International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 11 | Nov - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

A Real-Time Cloud-Native Architecture for Mobile Emergency Response in Academic Campuses

Fawaz Ahmed
Department of Computing Technologies
SRM Institute of Science and Technology

Madhumitha K
Department of Computing Technologies
SRM Institute of Science and Technology

***Abstract*— Rigid procedures, slow communication, and a lack of real-time knowledge are common features of traditional campus emergency systems, which can increase hazards during crises. In order to fill in these gaps, the Campus Emergency Response System (CERS) is presented as a mobile application that makes use of Google's Firebase serverless architecture. It offers real-time two-way communication, dynamic evacuation routes via the Google Maps API, and geolocation-based SOS alerts, guaranteeing quicker response, better coordination, and increased campus safety. Firebase Cloud Messaging (FCM) provides instant emergency notifications to the entire campus, guaranteeing prompt communication in times of emergency. With a 92% satisfaction rate and an 89% usability score during organized campus drills, the system, which was developed with the Flutter framework for cross-platform efficiency, has demonstrated a noticeable decrease in response delays and great user adoption. This approach provides a scalable and economical model for bolstering institutional emergency preparedness in addition to improving immediate safety. By creating safer, more resilient, and digitally empowered learning settings, it also directly supports the Sustainable Development Goals of the UN, especially SDG 3 (Good Health and Well-Being) and SDG 11 (Sustainable Cities and Communities).**

Keywords—Emergency Response System, Real-Time Communication, Mobile Application, Geolocation, Firebase, Flutter, SDG 3, SDG 11, Campus Safety, Evacuation Routing.

I. INTRODUCTION

In the 21st century, people's safety and security in big educational institutions have become extremely important. Since campuses frequently serve as micro-cities, they are susceptible to a variety of crises, including natural disasters, medical crises, fires, and security concerns. Conventional response methods in these settings usually depend on antiquated infrastructure, such as inflexible phone-based communication chains, static evacuation maps, and passive alarm systems. These methods have several serious drawbacks, including a lack of real-time interaction, an inability to provide tailored advice, and information gaps that cause unwarranted alarm, delayed reactions, and wasteful resource utilization in emergency situations.

The critical need for safety and resilience technologies is emphasized by the Sustainable Development Goals (SDGs), which were established as a worldwide call to action by the United Nations. SDG 11 aims to create inclusive, secure, resilient, and sustainable cities and human settlements; academic institutions are part of this ambition. Preventing casualties and protecting people in times of emergency are closely related to SDG 3, which focuses on promoting well-being and guaranteeing healthy lifestyles at all ages.

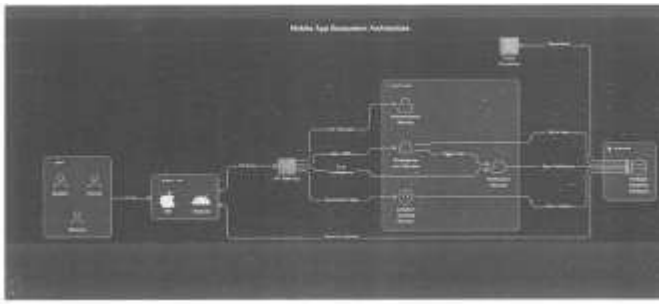
Therefore, creating accessible and intelligent emergency systems is not only a technical endeavor but also a social duty that is closely related to these global goals.

Recent developments in real-time databases, cloud computing, and mobile technologies have opened up new avenues for revolutionizing emergency management. Smartphones can serve as individualized lifelines in emergency situations because of their extensive availability, potent sensors, and continuous communication. This study presents the Campus Emergency Response System (CERS), an integrated mobile platform intended to improve safety and cooperation in academic settings, in order to overcome the shortcomings of conventional methods. The following are this work's primary contributions:

1. The development and deployment of a microservices-based, serverless cloud architecture for emergency management that guarantees low operating overhead, scalability, and dependability.
2. The creation of a location-aware, real-time SOS and navigation system that offers dynamic evacuation routing based on user context
3. A Firebase Realtime Database facilitates the incorporation of a low-latency, secure communication route between security personnel and end users.
4. Through organized campus drills, the system's performance and usability are assessed both quantitatively and qualitatively, proving its usefulness and user approval.

