Design and Development of a Mobile App for Public Security and Emergency Alerts in Albania

Novruz BILLA[1]

Teuta XHINDI[2]

Abstract

Albania currently lacks a centralized mechanism to quickly disseminate emergency alerts and public safety information to its citizens. This paper presents the design and development of a mobile application aimed at managing emergency alerts and strengthening public safety response in Albania. The proposed system leverages real-time push notifications to inform citizens of crises or hazards as they unfold. This study followed a Design Science approach to gather requirements, architect a three-tier system, and implement a prototype using modern web and mobile technologies. The application consists of a React Native mobile client (for both iOS and Android) and a Node.js/Express backend with a MongoDB database. Key features include secure user authentication via JSON Web Tokens (JWT), role-based access control for institutional users, and an initial implementation of intelligent alert classification based on incident urgency. Preliminary testing with end-users and domain experts indicates that the system delivers stable performance, a user-friendly interface, and accurate, timely delivery of alerts. While these early evaluations (including a mean System Usability Scale score of 84/100) are promising, they are not yet conclusive. The

[1] European University of Tirana, Faculty of Engineering, Informatics and Architecture, Department of Informatics and Technology, Tirana, Albania, nbilla@uet.edu.al

[2] European University of Tirana, Faculty of Engineering, Informatics and Architecture, Department of Informatics and Technology, Tirana, Albania, teuta.xhindi@uet.edu.al

study highlights the role of modern information technology in improving institutional emergency response and provides a foundation for further integration of the system into national public safety infrastructure. The results and insights from this project serve as an important step toward a full-scale deployment of a nationwide emergency notification system in Albania.

Keywords: *Public safety, Emergency alerts, Mobile application, Push notifications, Design Science Research, Albania.*

Introduction

Albania is exposed to a range of natural and man-made hazards, yet it currently lacks a unified platform for disseminating official emergency information to the public. Recent disasters have highlighted this gap. For example, during the November 2019 earthquake (magnitude 6.4) that struck Albania, over 50 people lost their lives and hundreds were injured. In the critical hours immediately after the earthquake, official information channels were largely absent, leading to public panic and confusion. Similarly, major floods in 2022 isolated thousands in the Shkodër and Lezhë regions, underscoring shortcomings in timely evacuation warnings. Beyond natural disasters, public safety incidents such as criminal attacks or abductions also demand rapid public alerting. Although the State Police introduced the "Digital Commissariat" app for citizens to report incidents, there remains no comprehensive system for broadcasting emergency alerts to the populace and enabling two-way citizen reporting in one unified platform.

Globally, governments are increasingly leveraging mobile technology for emergency communication. Traditional siren and broadcast systems are being augmented or replaced by direct alerts to mobile phones. Notably, the European Union's Electronic Communications Code (EECC) directive in 2018 mandated that all member states implement a cell-broadcast based public warning system by 2022. Many countries including France, Germany, Italy, and the UK have since deployed nationwide mobile alert systems, either via cell broadcast or smartphone applications. These systems can reach millions of people within seconds, as evidenced by the UK's recent nationwide alert tests (BBC News, 2022) and similar initiatives elsewhere. The demand for timely information during crises is clear: for instance, usage of a global earthquake alert app in Albania surged from under 3,000 to over 146,000 users in the week following the 2019 quake, as citizens scrambled for reliable real-time updates (EMSC, 2019). The recurring communication failures observed during these events demonstrate a systemic need for a centralized and reliable emergency-alerting mechanism in Albania. While various institutions use fragmented channels, websites, social

media posts, press releases, or independent applications, none of these operate as a unified, real-time platform capable of reaching the population instantly and enabling structured two-way interaction. The absence of such a system reduces institutional responsiveness, increases public uncertainty, and limits the effectiveness of disaster-management operations. **This study therefore focuses on evaluating whether the development and implementation of a unified mobile emergency-alert and citizen-reporting platform can effectively address these gaps by improving communication accuracy and reducing response times.**

Research Questions

RQ1. What are the specific needs and gaps in emergency communication and alerting in Albania?
RQ2. How effective is the prototype in test scenarios (SUS, task success/time, and communication quality)?

Hypothesis: Implementing a unified mobile application for emergency alerts and citizen reporting will significantly improve emergency-management effectiveness in Albania by shortening institutional and citizen response times and increasing the accuracy and trustworthiness of public communication.

In this context, this study aims to develop a modern emergency notification mobile application tailored to Albania's needs. The goal is to provide a unified, real-time communication platform whereby authorities can instantly send public safety alerts (evacuation orders, hazard warnings) to citizens' smartphones, and citizens can report incidents (fires, accidents, etc.) directly to authorities. This paper presents the system architecture, implementation, and preliminary evaluation of the proposed solution. Its focus is on the technical design of the system a three-tier architecture encompassing a mobile frontend, a cloud-based backend, and a database along with security and scalability considerations. It also describes initial usability testing results and feedback from emergency management experts, which informed our discussion of the system's potential impact and areas for future improvement.

