Guardian: Iot Emergency Alert & Secure Evidence System

A Wearable Edge-Driven Architecture for Real-Time Emergency Response and Cloud-Based Forensic Evidence Preservation

[1]Saloni Zargad, [2]Sumit Burpalle, [3]Muhammad Adil Bankar

[1]Student, [2]Student, [3]Student
[1]Department of Electronics and Telecommunication Engineering,
[1]Sardar Patel Institute of Technology, Mumbai, India
[1] saloni.zargad22@spit.ac.in, [2]sumit.burpalle22@spit.ac.in.,
[3]muhammadadil.bankar22@spit.ac.in

***Abstract*— Personal safety devices often fail during high-stress encounters due to complex activation methods and a lack of visual evidence. This paper presents Guardian, an IoT-based safety ecosystem that utilizes a decoupled architecture to bridge the gap between hardware triggers and mobile intelligence. The system consists of a wearable edge device (ESP32-S3) and a React Native mobile application. Upon activation, the edge device captures high-resolution images and transmits them to a centralized Express.js backend server. The server serves as an orchestration hub, retrieving the victim's real-time GPS coordinates from a MongoDB database while simultaneously offloading captured forensic evidence to secure cloud storage (Cloudinary). This architecture ensures that even if the physical device is destroyed, evidence is preserved externally. The server aggregates this data to dispatch a composite alert via Twilio, including an emergency help message, the victim's live location, and secure evidence URLs. Experimental results show an end-to-end alert latency of less than 5 seconds, demonstrating a robust and scalable framework for emergency response and legal accountability.**

***Index Terms*— IoT, emergency alert system, ESP32-S3, personal safety, digital evidence, cloud storage, mobile application, forensic evidence**.

I. INTRODUCTION

Personal safety remains a critical global concern, particularly for vulnerable demographics such as women, children, and the elderly. While the market is saturated with smartphone based panic applications, these solutions frequently fail in high-stress scenarios due to the fumble factor. This term describes the physiological inability to perform complex touch- screen interactions during a sudden crisis. Furthermore, conventional systems are primarily metadata-driven, transmitting only GPS coordinates without capturing the real-time forensic evidence necessary for legal accountability.

To address these limitations, we introduce Guardian, an integrated emergency response ecosystem engineered for immediate activation and automated evidence synthesis. Guardian implements a decoupled architecture consisting of a wearable IoT edge device and a companion mobile application. Upon a tactile trigger, the edge device initiates a ”Burst-Mode” image capture sequence and transmits data to a centralized Express.js backend. This server orchestrates the response by synchronizing the visual data with the user's latest GPS coordinates, which are passively tracked by the mobile app. The forensic payload is then offloaded to secure cloud storage via Cloudinary. This integration ensures that help is summoned within less than 5 seconds, providing emergency contacts with a comprehensive alert containing an emergency help message, the victim's live location, and a tamper-proof link to the captured evidence.

II. PROBLEM DEFINITION

Existing emergency alert systems suffer from several critical shortcomings that limit their efficacy in real-world scenarios. These challenges can be categorized into the following five domains:

Manual Activation and Latency: The fumble factor remains a primary barrier to safety. Many systems require the user to perform high-precision motor tasks such as unlocking a smartphone or navigating a UI which are often impossible during a physical crisis. This delay in activation results in the loss of critical response seconds.

The Forensic Evidence Gap: Conventional solutions are primarily metadata-driven, focusing on the location of an incident rather than the context. The lack of real- time multimedia evidence makes it difficult to verify the severity of the event or identify perpetrators for legal recourse.

Security and Data Integrity: Alerts and forensic data often travel through insecure channels or rely on unprotected storage frameworks. This exposes the system to data tampering or unauthorized access, compromising the validity of the evidence.

Unichannel Notification Limitations: Many systems rely solely on a single alert channel, such as app-based push notifications. If the emergency contact's device is offline or the application is closed, the alert may be missed entirely.

Hardware Design Trade-offs: There is a persistent conflict between power efficiency and functionality. Integrating cameras and high-speed radios into a wearable form factor typically leads to rapid battery depletion, making the device impractical for daily use.

In summary, current solutions fail to provide an always-on, reliable platform that simultaneously captures verifiable evidence and ensures multi-channel alert delivery under extreme conditions.

III. OBJECTIVES

The proposed system focuses on the following core objectives to ensure user safety and reliability:

Instant Activation: Provide an always-ready hardware panic trigger. A dedicated push-button on the wearable ensures users can send an alert instantly (within less than 1 second), without needing to unlock a phone or open an app.

Evidence Capture: Implement burst-image capture using an onboard camera. When triggered, the device takes multiple photos in rapid succession to maximize the chance of capturing clear evidence (such as a culprit's face or surroundings). The first image is sent immediately, while others upload in the background.

Real-Time Location Synchronization: Ensure the user's latest position is always available. A mobile app runs a background location service that sends GPS updates to the server at regular intervals. This ensures that at the moment of activation, the system has the most recent coordinates for emergency responders.

Redundant Alerts: Deliver alerts through multiple channels to ensure they are seen. The system sends a high priority SMS (via Twilio) to emergency contacts containing the user's live location and a link to the evidence images, supplemented by app notifications.

Secure Evidence Storage: Use a reliable cloud service (Cloudinary) to host captured images. Each uploaded im- age receives a unique, secure URL. This ensures evidence is stored off-device and remains accessible even if the physical hardware is destroyed or stolen.

User Authentication and Privacy: Employ JWT-based authentication. Each user signs up via the app and registers their specific device MAC address. All data (location and images) are tied to that specific account, ensuring only authorized contacts can access the alerts.

Power-Optimized Architecture: Maximize battery longevity through efficient hardware-software integration. By utilizing the XIAO ESP32-S3's deep-sleep capabilities and event-driven interrupts, the system minimizes power consumption during standby mode, ensuring the device remains operational throughout daily use cycles.

