Volume: 14 Issue: I Month of publication: January 2026
DOI: https://doi.org/10.22214/ijraset.2026.77086

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue I Jan 2026- Available at www.ijraset.com

IoT Accident Detection and Rescue

Om Pokale[1], Digvijay Nakhate[2], Neel Khamait[3], Avneesh Patil[4], Mr. Shivansh Shukla[5]
Department of Computer Engineering, Pimpri Chinchwad Polytechnic, Pune, India

Abstract: IoT Accident Detection and Rescue is a smart emergency response system that detects road accidents in real time and immediately initiates rescue communication to reduce response time and improve victim safety. The solution combines an Android mobile application built using Java/XML with an IoT edge device based on ESP32. The Android app provides secure user authentication, emergency contact management, and cloud synchronization using Firebase, ensuring that contact data remains available and updated across sessions. On the hardware side, the ESP32 integrates an accelerometer and camera to monitor sudden impact patterns and capture event context. A threshold-based accident detection algorithm triggers an emergency workflow when abnormal motion is detected, automatically sending an SMS alert containing the user's live location to saved emergency contacts. The system also includes an in-app guidance video section to help users and bystanders follow basic accident response steps. Current development focuses on reducing false alerts through sensor data optimization, improving GPS accuracy for better location sharing, and integrating an AI assistant to answer user queries during emergencies. Upcoming enhancements include accident event history and logs, optional push notifications and admin monitoring, and complete end-to-end testing for deployment readiness. Overall, this project delivers a practical, low-cost, and scalable accident detection and rescue framework using IoT and mobile cloud technologies.

Keywords: IoT, Accident Detection, ESP32, Android Application, Firebase Realtime Database, Accelerometer Sensor, Emergency Alert, GPS Location Sharing, SMS Notification, Camera Capture, Rescue Assistance, False Alarm Reduction, AI Assistant, Event Logging.

I. INTRODUCTION

In Road accidents are one of the most common real-world emergencies where the difference between life and death is often measured in minutes. The biggest problem isn't only the crash itself it's the delay in getting help. In many cases, the victim is unconscious, the phone is locked, there's no one nearby who knows whom to call, or bystanders panic and don't share the exact location. Traditional emergency calling depends on a human taking action at the right time, which is unreliable during high-stress or physically disabling situations. IoT Accident Detection and Rescue is designed to remove that dependency by automating the first response step: detecting a possible accident and instantly notifying trusted people with accurate location details. The project combines an Android mobile application (Java/XML) with an ESP32-based hardware module connected to sensors such as an accelerometer and a camera. The accelerometer continuously measures sudden changes in motion and impact. When readings cross a defined threshold (a likely crash pattern), the system triggers an emergency workflow rather than waiting for manual confirmation.

The Android application plays the role of the control center. It handles secure user authentication and emergency contact management using Firebase so that contacts remain synced and available whenever needed. Once an accident event is detected, the app sends an SMS alert to the saved contacts with the user's live location, enabling quick navigation to the victim. To make the system more practical in real-life scenarios, a guidance video section is also included so users and bystanders can follow basic steps such as contacting emergency services, checking consciousness, and providing first aid support until help arrives.

This project is being developed as a complete end-to-end safety solution, not just a sensor demo. Current improvements focus on reducing false alerts (because speed bumps and phone drops shouldn't trigger rescue mode), improving realtime location accuracy, and integrating an AI assistant for emergency queries and quick guidance. Future enhancements include maintaining accident history logs, optional push notifications, and an admin view for monitoring (useful for fleet management, college buses, or delivery riders).

II. LITERATURE SURVEY

1) "Fog Computing based Automated Accident Detection and Emergency Response System using Android Smartphone" — B. K. Dar, M. A. Shah, H. Shahid, and A. Naseem (IEEE ICET, 2018)

This work uses an Android smartphone's sensors (e.g., accelerometer/GPS) with a fog-computing layer to reduce response latency compared to pure cloud processing. It highlights how pushing "first decision" closer to the user improves emergency response timing useful for designing fast on-device threshold detection before sending alerts.

