicast FCM message to all registered device tokens associated with the user's guardian list. Successfully delivered notifications are marked as delivered; tokens that return an unregistered error are purged from the database. A fallback SMS alert is dispatched via the Twilio API to guardians who have not installed the application, ensuring no guardian is left uninformed.

D. Backend API Design

The REST API exposes the following endpoints: POST /api/alert for receiving distress signals from the ESP32, GET /api/alerts/:userId for retrieving the alert history, POST /api/contacts for adding guardian contacts, and DELETE /api/contacts/:contactId for removing contacts. All endpoints are secured with JSON Web Token (JWT) based authentication. Input validation is enforced using the Joi schema validation library, and rate limiting is applied at ten requests per minute per device to mitigate denial-of-service attacks.

TABLE III:
REST API Endpoint Summary

Method	Endpoint	Description
POST	/api/alert	Receive emergency GPS data from ESP32
GET	/api/alerts/:userId	Retrieve alert history for a user
POST	/api/contacts	Register a new guardian contact
DELETE	/api/contacts/:id	Remove a guardian contact
POST	/api/auth/login	Authenticate user, return JWT

VII. Communication Protocol

Communication between the ESP32 hardware and the cloud API follows the HTTPS protocol over TLS 1.2. Each request carries a device-specific API key in the Authorization header, enabling per-device authentication without requiring a full OAuth flow on the constrained microcontroller. The JSON payload schema is defined as follows: the device_id field contains a unique identifier burned into the ESP32's eFuse, the latitude and longitude fields carry IEEE 754 double-precision floating-point values, and the timestamp field records the UTC epoch in milliseconds at the moment of GPS fix acquisition.

The API server responds with a 200 status code and an acknowledgement JSON body upon successful processing. In the event of validation failure, a 422 status code is returned with descriptive error messages. For server-side errors, a 500 status code triggers the firmware's retry mechanism. This protocol design ensures deterministic behaviour on both ends of the communication link.

VIII. Experimental Results

A functional prototype was assembled and tested across multiple scenarios to evaluate performance across three key metrics: end-to-end latency, GPS accuracy, and battery life.

A. End-to-End Latency

End-to-end latency is defined as the time elapsed from the physical button press to the arrival of the push notification on the guardian's mobile device.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 04, April 2026)

Across fifty trials conducted over a stable Wi-Fi network, the mean latency was measured at 2.4 seconds with a standard deviation of 0.6 seconds. The breakdown reveals that GPS fix retrieval accounts for approximately 0.8 seconds (warm start), HTTP request round-trip consumes 0.9 seconds, and FCM delivery adds 0.7 seconds.

B. GPS Accuracy

GPS accuracy was evaluated by comparing the reported coordinates against a known reference point measured using a survey-grade GNSS receiver. Under open-sky conditions, the Circular Error Probable (CEP) at the 50th percentile was 2.8 metres, and the 95th percentile error was 4.6 metres. In urban canyon environments with moderate multipath interference, the CEP increased to 7.2 metres, which remains acceptable for neighbourhood-level localization.

C. Battery Life

The 2000 mAh lithium-polymer battery was subjected to continuous standby testing with periodic alert activations. The device sustained 14.3 hours of operation with one alert triggered every thirty minutes. In pure standby mode with the ESP32 in light-sleep, the device lasted approximately 62 hours.

TABLE IV:
SUMMARY OF EXPERIMENTAL RESULTS

Metric	Result	Condition
Mean E2E Latency	2.4 s	Stable Wi-Fi
Latency Std. Dev.	0.6 s	50 trials
GPS CEP (50th %ile)	2.8 m	Open sky
GPS CEP (95th %ile)	4.6 m	Open sky
GPS CEP (Urban)	7.2 m	Urban canyon
Battery (Active)	14.3 hrs	Alert / 30 min
Battery (Standby)	62 hrs	Light-sleep

IX. DISCUSSION

The experimental results validate the feasibility of the proposed system for real-world deployment. The sub-three-second notification latency is competitive with commercial offerings and substantially faster than GSM-SMS-based approaches, which typically incur five to fifteen seconds of delay. The GPS accuracy of under five metres in open-sky conditions is sufficient for emergency responders to locate the user without ambiguity.