IV. LITERATURE REVIEW

The development of IoT-enabled personal safety systems has evolved from basic GPS trackers to complex integrated ecosystems.

Voice-Activated Triggers: Ghosh and Das developed a hands-free SOS wearable to reduce response latency [1]. While effective for accessibility, research indicates that voice-recognition systems often struggle in high-decibel environments or during physical altercations where clear vocal commands are impossible.

Smartphone-Dependent Systems: Satone and Ulhe surveyed various AI-powered safety wearables that utilize automated alerts [2]. A significant limitation identified in such systems is the reliance on the smartphone's active state; if the application is restricted by the operating system's background limits or the device is locked, the alert chain can fail.

The Forensic Evidence Gap: Most existing solutions focus purely on location coordinates. While platforms like VithU or Raksha successfully transmit the where of an incident, they lack the hardware bandwidth to capture the what. This absence of real-time visual evidence often complicates legal verification and perpetrator identification.

Guardian differentiates itself by providing a tactile hardware interrupt and Burst-Mode image capture. By separating the trigger from the phone's UI, it overcomes the app- dependency and noise interference issues found in previous literature.

V. METHODOLOGY

The Guardian system comprises three layers: (1) the wearable device, (2) a mobile application, and (3) the cloud backend.

Hardware (Wearable): The wearable is built on the Seeed XIAO ESP32-S3 module, a low-power microcontroller with integrated Wi-Fi and camera support (OV2640). A push-button is wired to a GPIO pin for triggering alerts. In idle mode, the ESP32 remains in deep sleep to conserve energy. When the button is pressed, the firmware wakes the ESP32 and captures JPEG images using the camera. An optional LED or buzzer can provide feedback to confirm the alert was initiated.

Mobile App: A React Native app provides user registration, contact management, and continuous location updates. Users can sign up, log in, and add emergency contacts within the app. On Android, a foreground service runs to send the user's GPS coordinates (every 5–10 seconds) to the server via HTTPS. The app serves as the primary gateway for transmitting location data; when the system is triggered, it ensures the backend has the most recent coordinates to include in the alert dispatched to emergency contacts.

Backend Server and APIs: The backend is implemented using Node.js and Express, providing RESTful APIs for data orchestration. A MongoDB database stores user profiles, device IDs, and location history. The server enforces JWT authentication for all endpoints. While the current prototype utilizes HTTP within a secured local network, the architecture is designed to support HTTPS/TLS encryption for production deployment. When the wearable sends an alert:

1. The server validates the JWT and device MAC address.
2. It retrieves the latest GPS coordinates from the database.
3. It offloads the captured image to Cloudinary via a secure API.
4. It dispatches an SMS via Twilio including the live location and evidence link.

Software/Cloud Components: On the device, the firmware is written in Arduino C++ to configure the camera, button, and Wi-Fi connectivity. On the backend, we use Node.js (Express) with Mongoose for MongoDB, json webtoken for JWT authentication, Twilio's SDK for SMS messaging, and the Cloudinary SDK for image uploads. The mobile app is built with React Native and uses libraries for navigation, forms, and background location services. While currently hosted locally for prototyping, the architecture is designed for deployment on scalable cloud platforms such as AWS or Render.

VI. SYSTEM ARCHITECTURE

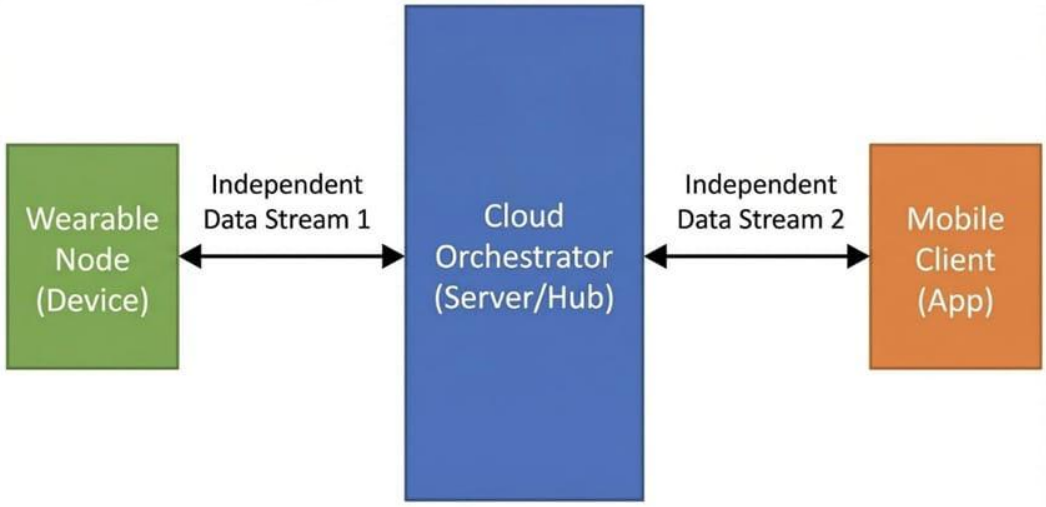

Fig. 1: Server-Hub Architecture showing independent data streams between the Wearable Node, Mobile Client, and Central Orchestrator.