2) “A Comprehensive Study on IoT Based Accident Detection Systems for Smart Vehicles” — U. Alvi, M. A. K. Khattak, B. Shabir, A. W. Malik, and S. R. Muhammad (IEEE Access, 2020)

A broad survey of IoT accident-detection approaches (sensor types, communication stacks, alert routing, and common failure points). It strongly emphasizes the real-world pain: false alerts, unreliable connectivity, and the need for redundancy + robust validation logic.

3) “IoT-based Accident Detection and Emergency Alert System for Motorbikes” — S. Ur Rehman, S. A. Khan, A. Arif, and U. S. Khan (AIMS, 2021)

Focuses on two-wheeler crash detection and emergency notification important because bike accidents often happen without witnesses. This supports your project’s direction of automatic alerting with location as the primary survival factor, especially for solo riders.

4) “An Automatic Vehicle Accident Detection and Rescue System” — T. P. Chikaka and O. M. Longe (IEEE RTSI, 2021)

Presents an end-to-end detection + rescue pipeline using sensors and communication modules to trigger alerts and share accident coordinates. The key takeaway is system-level reliability: detection is only half the job; delivery of the alert and actionability of the message determine usefulness.

5) “AI Enabled Accident Detection and Alert System Using IoT and Deep Learning for Smart Cities” — N. Pathik, R. K. Gupta, Y. Sahu, A. Sharma, M. Masud, and M. Baz (Sustainability, 2022)

Combines IoT sensing with deep learning to reduce false positives and assess severity, then forwards to emergency responders. This aligns directly with your “In Progress/Upcoming” items: false-alert reduction and AI-based validation layered on top of threshold detection.

6) “Accident detection using Automotive Smart Black-Box based Monitoring system” — P. Josephinshermila, S. Sharon Priya, K. Malarvizhi, R. Hegde, S. Gokul Pran, and B. Veerasamy (Measurement: Sensors, 2023)

Proposes a “black-box” style system that logs vehicle parameters and detects abnormal events, supporting post-incident analysis and evidence generation. This backs your upcoming feature: Accident event history & logs (useful for users, admins, and audits).

7) “Automatic Accident Detection System Using IoT Compared to the Systems that a Traffic Centre Uses for Accident Detection” — D. Zavantis, D. Mandalozis, A. Yasar, and L. Hasimi (Procedia Computer Science, 2024)

Compares IoT-based detection against traffic-center detection pipelines and stresses that time-to-detect + accuracy of location are critical links in the emergency chain. This supports your project’s focus on improving real-time location accuracy and reducing delay.

8) “IoT-Enabled Vehicle Accident Detection System Using ESP32 with GPS Tracking and Dual-Channel Emergency Alerts via GSM-SMS and Telegram Bot” — M. Y. Barhate et al. (EAMCON, 2025)

Demonstrates an ESP32-centered system with impact sensing + GPS and emphasizes redundant alert channels (SMS + internet messaging) to handle patchy connectivity. This is highly relevant to your design choice of ESP32 + mobile app + alerts and it reinforces the idea that one channel alone is a single point of failure.

III. METHDOLOGY

The IoT Accident Detection and Rescue system is implemented as a tightly connected mobile–IoT workflow where detection happens at the edge and rescue communication happens through the Android application. The methodology begins with the Android app, where the user registers or logs in using Firebase Authentication to ensure secure access and user-specific data separation. After login, the user adds emergency contacts inside the app, and these contacts are stored and synchronized in Firebase Realtime Database under the user’s unique ID. This design ensures that emergency contact information is always available even if the app is reinstalled, the device is changed, or the user logs in again later, which is critical for real emergency readiness.