Literature review

Early warning and public alert systems have evolved significantly in recent decades. Traditional emergency alert systems (EAS) relied on sirens and broadcast media interruptions to reach the public. In the United States, for example, the legacy EAS breaks into radio and TV programming for urgent messages (FCC,

2019). With the ubiquity of mobile phones, attention has shifted toward cellular-based alerting. Modern systems like the U.S. Wireless Emergency Alerts (WEA) under FEMA's Integrated Public Alert & Warning System (IPAWS) can push geo-targeted text alerts to every compatible mobile phone in an affected area, with no need for any special app . These alerts appear automatically on devices and have proved capable of reaching broad populations almost instantly. The European Union similarly adopted a "Reverse-112" cell broadcast approach, mandating that all member states implement mobile public warning systems by 2022. Countries such as France, Germany, Italy, and Sweden have since deployed nationwide cell-broadcast systems (FR-Alert in France) to comply with this mandate.

Alongside broadcast-based solutions, many jurisdictions have experimented with mobile applications for emergency alerts. Such apps can offer richer interactivity (two-way reporting, multimedia content) but face challenges in achieving widespread adoption. A notable example is France's SAIP alert app, launched in 2016 and intended to notify citizens of terror attacks. SAIP suffered from technical failures and low usage, covering only about 1% of the population, and was ultimately discontinued in 2018. France pivoted to the FR-Alert cell broadcast system thereafter. In contrast, Germany has found some success with apps like NINA and KATWARN, which deliver alerts to users who opt in, complementing Germany's newer cell broadcast system (Bundesnetzagentur, 2022). The United Kingdom launched its own nationwide mobile alert system in 2023 using cell broadcast, conducting a full population test to ensure reach (BBC News, 2022). These experiences suggest that while dedicated apps can provide advanced features, integrating with device-native alert channels (like SMS Cell Broadcast) is crucial for maximum reach and reliability. Purely app-based systems risk leaving many users uninformed if the app is not installed or promptly maintained.

Another line of related research focuses on leveraging artificial intelligence (AI) to enhance emergency communications. AI and machine learning techniques have been explored for tasks such as automatic detection of incidents (earthquake early detection or social media monitoring) and intelligent prioritization of alerts. These approaches could enable future systems to filter false reports and highlight the most critical information. However, the use of AI in public safety also raises important ethical and transparency considerations (Smith *et al.*, 2023). Lastly, prior studies underscore the importance of usability in emergency alert tools. Users must be able to quickly understand and trust alerts under stressful conditions. Standardized usability metrics like the System Usability Scale are often applied to evaluate emergency apps (Brooke, 2013; Bangor *et al.*, 2009). Our work builds on these insights from the literature, aiming to combine a robust technical architecture with user-centric design and lessons learned from global best practices and pitfalls in emergency alert systems.

Methodology

The methodology combines mixed methods with a Design Science Research (DSR) approach, integrating theoretical/secondary research, iterative prototyping, and formative usability evaluation with end users and institutional experts (Hevner et al., 2004; Johnson & Onwuegbuzie, 2004; ISO 9241-210, 2019).

Research Approach and Design

A pragmatic paradigm with mixed methods was adopted to channel quantitative and qualitative evidence toward engineering decision-making (Johnson & Onwuegbuzie, 2004). The approach is grounded in DSR: building the mobile application and evaluating it as a research contribution (Hevner et al., 2004). In analogy with human-centered design, ISO 9241-210 (2019) principles were followed.

The research proceeded in five phases, each building on the previous:

- Phase 1 - Requirements Analysis: Conduct secondary research and consult experts to identify user needs, system constraints, and success criteria for an Albanian emergency alert system.
- Phase 2 - System Design: Develop the system's overall architecture, data schema, security model, and interface design based on the requirements.
- Phase 3 - Implementation: Iteratively build the prototype (both frontend mobile app and backend server), refining features through multiple development cycles.
- Phase 4 - Expert Evaluation: Use semi-structured interviews and walkthroughs with institutional stakeholders to assess the prototype's relevance, identify gaps, and gather suggestions, particularly regarding integration with existing systems.
- Phase 5 - Preliminary Validation: Conduct formative usability testing with a small group of end-users to validate core functionality and identify any critical usability issues before larger-scale deployment.

This structured yet iterative approach allowed the project to remain flexible. Feedback and findings from each phase fed into subsequent design adjustments, aligning the artifact closely with both user expectations and institutional requirements.