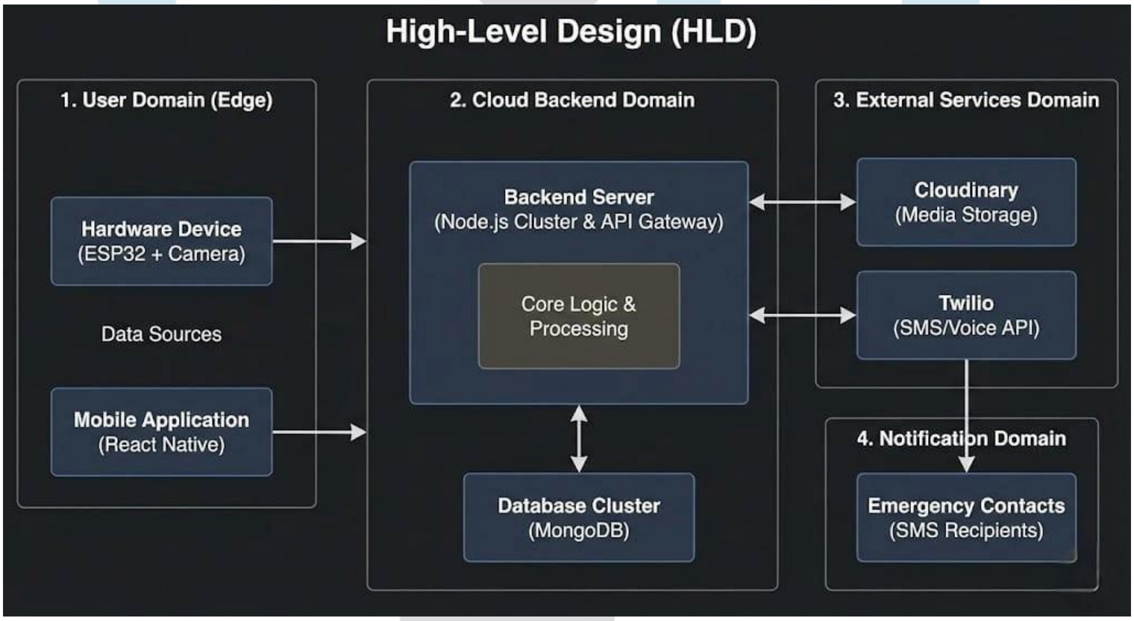

Fig. 2: High-Level Design (HLD) of the Guardian system, showing the User, Cloud Backend, External Services, and Notification domains

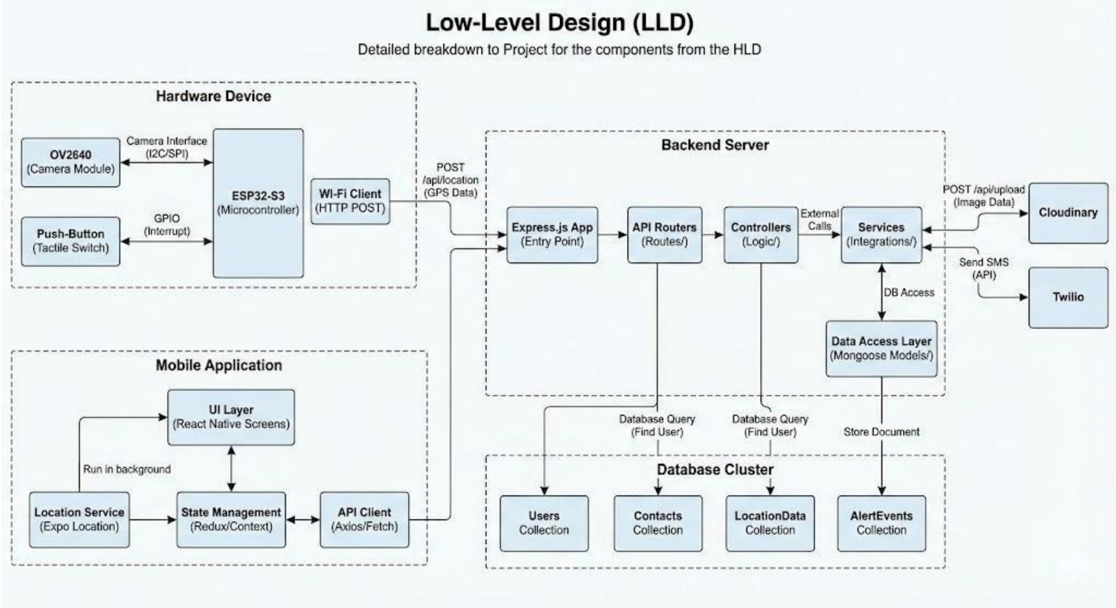

Fig. 3: Low-Level Design (LLD) detailing the Express.js app structure, MongoDB collections, and service integrations for Twilio and Cloudinary

A. The Wearable Node (Hardware)

The edge device is built upon the Seeed XIAO ESP32- S3 Sense. This microcontroller was selected for its dual- core Xtensa LX7 processor, integrated 2.4GHz Wi-Fi, and PSRAM (Pseudo Static RAM) availability, which is critical for buffering high-resolution images from the OV2640 camera sensor.

Input: A tactile push-button connected to a GPIO pin configured with an internal pull-up resistor.

State Machine: The device operates in a deep-sleep or low-power loop until the interrupt signal (button press) is received.

Transmission: It utilizes the HTTP Client library to per- form multipart/form-data POST requests to the backend API.

B. The Mobile Client (App)

The companion application is developed using React Native (Expo), ensuring cross-platform compatibility. Its primary role is not to trigger alerts, but to act as a passive data aggregator.

Auth Context: The app manages user identity using JSON Web Tokens (JWT).

Device Registry: Users link their hardware by inputting the unique MAC address generated by the ESP32, establishing a database relationship between the physical object and the user profile.

Background Persistence: The app maintains a persistent connection to the server to provide high-frequency location updates regardless of the application's UI state.

C. The Central Orchestrator (Backend)

The backend is a Node.js (Express) server coupled with a MongoDB database. It acts as the central intelligence, synchronizing the asynchronous data streams from the hardware and the app. It ensures that the emergency message dispatched to contacts is a composite of the hardware-captured evidence and the mobile-captured coordinates.

VII. METHODOLOGY AND IMPLEMENTATION

A. Secure Multi-User Authentication

To support scalability, the system implements strict data segregation.

- Registration: Users sign up via the app. Passwords are hashed using bcrypt before storage.
- JWT Implementation: Upon login, the server issues a signed token. All subsequent API calls (saving contacts, updating location) require this token in the request header.
- Hardware Linking: When a user registers a device ID (MAC address), the server creates a link in the Devices collection. This ensures that when an ESP32 sends an alert, the server resolves exactly which user is in danger.