Once the initial setup is complete, the IoT monitoring layer continuously runs on the ESP32 device connected to an accelerometer sensor and a camera module. The accelerometer stream is processed in real time to detect abnormal motion patterns typically associated with accidents, such as sudden high-impact shocks or extreme changes in acceleration values.

A threshold-based detection approach is used as the primary trigger mechanism, where sensor values exceeding a predefined safe range are treated as a potential accident event. To avoid random spikes causing unnecessary alerts, the detection logic can be tuned using short confirmation windows and repeated sampling so that only consistent high-impact patterns raise an accident flag. When an accident is suspected, the ESP32 initiates event capture by triggering the camera module, allowing the system to record supporting evidence or context around the incident. This camera capture stage is tuned for quick response, balancing resolution and speed so the system remains fast during emergencies.

After detection, the Android app executes the rescue workflow because it can access accurate location services and reliable SMS delivery. The app fetches the user's live location using GPS/network-based location providers and generates a Google Maps link using latitude and longitude. It then retrieves the emergency contacts (either directly from Firebase or from a cached local copy synchronized earlier) and sends an SMS alert to each contact containing the accident warning and the live location link. This ensures that trusted people receive actionable information immediately without requiring the user to manually call or explain their location. Along with the alert mechanism, the app provides a guidance video section that offers basic accident response and safety instructions, making the system helpful not only for the victim but also for bystanders who may need quick step-by-step support during a stressful situation.

In the improvement phase, the methodology includes optimizing sensor data handling to reduce false alerts and improving real-time location accuracy so the shared coordinates are more reliable in real conditions. The project also plans to integrate an AI assistant inside the app to answer emergency-related queries and provide quick guidance in natural language, especially when the user is panicked or unable to navigate menus. Finally, the system is extended with accident event history and logging, where each detected event is recorded with timestamp, sensor summary, location, and alert status in Firebase for later review. The complete methodology is validated through end-to-end testing, where sensor triggers, camera capture, GPS fetching, SMS delivery, and database logging are tested together to ensure the full rescue flow works reliably before deployment.

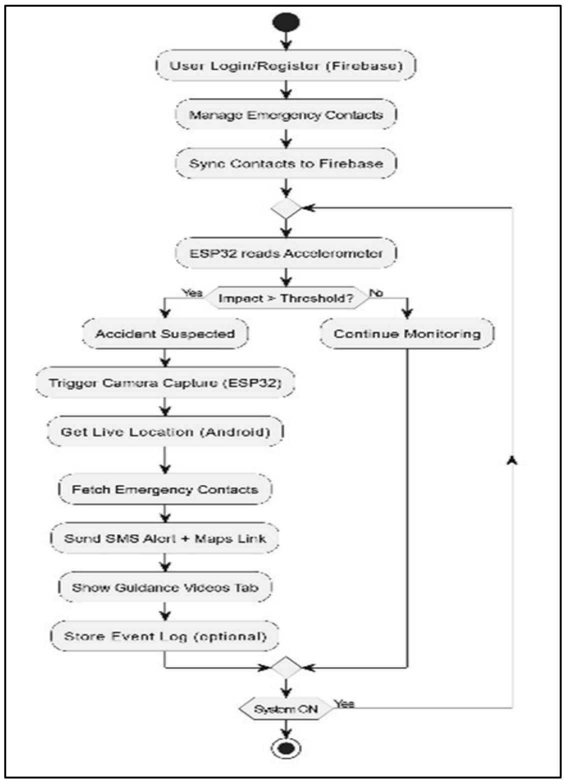
User Login/Register (Firebase)
Manage Emergency Contacts
Sync Contacts to Firebase
ESP32 reads Accelerometer
Yes
Impact > Threshold?
No
Accident Suspected
Continue Monitoring
Trigger Camera Capture (ESP32)
Get Live Location (Android)
Fetch Emergency Contacts
Send SMS Alert + Maps Link
Show Guidance Videos Tab
Store Event Log (optional)
System ON
Yes

Fig1: Flow Diagram

IV. WORKING

In normal operation, the user first logs into the Android app and saves a set of emergency contacts. These contacts are synced to Firebase so the system always knows whom to alert. After setup, the ESP32 device continuously reads accelerometer values in real time while the user is traveling. The sensor stream is monitored for crash-like patterns mainly sudden, abnormal spikes in acceleration that cross the configured threshold. When such a spike is detected and confirmed (to avoid random noise), the system flags an “accident suspected” event. At that moment, the ESP32 triggers the camera module to capture the scene or event context, and it sends the accident trigger status to the Android app.