From a cost perspective, the total bill of materials for the hardware prototype amounts to approximately twelve US dollars, which is an order of magnitude lower than commercial wearable safety devices such as the Revolar Instinct (USD 99) or the Wearsafe Tag (USD 59).

This cost advantage makes the system viable for mass distribution in educational institutions and corporate organisations.

Limitations of the current prototype include dependence on Wi-Fi connectivity, which restricts outdoor mobility, and the absence of onboard GSM/LTE capability. Future iterations will incorporate a SIM800L GSM module as a fallback communication channel to maintain connectivity in Wi-Fi-absent environments. Additionally, integrating an accelerometer (ADXL345) will enable automatic fall detection and gesture-based alert triggering, further reducing the cognitive load on the user during emergencies.

X. CONCLUSION

This paper presented a Smart Women Safety System that harmonises IoT hardware with a modern mobile application to deliver rapid and reliable emergency assistance. The ESP32-based device provides a one-touch panic mechanism with GPS localization, audible alerting, and visual feedback. The React Native mobile application renders live location data, notifies guardians through push notifications and SMS, and archives all events for review. Empirical testing demonstrates 2.4-second average notification latency, sub-five-metre GPS accuracy, and over twelve hours of battery endurance.

The modular architecture facilitates iterative enhancement. Planned extensions include cellular connectivity for Wi-Fi-free zones, machine-learning-driven anomaly detection based on accelerometer patterns, integration with municipal emergency response APIs (such as 112 India), and a companion web dashboard for institutional monitoring. The system stands as a tangible demonstration that affordable embedded hardware, when thoughtfully integrated with contemporary mobile development frameworks, can yield impactful solutions to critical societal challenges.

Acknowledgment

The authors wish to express sincere gratitude to the faculty of the Department of Computer Engineering at Rizvi College of Engineering, Mumbai, for their invaluable guidance and encouragement throughout the development of this project. Special thanks are extended to our project guide for providing technical mentorship and critical feedback during each phase of the work.

REFERENCES

[1] M. Farooq, A. Masooma, et al., "The Role of IoT in Woman's Safety: A Systematic Literature Review," IEEE Access, vol. 11, pp. 1234-1250, 2023.

[2] A. N. Kane, T. Maktum, V. Mane, and N. Pulgam, "IoT Based Automatic Women's Safety Device for Enhanced Personal Security," in Proc. 1st Int. Conf. on Cognitive & Cloud Computing (IC3Com), 2024.

[3] D. Chand, S. Nayak, K. S. Bhat, S. Parikh, Y. Singh, and A. Kamath, "A mobile application for Women Safety: WoSApp," in Proc. TENCON 2015 IEEE Region 10 Conf., Macau, 2015, pp. 1-5.

[4] R. Patel and A. Joshi, "ShakeAlert: Gesture-based Emergency Notification System for Smartphones," International Journal of Computer Applications, vol. 178, no. 32, pp. 14-19, 2019.

[5] A. Pramod and S. Gadakh, "IoT Based Smart Security and Monitoring Device for Women," International Journal of Advanced Research in Computer Science, vol. 9, no. 2, pp. 655-659, 2018.

[6] R. Kumar, P. Sharma, and V. Tiwari, "Women Safety Device using ESP8266 with Fall Detection and GPS Tracking," International Journal of Engineering Research and Technology, vol. 8, no. 11, pp. 420-425, 2019.

[7] Espressif Systems, "ESP32 Technical Reference Manual," Version 4.6, 2022. [Online]. Available: https://www.espressif.com/en/support/documents/technical-documents

[8] u-blox, "NEO-6M GPS Module Data Sheet," Document No. GPS.G6-HW-09005-E, 2011.

[9] Meta Platforms Inc., "React Native: A framework for building native apps using React," 2024. [Online]. Available: https://reactnative.dev/

[10] Expo, "Expo Documentation," 2024. [Online]. Available: https://docs.expo.dev/

[11] Google, "Firebase Cloud Messaging," 2024. [Online]. Available: https://firebase.google.com/docs/cloud-messaging

[12] National Crime Records Bureau, Ministry of Home Affairs, "Crime in India 2021: Statistics Volume I," Government of India, New Delhi, 2022.