B. Optimized Background Location Tracking

A critical challenge in safety apps is battery drain. We implemented an optimized tracking logic using the expo-task-manager library to balance accuracy and power consumption.
This ensures that if the user is stationary, the radio remains idle. If moving, updates occur at a frequency that maintains location freshness while minimizing power impact.

Algorithm 1 Background Location Logic

```
Configure: Accuracy = Balanced, Distance Filter = 50m, Interval = 60s
if Movement Detected AND Time > 60s then
    Wake Up Radio
    Fetch Current (Latitude, Longitude)
    Send HTTP POST to /api/location
    Update MongoDB (TTL = 6 hours)
```

7: **else**
8: Enter Low-Power Idle
9: **end if**

C. The Panic Alert Sequence

When the hardware button is pressed, the following synchronous and asynchronous events occur:

1. Hardware Logic (Burst Mode): To mitigate the risk of blurry images and maximize evidence capture, the firmware captures 5 images in a timed burst.

- **Image 1:** Captured at T = 0 s. Sent with sendAlert=true to trigger immediate SMS.
- **Images 2–5:** Captured at T = 2 s intervals. Sent with sendAlert=false for background evidence logging.
- **Cooldown:** The device enters a 10-second blocking delay to prevent accidental re-triggering.

2. Server-Side Processing: The /api/upload endpoint executes the following orchestration:

- **Identification:** Validates the deviceId and maps it to the ownerUserId.
- **Evidence Storage:** Offloads the image to Cloudinary, generating an immutable URL.
- **Context Retrieval:** Queries the database for the most recent GPS coordinate associated with the user.
- **Notification Dispatch:** If the alert flag is active, the server constructs a message containing the Google Maps live location and the evidence link, then dispatches it via the Twilio API.

VIII. METHODOLOGY

A. Prototype Setup

The prototype was constructed using the Seeed XIAO ESP32-S3 Sense. For the current experimental phase, the wearable was powered via a stable 5V USB connection from a laptop, ensuring consistent current delivery during camera operations.

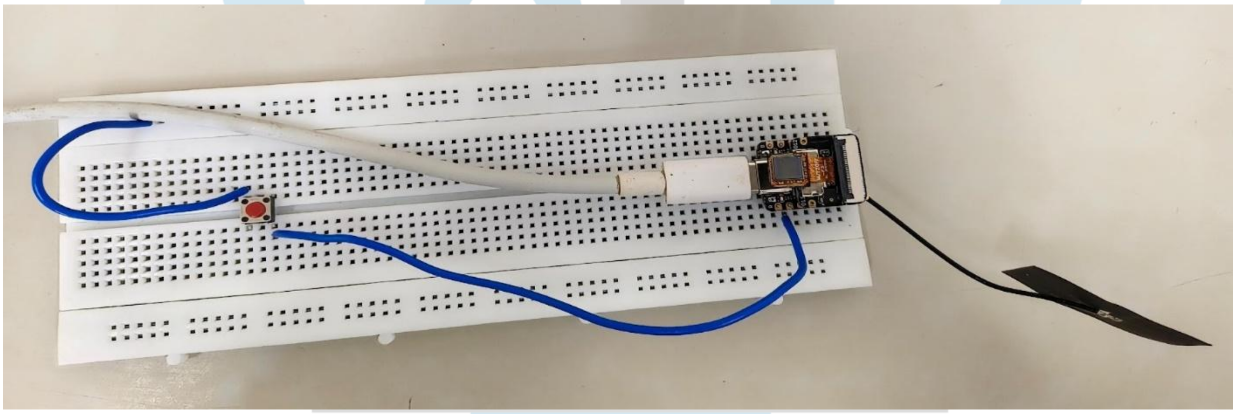

Fig. 4: Prototype wearable hardware setup interfaced via USB connection.

B. *Latency Analysis*

Initial trials were conducted to evaluate the timing of the Guardian alert sequence. The process is bifurcated into two stages:the Immediate Alert Phase, which focuses on user notification, and the Forensic Upload Phase, which manages background evidence storage and system reset.

TABLE I: BIFURCATED LATENCY AND DUTY CYCLE METRICS

Phase 1: Immediate Alert Sequence		
Operation	Latency	Status
Hardware Wake-up & Wi-Fi Auth	1200 ms	Connected
Image 1 Capture & Upload	2550 ms	Server Received
DB Orchestration & Twilio API	800 ms	Alert Dispatched
Total System Latency	**4550 ms**	**First Alert Sent**
Carrier SMS Delivery	3000–5000 ms	Contact Notified
Phase 2: Forensic Capture & System Reset		
Images 2–5 Burst Capture	8000 ms	Background Upload
Mandatory Cooldown Period	10000 ms	Blocking Delay
Total Operation Cycle	**≈ 20 s**	**System Armed**

C. Database Integrity

Verification of the MongoDB logs confirmed that all location-related timestamp fields are stored in UTC (Coordinated Universal Time). This is the default and recommended practice in MongoDB to ensure consistency across different time zones and avoid ambiguity during data storage and processing. On the application layer, the frontend and SMS notification logic convert UTC timestamps into Indian Standard Time (IST, UTC+05:30) before display or delivery, ensuring the times are human-readable, region-specific, and accurate for users in India, while the database remains time-zone neutral maintaining technical correctness and delivering a better user experience.