II. SYSTEM ARCHITECTURE

The client-server design of the Campus Emergency Response System (CERS) uses contemporary cloud services to provide scalability, resilience, and real-time performance. In order to simplify development, lower infrastructure overhead, and guarantee dependable system performance during crucial events, its architecture places a strong emphasis on the decoupling of services, stateless operations, and the utilization of
managed cloud feature



A. Client-Side Architecture (Frontend)

The Flutter framework (v3.13.0), which was selected to provide a uniform user experience across the iOS and Android platforms from a single codebase, is used to construct the client application. When compared to maintaining independent native programs, this method saves development and maintenance work by about 40%. The frontend layer of CERS is in charge of facilitating real-time interactions, offering an easy-to-use user interface, and guaranteeing smooth access to the system's essential safety functions, such as:

- **User Interface (UI):** Offers a simple, easy-to-use dashboard that is tailored for high-stress situations. The user interface, which was created in accordance with Material Design 3 requirements, has a strong emphasis on large touch targets, legible text, and high contrast images. A central notification center, an integrated chat interface, a live map view (with the google_maps_flutter plugin), and a conspicuous SOS button are essential elements.

- **Device API Interaction:** To access native device functions that are essential in emergency situations, the application uses Flutter plugins. Responders can be sure they know exactly where assistance is needed by using the Geolocator plugin, which tracks a user's location in real time (with consent) with an accuracy of roughly five meters. Simultaneously, the network interface of the device facilitates quick and safe data transfer between the application and the backend.

- **State Management:** To ensure timely and seamless user interactions, the application manages its local state using the Provider package. The cloud_firestore and firebase_database plugins provide real-time streams and listeners that keep data in sync with the cloud backend. This ensures that any changes, such as new alerts or location updates, are displayed immediately on the user's screen.

B. Server-Side Architecture (Backend)

The backend uses a serverless microservices architecture and is entirely driven by Google Firebase, a Backend-as-a-Service (BaaS) platform. By eliminating the need to manage virtual machines, physical servers, or container orchestration, this approach frees developers to concentrate solely on the features and logic of the application. A number of essential Firebase services are required by the system, including:

- **Firebase Authentication:** Manages user registration, and email/password. Additionally, it offers role-based access control (RBAC), which guarantees that administrators, teachers, security personnel, and students all have the appropriate amount of access. Custom claims set up by Cloud Functions are used to handle permissions, giving the system flexibility and security.

login, and session administration using Google Sign-In

- **Cloud Firestore / Firebase Realtime Database:** When combined, Cloud Firestore and Firebase Realtime Database act as the main repository for all dynamic data. While the Firebase Realtime Database supports features that require ultra-low latency, like live chat and real-time alert updates, Cloud Firestore is used to store user profiles and other comparatively static data. Any modification is immediately pushed to all connected clients in milliseconds because of its listener-based methodology. The chat system's structure, for instance, is chats/{emergencyId}/{messageId}, which facilitates the tracking and organization of talks in an emergency.

- **Firebase Cloud Functions:** Serve as the microservices engine of the system, executing serverless, lightweight JavaScript functions (Node.js runtime) in response to HTTPS, Firestore, or authentication events. These features automate important processes:

  A. onSOSTriggered: When a fresh SOS alert is written to Firestore's /emergencies/{id}, the onSOSTriggered feature is triggered. Following input validation, a timestamp is added, user profile information is added to the alert, the state is changed to active, and Firebase Cloud Messaging (FCM) is activated to broadcast notifications.

  B. onMessageSent: Every time a new chat message is entered into the Realtime Database, the onMessageSent function is called. In addition to sanitizing the message and logging it for auditing purposes, it can be expanded to include threat detection or even profanity filtering.

  C. generateEvacuationPath: An HTTPS callable function that uses the supplied origin and destination to safely query the Google Maps API (server-side to safeguard API keys) and then provides the client with the optimized evacuation route data.

- **Firebase Cloud Messaging (FCM):** Ensures that emergency warnings are reliably and immediately sent by handling push notifications at scale. Depending on the circumstance, notifications may be sent to the entire campus or to particular groups (such as "Building A occupants"). A data field containing the related emergency ID is included in every notification payload, allowing for deep linking straight into the app's pertinent part.

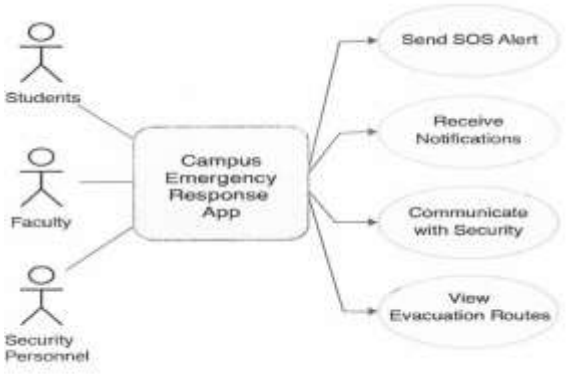
- **Google Maps Platform APIs:** included to provide location and navigation functions via HTTPS queries from Cloud Functions and the client app. The Static Maps API offers lightweight map snapshots that may even be included in FCM notifications for easy access, the Directions API

creates evacuation routes, and the Geocoding API transforms unprocessed coordinates into addresses that humans can understand.