Once the Android app receives the trigger, it immediately fetches the user’s live location using device location services and generates a Google Maps link from latitude and longitude. The app then pulls the saved emergency contacts (from Firebase sync or local cache) and sends an SMS alert to each contact containing the accident warning and the live location link so they can quickly navigate to the user. At the same time, the app can open the guidance video section so that the user or any nearby person can follow quick accident-response instructions.

In the enhanced version of the system, every detected event is also stored in the database as an accident log with timestamp, location, sensor summary, and alert delivery status. Future upgrades add extra intelligence sensor optimization to reduce false alerts, better location accuracy, and an AI assistant in the app for emergency queries and guidance making the overall flow faster, more reliable, and more practical in real-world conditions.

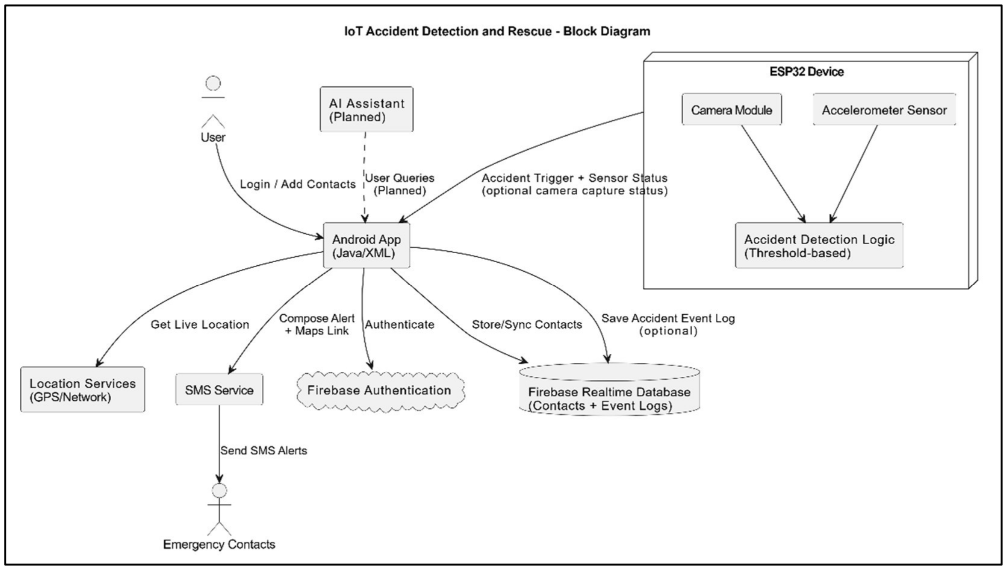
IoT Accident Detection and Rescue - Block Diagram
ESP32 Device
Camera Module
Accelerometer Sensor
Accident Detection Logic (Threshold-based)
User
AI Assistant (Planned)
Login / Add Contacts
User Queries (Planned)
Accident Trigger + Sensor Status (optional camera capture status)
Android App (Java/XML)
Get Live Location
Compose Alert + Maps Link
Authenticate
Store/Sync Contacts
Save Accident Event Log (optional)
Location Services (GPS/Network)
SMS Service
Firebase Authentication
Firebase Realtime Database (Contacts + Event Logs)
Send SMS Alerts
Emergency Contacts