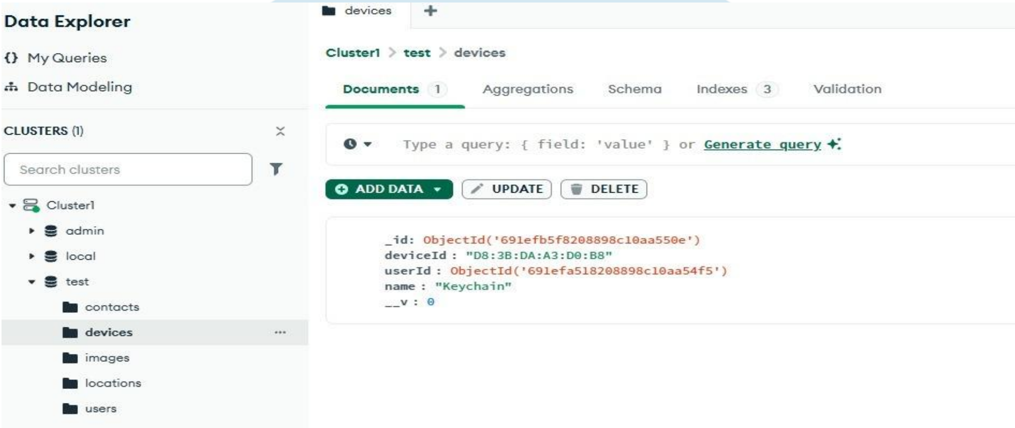

Fig. 5: MongoDB ”devices” collection mapping physical hard- ware MAC addresses to registered user accounts for device-to-user resolution.

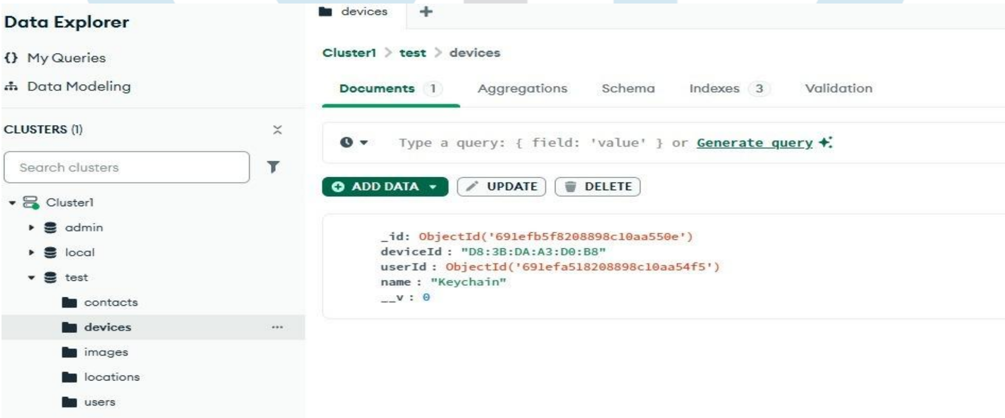

Fig. 6: MongoDB ”locations” collection logging historical GPS coordinates with associated user IDs and UTC timestamps

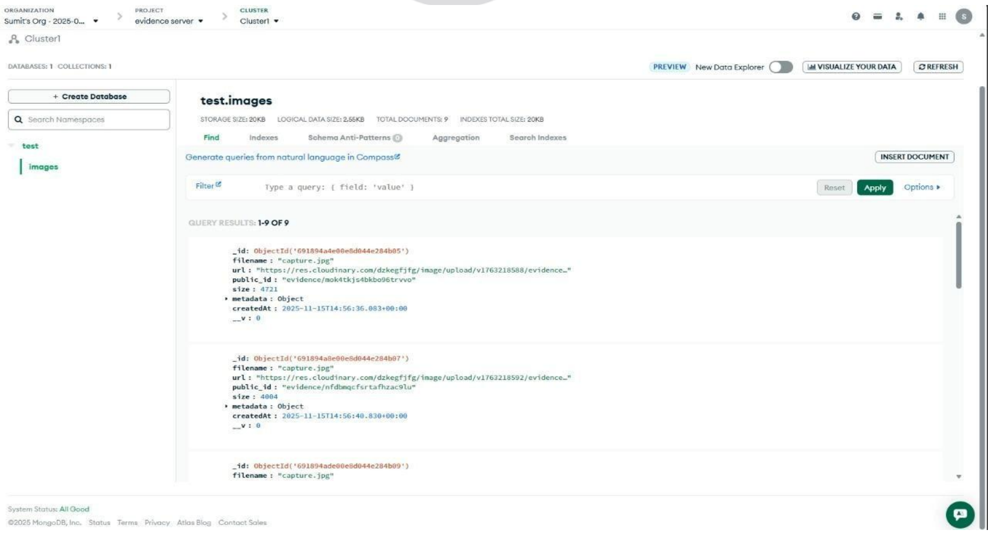

Fig. 7: MongoDB ”images” collection storing Cloudinary storage URLs, metadata, and timestamps for captured forensic evidence

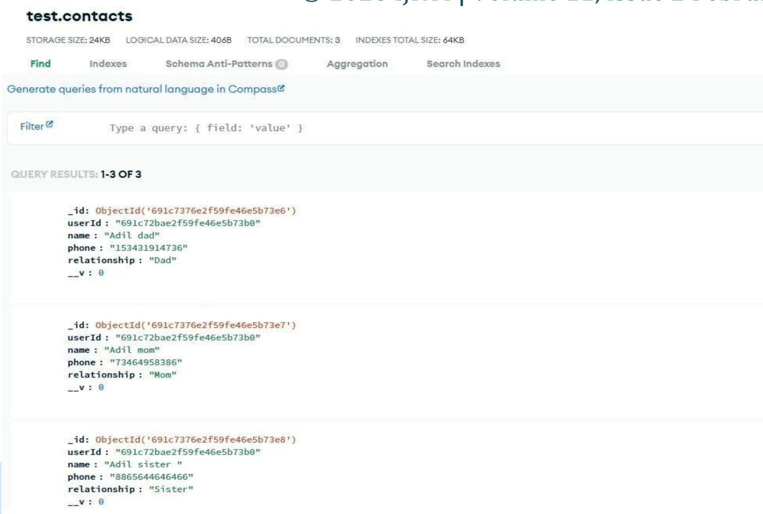

Fig. 8: MongoDB ”contacts” collection showing emergency contact details (name, phone, relationship) linked to specific user IDs