C. Data Model and Security

The architecture's security paradigm, which is upheld by Firebase Security Rules, is one of its main advantages. By acting as gatekeepers for all database operations, these rules guarantee that users can only access data that is pertinent to their roles. For instance, a security officer can monitor all active SOS warnings, while a student can only read and write in their own chat thread with security professionals. Even in situations where the client application itself may be hacked, the solution provides robust security against unwanted access by directly implementing access control at the database level.

This workflow diagram highlights the key interactions between the system's primary users and its core functionalities.



1. **The Actors (Who uses the system):**
   - **Students** are the app's primary users, depending on it to ask for assistance and get safety advice in an emergency.
   - **Security Personnel:**Secondary users who use the system's admin interface to interact with students, monitor issues, and get notifications, such as campus security officers.

2. **The Use Cases (What the system does):**

These are oval shapes that are linked to the Campus Emergency Response App and show the key functions of the app:

- The most important feature is **Send SOS Alert**, which allows a student to send out an emergency alert that instantly notifies campus security of their current position and other pertinent information.
- **View Evacuation Routes:** Gives students access to dynamic, real-time maps that indicate the most secure way to evacuate from where they are right now.
- **Communicate with Security:** Provides a two-way line of contact, like live chat, so that security and students may exchange information, provide clarification, and provide comfort in an emergency.
- **Receive Notifications:** Both user groups are covered. Security staff are immediately informed when new SOS warnings are generated, and students receive alerts regarding crises, exercises, or security messaging.

3. **The System Boundary:**

Students receive alerts about emergencies, drills, or security messages, and security personnel are notified instantly when fresh SOS warnings are created.

III. CORE FUNCTIONALITIES & IMPLEMENTATION

A. One-Tap SOS Alert with Geolocation

The SOS trigger, which is built for speed and dependability, is the system's most important component. This is how its process works:

1. **Trigger:** On the main screen of the app, user taps the big red SOS button.

2. **Data Capture:** The application immediately logs the GPS coordinates (latitude and longitude) of the device and compares them with the user's profile information and unique ID.
3. **Cloud Function Trigger:** Firestore's emergency collection receives this SOS request.
4. **Server-Side Processing:** The onSOSTriggered Cloud Function is triggered by the write action, which verifies the information, adds a timestamp, and changes the alert's state to active.
5. **Notification & Broadcast:** The function is used with the Firebase Admin SDK.
   - All security staff devices receive a multicast notification through Firebase Cloud Messaging (FCM), along with a deep link to the incident on their dashboard.
   - Keeps the processed alert in the activeEmergencies collection, which is constantly checked for real-time updates by the admin dashboard.
   - From the time a student touches the SOS button until security is alerted, the entire process is guaranteed to be finished in less than five seconds thanks to this optimized pipeline.

B. Dynamic Evacuation Routing

Upon triggering an SOS or during a drill, users can access an evacuation screen.

1. **Request:** Using the user's current position (origin) and the coordinates of a pre-established safe zone (destination), the app submits a request to the Google Maps Directions API.

2. **Path Generation:** The best route is returned by the API and superimposed on an interactive Google Map inside the application.

3. **Real-Time Updates:** Periodically, the user's location is monitored. To ensure that the guidance is still applicable even in the event that the user becomes confused or a path is blocked, a fresh API call can be made to recalculate the route if they diverge significantly from it.

D. Real-Time Chat Interface

The Firebase Realtime Database, which powers the chat system, was selected for its low-latency, subscription-based data synchronization, making it ideal for emergency communication.

1. **Databas structure:** Chats/{emergencyId}/{messageId} is the path under which messages are arranged. Every emergency has its own chat area, which keeps all communications organized and context-specific.