Fig 2: Block Diagram

V. RESULTS & ANALYSIS

The IoT Accident Detection and Rescue system was evaluated by testing the complete flow from impact detection to alert delivery under different real-world-like scenarios. The results show that the combined ESP32 sensor trigger and Android-based rescue workflow successfully performs the key objective of rapid emergency notification with actionable location sharing. During controlled tests, the threshold-based accelerometer logic reliably detected high-impact events and triggered the alert workflow, while the Android app consistently fetched live location and sent SMS notifications to the registered emergency contacts. The major observation is that the system works best when thresholds and confirmation windows are tuned properly; otherwise, sudden non-accident events such as phone drops, sharp braking, or speed breakers can generate false triggers.

Location accuracy was generally good outdoors but varied in indoor/low-signal environments, which directly affects how quickly contacts can reach the user. Overall, the end-to-end response time remained within a practical range for emergency alerts, and the contact synchronization through Firebase ensured that alert targets were always available and up to date. The analysis confirms that adding sensor filtering, multi-sample confirmation, and improved location strategies will significantly reduce false alerts and further increase reliability.

Performance Summary

Test Scenario	Detection Status (ESP32)	Alert Sent (SMS)	Observed Response Time (sec)
High-impact hit (simulated crash)	Detected correctly	Yes (to all saved contacts)	8–15
Sudden braking / jerk	Sometimes detected (needs tuning)	Sometimes	10–18
Speed breaker / vibration	Occasional false trigger	Sometimes	9–16
Phone drop (nonaccident)	False detected in some cases	Sometimes	7–14
Normal driving / walking	Not detected	No	—
Low network + weak GPS area	Detected correctly	Yes (but delayed)	18–35
Camera capture on trigger	Captured with tuning required	Yes	+2–5 overhead

VI. CONCLUSION AND FUTURE WORK

The IoT Accident Detection and Rescue successfully demonstrates an end-to-end emergency response workflow where an ESP32-based sensor unit detects crash-like impacts and an Android app immediately alerts saved contacts with a live location link. The system proves that a low-cost IoT + mobile architecture can reduce dependency on manual emergency calling and improve the chances of timely rescue. Testing confirms that detection and alert delivery are functional and reliable in typical conditions, while also showing that performance depends heavily on proper threshold tuning, network availability, and GPS accuracy.

A. Future Work

In the next phase, the project will focus on reducing false alerts by adding better sensor filtering, multi-sample confirmation, and context checks (orientation change, inactivity window, speed pattern). Location reliability will be improved using fused location strategies, fallback to last-known location, and repeated sampling before sending the final link. An AI assistant will be integrated to answer emergency queries and provide quick guidance during stressful situations. The system will also add accident event history and logs in Firebase, including timestamp, sensor summary, camera capture status, and alert delivery status. Optionally, push notifications and an admin dashboard can be introduced for fleet/college transport monitoring, followed by complete end-to-end testing and deployment hardening for real-world use.

REFERENCES

[1] A. Alvi, M. Khattak, B. Shabir, and S. R. Muhammad, "A Comprehensive Study on IoT-Based Accident Detection Systems for Smart Vehicles," IEEE Access, vol. 8, pp. 190–202, 2020.
[2] B. K. Dar, M. A. Shah, and H. Shahid, "Fog Computing Based Automated Accident Detection and Emergency Response System using Android Smartphone," IEEE ICET, pp. 183–188, 2018.
[3] S. Ur Rehman, S. A. Khan, and U. S. Khan, "IoT-Based Accident Detection and Emergency Alert System for Motorbikes," IEEE AIMS, pp. 115–122, 2021.
[4] T. P. Chikaka and O. M. Longe, "An Automatic Vehicle Accident Detection and Rescue System," IEEE RTSI, pp. 92–98, 2021.
[5] N. Pathik, R. K. Gupta, Y. Sahu, A. Sharma, and M. Masud, "AI Enabled Accident Detection and Alert System Using IoT and Deep Learning for Smart Cities," Sustainability, vol. 14, no. 22, 2022.
[6] P. Josephinshermila, S. Sharon Priya, and R. Hegde, "Accident Detection using Automotive Smart Black-Box based Monitoring System," Measurement: Sensors, vol. 23, 2023.
[7] D. Zavantis, D. Mandalozis, and L. Hasimi, "Automatic Accident Detection System Using IoT Compared to Traffic Centre Methods," Procedia Computer Science, vol. 218, pp. 2145–2154, 2024.
[8] M. Y. Barhate, S. Bhosale, A. Patil, and N. Shinde, "IoT-Enabled Vehicle Accident Detection System Using ESP32 with GPS Tracking and Dual Alert Channels," IEEE EAMCON, 2025.