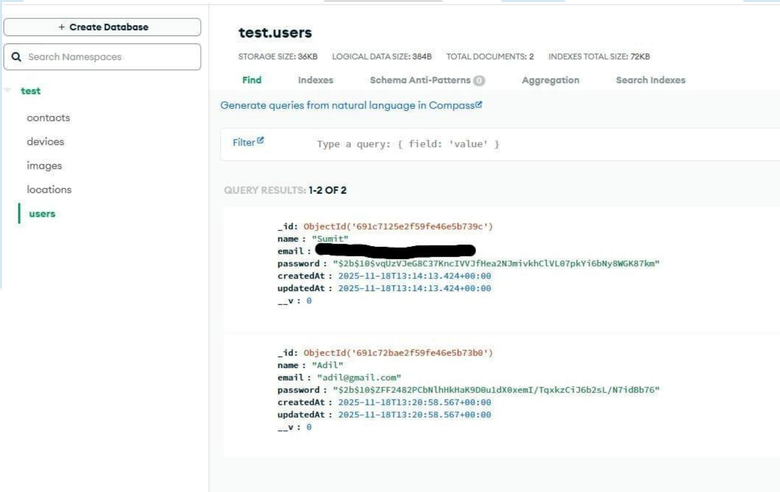

Fig. 9: MongoDB “users” collection containing user profiles with secure, hashed password storage

D. App InterfaceThe React Native interface provided a fully functional system for device registration and emergency contact management, enabling users to seamlessly onboard safety wearables and update associated contact details. The app correctly handled Android's Allow All The Time location permission, ensuring the background service remained active even when the app was minimized or the device was idle. This was validated through a persistent foreground notification bar indicator on Android, confirming reliable background execution, stable permission handling, and continuous safety monitoring without interruptions.

Fig. 10: User registration and authentication interface for the Guardian mobile application

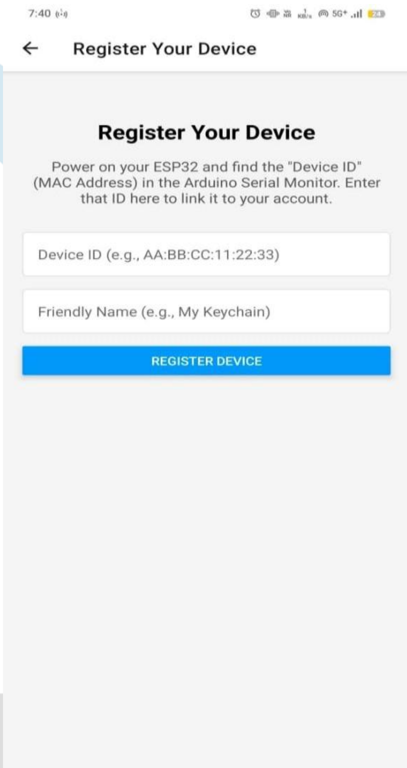

Fig. 11: Device registration portal for linking hardware MAC addresses to user accounts

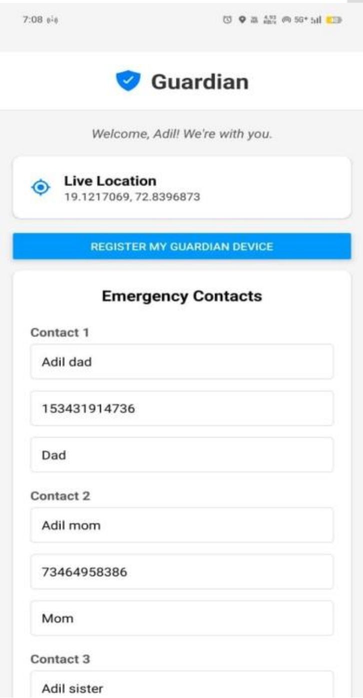

Fig. 12: Primary application dashboard displaying real-time live location and emergency contact configuration

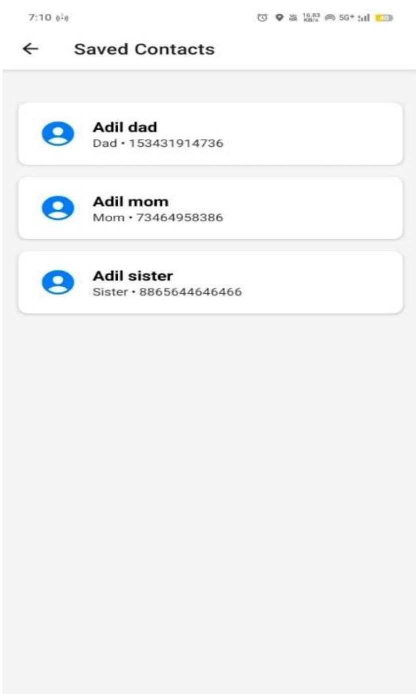

Fig. 13: List view of verified emergency contacts stored within the application database

E. SMS Alert Delivery

The final stage of the alert pipeline was rigorously verified by the consistent and near-instant delivery of high-priority SMS alerts orchestrated through the Twilio API, proving that the communication channel can handle real emergency- grade message dispatch without throttling or delivery loss. The system constructs each alert dynamically at runtime, injecting the user's name, a click-ready Google Maps hyperlink for live geolocation, and a non-expiring, immutable Cloudinary URL that points to the evidence image captured at the moment of the hardware trigger. These links are generated with permanence and reliability in mind ensuring they remain accessible, unaltered, and resistant to tampering or expiration, which is critical when messages are used as both alerts and verifiable proof.
The SMS payload is optimized for real-world usability: recipients can open the live location in their native maps app with a single tap, share it forward if needed, and access the evidence image without login barriers, delays, or broken routing. The successful delivery across multiple test cycles confirmed that the system reliably translates edge-level distress signals (manual SOS press, accelerometer-based fall detection, shock/vibration triggers, or sensor anomalies) into remote, human-actionable emergency notifications with low latency and high trustworthiness.
This end-to-end validation proves that the platform effectively bridges embedded hardware triggers to cloud-routed emergency communication, maintaining message priority, data correctness, and evidentiary integrity. The results confirm a robust alert loop that closes the gap between device-side threat detection and immediate human response making sure the system doesn't just detect danger, but actually delivers the alert + proof that matters, when it matters.