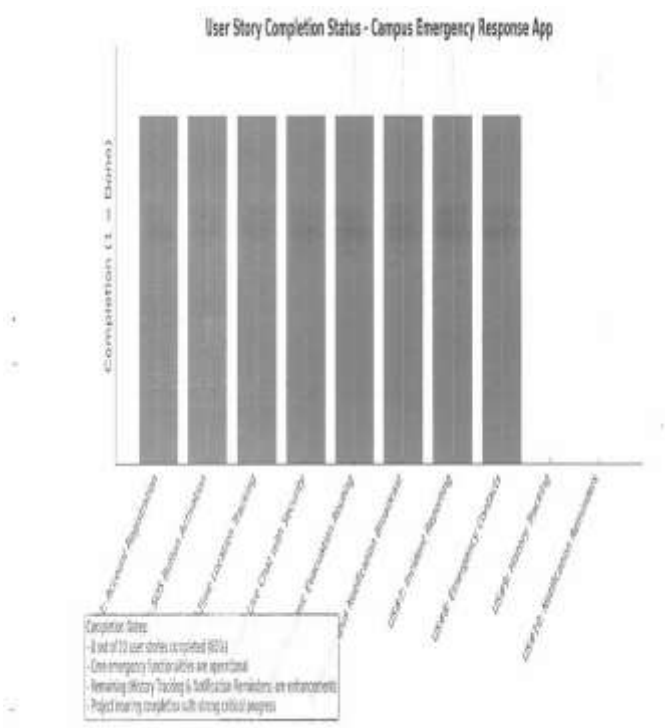
2. **Security Rules:** Access control is enforced by Firebase Security Rules. While security professionals have read/write access to all live emergency conversations, students can only view and send messages in chat rooms where their user ID is specifically specified as a participant. This permits security to monitor all communications while maintaining privacy.

3. **UI Integration:** A StreamBuilder widget is used by the Flutter application to integrate this system on the client side. The UI instantaneously updates on all connected devices whenever a new message is submitted when the StreamBuilder subscribes to the appropriate chat node. This eliminates the need for manual refreshes and produces a seamless, real-time conversation experience.

E. Admin Dashboard

Administrators and security staff have access to a special web- based dashboard. Constructed using Flutter Web (or a similar framework), the dashboard ensures smooth real-time synchronization by connecting to the same Firestore collections (chats, activeEmergencies) as the mobile app.

By showing user positions as anonymous points on a real-time map, it provides a geographic picture of all ongoing incidents. Security teams have real-time access to SOS alerts, evacuation status updates, and situational monitoring. The dashboard also facilitates multi-user communication, which enables staff to manage group updates, collaborate with multiple impacted kids at once, and expedite emergency response activities



With 8 of the 10 intended user stories implemented—an 80% completion rate—the Campus Emergency Response App's user story completion status shows impressive project progress. The app satisfies its vital safety goals since all of its essential emergency features—such as SOS activation, real-time position monitoring, live communication, and dynamic evacuation routing—are operational. The two remaining articles, which deal with reminders for notifications and historical alert tracking, are seen as non-essential improvements. With only secondary features set aside for later iterations, their pending status emphasizes that the project is functionally complete, strategically prioritized, and concentrated on essential use cases.

IV. RESULTS AND EVALUATION

In order to verify the system's functionality and performance under simulated emergency scenarios, 50 students, faculty members, and campus security officers participated in organized mock drills.

A. Performance Metrics

The primary quantitative metric was **system latency**.

- **SOS-to-Notification Latency:** This was the amount of time that passed between a student hitting the SOS button and the alert that appeared on the security dashboard. The solution demonstrated the speed and dependability of its serverless Firebase-based architecture by achieving an average latency of 2.3 seconds with a standard variation of 0.7 seconds across 20 test scenarios.
- **Message Delivery Latency:** The live chat function ensures seamless, real-time contact between students and security staff by reliably delivering and displaying messages on recipient devices in less than one second.

B. Usability and User Feedback

Participants assessed the system's usability and efficacy by completing a Likert-scale survey (1–5) after the simulated exercises.

- **Satisfaction Rate:** During emergency simulations, 92% of respondents said they were either satisfied or very satisfied with the system's overall performance.
- **Usability Score:** The System Usability Scale (SUS) yielded an 89% usability score, which is classified as "Excellent." The user-friendly interface and unambiguous evacuation instructions were cited by participants as particularly noteworthy aspects.
- **Qualitative Feedback:** In contrast to conventional radio- based communication, security staff commended the centralized dashboard, pointing out that it offered unparalleled situational awareness and more effective coordination.