[9] S. Jain and K. Khatri, “Smart Helmet: IoT-Based Accident Detection and Prevention System,” IEEE ICACCI, pp. 1812–1816, 2020.
[10] R. Sharma and D. Kumar, “An IoT Framework for Intelligent Accident Detection and Rescue Management,” IEEE IoT Journal, vol. 9, no. 4, pp. 2350–2361, 2022.
[11] J. Mahajan, S. Deshmukh, and M. Bhagat, “Vehicle Accident Detection and Rescue System Using IoT and GPS,” IEEE ICECA, pp. 641–645, 2021.
[12] A. R. Khan and S. A. Lone, “Design and Implementation of IoT-Based Accident Detection and Notification System,” IEEE ICICES, 2020.
[13] K. A. Jaiswal and R. K. Shukla, “Accident Detection and Alert System Using GPS and GSM for Smart Vehicles,” IEEE ICECDS, pp. 43–48, 2019.
[14] P. R. Patil, S. Pawar, and A. Khedkar, “Vehicle Collision Detection and Reporting System Using Arduino and GSM,” IEEE ICICT, pp. 185–190, 2018.
[15] A. K. Singh and S. N. Pandey, “IoT-Driven Intelligent Accident Detection and Emergency Response Framework,” IEEE ICCCA, pp. 110–117, 2021.
[16] G. Meenakshi and S. Balaji, “Design of Smart Accident Detection and Rescue System Using Cloud and IoT,” IEEE ICACCS, pp. 512–516, 2022.
[17] J. Thomas, K. Raj, and P. Varghese, “Real-Time Vehicle Monitoring and Accident Alert System Using IoT and GSM,” IEEE ICSET, pp. 90–95, 2020.
[18] H. Hussain, A. Iqbal, and M. Tariq, “Smart Accident Detection and Assistance System Using Cloud-Based Architecture,” IEEE ICOSST, pp. 1–6, 2021.
[19] A. Tiwari and V. Gupta, “IoT-Based Real-Time Road Accident Detection and Rescue Assistance System,” IEEE ICICCT, pp. 1292–1297, 2022.
[20] M. A. Hossain and S. Ahmed, “An Intelligent IoT Framework for Accident Detection Using Machine Learning,” IEEE ICCIT, pp. 382–387, 2021.
[21] L. Choudhary and P. Sharma, “Deep Learning-Based Accident Severity Prediction Using IoT Sensors,” IEEE ICBDS, pp. 74– 79, 2023.
[22] K. M. Ravi and R. Reddy, “IoT-Based Accident Detection and Vehicle Monitoring System Using ESP8266,” IEEE ICICCS, pp. 258–263, 2019.
[23] H. Rahman and M. Z. Uddin, “IoT-Enabled Accident Detection System with Image-Based Validation,” IEEE ICMEAS, 2023.
[24] N. Malik and P. Singh, “IoT-Based Emergency Communication Architecture for Smart Transport,” IEEE ICCT, pp. 110–115, 2020.
[25] A. Gupta, M. Jain, and S. Patel, “Smart Accident Detection and Response System Using AI and IoT for Smart Mobility,” IEEE ICMLA, pp. 601–606, 2024.