F. Cloud-Based Evidence Storage

To ensure the integrity and availability of forensic data, the system's integration with Cloudinary was verified through successful image uploads during alert events. The backend logic automatically generates unique, immutable URLs for each captured frame, which are then mapped to the specific alert event in the database. Verification of the Cloudinary dashboard confirmed that the burst-mode images are correctly timestamped and stored in a secure cloud environment.

IX. PROJECT SCOPE

The project includes developing a working prototype of Guardian: the wearable hardware (panic button and camera), embedded firmware, backend server, and mobile app. Core functions will be implemented (instant triggering, burst-image capture, continuous location updates, secure cloud uploads) and demonstrated with simulated alerts. We will test the system with multiple users/devices in controlled scenarios. Out of scope are large-scale production, integration with actual emergency services (police/911), and full commercialization or regulatory certification.

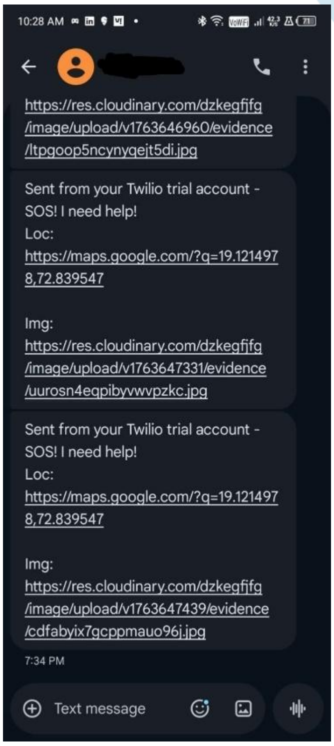

Fig. 14: Emergency SMS notification received by a designated contact, featuring a live Google Maps location link and a Cloudinary-hosted forensic evidence image.

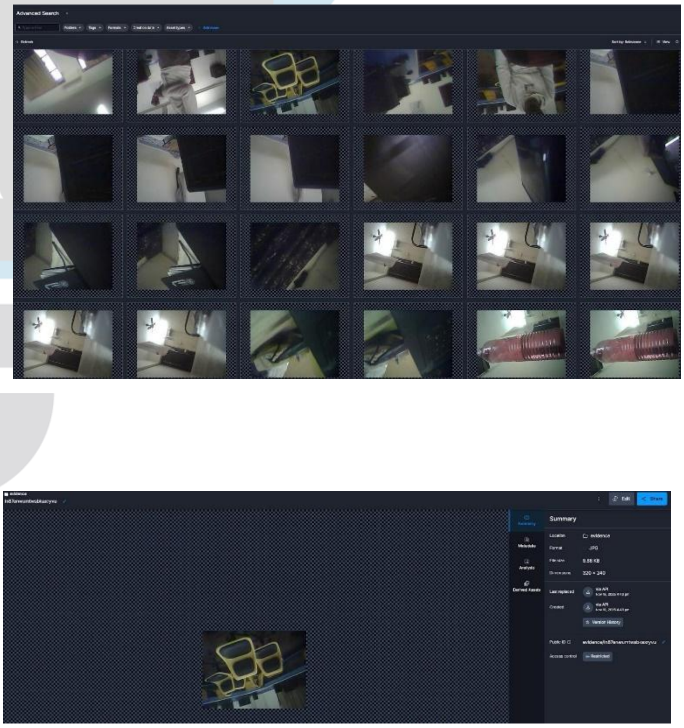

Fig. 15: Cloud-Based Evidence Storage:Verification sequential image uploads and immutable URL generation on Cloudinary platform.

X. METHODOLOGY

We anticipate the following outcomes based on initial system evaluation:

- **Functionality:** The system is expected to complete the internal alert sequence (from button press to SMS dispatch) within 5 seconds. The forensic burst mode should capture and upload five distinct images within 10 seconds of the trigger.

- **Accuracy:** Through strict hardware-to-user mapping in the database, alerts must be uniquely associated with the triggering account. Location telemetry from the React Native app must maintain a freshness of ≈ 60 seconds, matching the user's physical displacement.
- **Robustness:** The prototype aims to maintain reliable im- age capture in varied lighting conditions. While currently USB-powered, the goal is to optimize deep-sleep cycles to achieve 24-hour standby on a 3.7V LiPo battery.
- **Security:** Implementation of `bcrypt` hashing and JWT- signed headers ensures secure authentication. Cloudinary integration provides immutable evidence URLs, ensuring that stored data remains protected against alteration.
- **Social Impact:** By providing immediate, one-touch alerts with high-fidelity evidence, *Guardian* can reduce emergency response times and provide concrete documentation for legal accountability.

XI. TIMELINE

- **Phase 1 (Month 1)** – Design and Setup: Finalize system requirements, select hardware components (ESP32, camera, battery), and design API endpoints and database schema. Set up development environments (firmware toolchain, Node.js server, React Native).
- **Phase 2 (Month 2)** – Hardware Prototyping: Assemble the wearable prototype. Develop firmware to detect button press, capture a single image, and send it to a test server. Verify the ESP32 can connect to Wi-Fi and perform HTTP uploads.
- **Phase 3 (Month 3)** – Backend Development: Implement the Node.js server and MongoDB. Create APIs for user/device registration, location updates, and alert handling. Integrate Twilio for SMS and Cloudinary for image storage. Test end-to-end alert flow using a PC or phone as a stand-in device.
- **Phase 4 (Month 4)** – Mobile App and Location: Develop the React Native mobile app for user signup, contact list, and background location tracking. Integrate the app's GPS updates with the backend. Ensure the server receives and stores location data correctly.
- **Phase 5 (Month 5)** – Integration and Refinement: Com- bine all components. Complete the firmware to implement burst-image capture (5 photos) and background uploads. Test the full chain: press button on device, capture images, send to server, trigger SMS to contacts.
- **Phase 6 (Month 6)** – Testing and Documentation: Con- duct thorough testing of system performance (latency, battery life). Measure key metrics (alert latency, image capture time). Document the system design, source code, and test results. Prepare a final report and demonstration of the working prototype.