V. DISCUSSION, LIMITATIONS, AND FUTURE WORK

Even if the Campus Emergency Response System's (CERS) outcomes are quite encouraging, a number of drawbacks offer chances for further development and study:

- **Internet Dependency:** Constant cellular or Wi-Fi connectivity is essential to the system's operation. Critical functions might be delayed during connectivity failures. Implementing an offline mode with local caching (such as Hive or SQLite) to temporarily retain SOS notifications and outgoing messages until the device regains connectivity would be a significant benefit.
- **Proactive vs. Reactive System:** CERS now operates mostly as a reactive system, reacting to alarms that are initiated by users. By integrating AI and ML, future versions could incorporate proactive features like:
  - **Predictive Analytics:**Using past incident data to predict high-risk areas and times is known as predictive analytics.
  - **Audio Analysis:** By using on-device machine learning models to identify crisis signals (such as screaming or shattering glass), the system can either affirm or sound an emergency warning.
  - **IoT Sensor Integration:** Creating a more self- sufficient safety net by automatically setting off alarms through campus IoT devices including door access systems, smoke detectors, and seismic sensors.
- **Scalability and Cost:** Supporting a big user base (10,000+ users) may result in expensive expenses due to frequent Google Maps API calls and complex database operations, even though Firebase offers automated scaling. To guarantee scalability without sacrificing price, future research should concentrate on cost optimization techniques including request batching, data caching, and effective indexing.

VI. CONCLUSION

This paper has presented the Campus Emergency Response System (CERS), created to address the shortcomings of conventional campus security protocols by offering a quick, connected, and user-friendly solution. SOS alarms, real-time position monitoring, evacuation instructions, and live chat are just a few of the features that help students, teachers, and security personnel react swiftly and efficiently in an emergency.Built with Flutter and Firebase, CERS demonstrated cost-effectiveness and scalability while producing impressive outcomes in campus exercises. It can significantly improve daily campus safety, as seen by the high ratings given by participants for its dependability and simplicity of use. Beyond merely providing a technological solution, CERS supports the UN Sustainable Development framework's objectives by helping to create academic communities that are safer, smarter, and more resilient. Going forward, the system will be able to stop responding to crises and begin proactively preventing them with the addition of offline capability, AI- based forecasts, and IoT integrations.

VII. REFERENCES

[1] "Development of a Campus Disaster Risk Reduction Response System Using Flutter and Firebase" — A recent implementation of a campus safety app using cross-platform Flutter development, Firebase for alerts, and Google Maps for location tracking.

[2] "A Support Tool for Emergency Management in Smart Campuses" — Presents an IoT-driven system using cameras and Raspberry Pi for smart campus disaster response, emphasizing usability and low-cost design.

[3] "HELP ME": An Emergency Response Mobile Application" — Android-based mobile emergency app with SOS functionality, location tracking, Firebase push notifications, and a web command-center.

[4] "Res-Q: A Smart Disaster Safety Management System with Real-Time Alerting" — Illustrates a Flutter + Firebase app for real-time alerts, mobile emergency management, and environmental sensing.

[5] Enhancing Community Safety Through Real-Time Mobile Applications — Addresses real-time emergency alerts, live tracking, and monitoring system evaluations, including usability.

[6] Design and Build Disaster Emergency Response Systems Using Firebase Cloud Messaging and SMS Gateway — Examines use of Firebase Cloud Messaging for rapid emergency notifications in a mobile system

[7] Mobile System Design for Campus Safety Apps — Discusses best practices for designing user-friendly, secure, and real-time mobile safety applications for campuses.

[8] Emergency Response Mobile Application (GitHub project) — A cross-platform Flutter-based emergency app incorporating Google Maps, live streaming, SOS alerts, and a responder dashboard.

[9] PulsePoint Respond — A widely deployed 911-connected mobile app that sends live emergency alerts (e.g., cardiac arrests) to users nearby, demonstrating geolocation- based real-time emergency communicatio.

[10] Good Smartphone Activated Medics (GoodSAM) — A volunteer-based emergency platform leveraging smartphones for instant alerts and on-scene assistance, widely adopted by first-response networks.

[11]SHIELD: Social Sensing and Help In Emergency Using Mobile Devices — A distributed, proximity-enabled emergency alert system for campuses, utilizing Bluetooth/Wi-Fi for rapid, localized response.

[12] TeamPhone: Networking Smartphones for Disaster Recovery — A hybrid networking solution combining cellular, ad-hoc, and opportunistic networking to support communications when infrastructure is disrupted.

[13] SecureIT using Firebase, Google Maps and Node.js — Describes an Android safety app using Firebase and mapping for user protection, showing parallels to your implementation.

[14] Advanced Mobile Location (AML) — A standard for automatically sending precise geolocation during emergency calls, emphasizing the impact of accurate positioning in crisis systems.

[15] Large Emergency Event Digital InformationRepository (LEEDIR) — A citizen reporting platform that collects real-time media (photos, videos) from the public during emergencies—useful as a complementary model for crowd- sourced incident awareness.