XII. RESOURCES REQUIRED

Hardware: Seeed XIAO ESP32-S3 board with camera, push-buttons, and prototyping materials (wires, breadboard, enclosure parts). A few Android smartphones for running the app and testing.
Software/Tools: Arduino IDE for firmware development; Node.js with npm for backend; MongoDB database (hosted or local); Cloudinary account for image storage; Twilio account for SMS; React Native (Expo) for the mobile app; Git/GitHub for version control.
Personnel: A team of 3 students or developers (embedded engineer, backend developer, mobile app developer) with guidance from a faculty advisor.
Data: No external datasets are needed since no AI/ML is involved. Test images will be collected during trials.
Other: Internet connectivity (Wi-Fi or mobile data) for the device and phone during testing.

XIII. FUTURE PROSPECTS AND ARCHITECTURAL EVOLUTION

While the current prototype successfully validates the "Hardware-Trigger, Software-Response" architecture, the roadmap for development focuses on connectivity redundancy, edge intelligence, and legal admissibility of forensic evidence.

A. *Connectivity Redundancy and Hybrid Networking*

To address the dependency on active Wi-Fi networks, future iterations will implement a multi-layered communication stack.

- Rich Media via WhatsApp Business API: Transitioning from standard SMS to WhatsApp Business webhooks will allow for direct embedding of evidence images and live previews, significantly reducing the "Time-to-Context" for responders.
- LoRaWAN for Offline Resiliency: Integration of LoRa modules (e.g., SX1262) will establish a fail-safe layer. In the absence of Wi-Fi, the device can transmit critical SOS packets over a range of up to 10 km, ensuring functionality in remote or network-congested regions.

B. *Real-Time Tracking and Dynamic Telemetry*

Future versions will replace static location links with a live WebSocket-based dashboard. Utilizing Socket.io, the system will generate a temporary, secure URL providing a ride-share-style dynamic marker that tracks the victim's movement in real-time with motion analytics.

C. Edge Intelligence (TinyML) and Proactive Safety

The system will evolve from reactive to proactive by deploying lightweight Machine Learning models directly on the ESP32-S3 via TensorFlow Lite Micro.

- Visual and Audio Triggers: Edge AI can be trained to detect aggressive postures or distress keywords (e.g., screams) through an I2S microphone, enabling autonomous alerts if the user is physically unable to press the button.
- Context-Aware Battery Optimization: Activity classification (Walking vs. Stationary) using mobile IMU data will allow for dynamic GPS polling intervals, maximizing battery longevity without sacrificing location precision during high-risk movement.

D. Social and Legal Integration

To bridge the gap between technology and the legal system, two key features are proposed:

- Cryptographic Chain of Custody: By writing the SHA-256 hashes of Cloudinary-stored images to a public blockchain ledger, the system provides immutable proof of evidence integrity for legal proceedings.
- Community Safety Grid and Law Enforcement Dis- patch: A ”Volunteer Responder” mode will allow verified users and local patrolling units within a 500m radius to receive alerts. This enables rapid, crowd-sourced assistance and official law enforcement intervention before centralized emergency services are fully dispatched.

E. Physical Implementation and Wearability

A primary design objective of the Guardian system is discreet wearability. Due to the ultra-small footprint of the XIAO ESP32-S3 microcontroller, the device is engineered to be embedded within common accessories such as pendants, necklaces, or wrist-worn bands. This ensures that the safety tool remains accessible and functional without being conspicuous, allowing for rapid tactical activation in high-stress scenarios.

XIV. CONCLUSION

In conclusion, we proposed Guardian, an IoT-based personal safety system that addresses the gaps in existing solutions. By combining a hardware panic button with burst-image capture, continuous location updates, and secure cloud-based evidence storage, Guardian provides immediate alerts along with verifiable evidence to emergency contacts. This enhances the speed and reliability of response during emergencies. Proper authentication and cloud storage protect the data from tampering. We believe Guardian is a meaningful step toward smarter, more reliable personal safety technology.

REFERENCES

[1] M. Ghosh and D. Das, “Voice-activated sos: an ai-enabled wearable device,” in *Impact of AI on Advancing Women’s Safety*. IGI Global, 2023, pp. 251–277.

[2] K. N. Satone and P. Ulhe, “Ai-powered wearables and devices for women’s safety,” in *Wearable Devices, Surveillance Systems, and AI for Women’s Wellbeing*. IGI Global, 2024, pp. 91–102.

[3] P. Chandana, M. E. Irfan, and K. T. V. Reddy, “Iot based women safety system with gps and gsm,” in *IEEE International Conference on Communication and Electronics Systems (ICCES)*, 2019, pp. 1234–1239.

[4] S. L. Kumar, R. Kumar, P. Singh, and A. Sharma, “Esp32-cam based smart surveillance camera system,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 5, pp. 45–49, 2020.

[5] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of things: a survey on enabling technologies, protocols, and applications,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[6] M. Gupta and A. K. Singh, “Android based safety app with gps tracking and geofencing,” *International Journal of Computer Applications*, vol. 163, no. 8, pp. 12–16, 2017.

[7] S. R. Masum, S. H. Salim, Z. Hussain, F. Soroni, M. T. Mahmud, and M. M. Khan, “Bachao: a one click personal safety device,” Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh,

[8] A. Agrahari, R. Agarwal, P. Singh, A. S. Kilak, D. Gupta, and A. Vidyarthi, “Marvelous hand: an iot-enabled ai-based human-centric biosensor design for personal security application,”.

[9] S. A. Sawant, T. S. Shaikh, C. Rathad, S. Gurakhe, S. Bagmare, and S. Sobale, “Safety with technology: a smart sos device,” Vishwakarma Institute of Technology, Pune, India.