Data Collection

End Users (SUS and Scenario-Based Tasks): Selection combined convenience sampling (users with smartphones) and ease with aim of representing ages 18–55 and basic digital skills. Instruments included: (a) SUS (10 items, Likert 1–5) for usability assessment (Brooke, 2013); (b) scenario-based tasks (receiving an emergency notification, viewing it, sending an emergency report); (c) post-test mini-interview (5–7 minutes) for qualitative comments; and (d) procedure involving 2–3 minute orientation, performing three tasks without guidance with measurements (success/failure, task time, observations), SUS completion, and mini-interview. Sessions lasted approximately 20–25 minutes per user.

Expert Interviews: Purposeful selection of three key institutional profiles. Format: semi-structured, 25–35 minutes, online or face-to-face. Thematic guide covered: current alert flows, integration with existing systems, operational requirements (CAP, 112), technical/legal limitations, success criteria, and operational indicators.

Secondary Data: Strategic documents, standards, and methodological literature were used as secondary data to justify findings and design choices (Hevner et al., 2004; ISO 9241-210, 2019).

Data analysis methods

SUS and Quantitative Data: SUS calculation followed Brooke's scheme (0–4 per item conversion), sum × 2.5 → 0–100 (Brooke, 2013). Descriptive statistics included: SUS mean, standard deviation, score range; average task time, success rate.

Instrument reliability: Cronbach's α for SUS (expectation $\alpha \geq 0.70$).

Interpretive synthesis: comparison of results with guidance thresholds (SUS ≈ 68 average; >80 "good–very good") (Bangor et al., 2009). Result calculation used Python.

Qualitative Data: Thematic analysis following Braun and Clarke (2006): familiarization, initial coding, theme construction (instruction clarity, navigation intuitiveness, notification trust), theme review and definition, representative quote selection (without personal identification).

Triangulation: convergence between quantitative evidence (SUS/performance) and expert interview data to reach verifiable design recommendations.

Validity, Reliability, and Methodological Limitations

Content validity was ensured by relying on standards and literature for task design and SUS interpretation (ISO 9241-210; Brooke, 2013). Reliability increased through use of a standardized instrument (SUS) and clear observation rules; where

possible, Cronbach's α was assessed as an internal consistency indicator. Source triangulation (users, experts, documents/standards) and method triangulation (quantitative/qualitative) reduced bias risk. However, the design was formative: the small sample and controlled environment do not represent stress of real scenarios; therefore, field piloting and high-load testing is needed before large-scale deployment.

Ethical Considerations

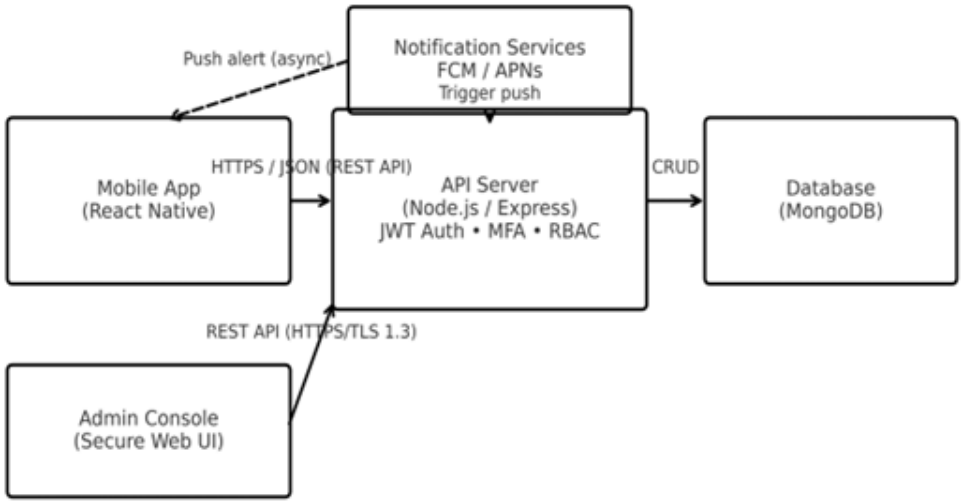
Research was conducted in accordance with scientific integrity and data protection norms: electronically documented informed consent with withdrawal option at any time; user data anonymization and expert pseudonymization (institutional role only); data minimization and encrypted storage/transmission with access limited to research team only; research data retention for five years in compliance with Law No. 9887/2008 and GDPR principles (Reg. (EU) 2016/679); and non-harm principle with scenarios and questions formulated to avoid unnecessary stress, with session interruption if concern arises.

Methods and Analysis

System Architecture

The system follows a three-tier architecture consisting of: (1) a presentation layer (the mobile client), (2) an application logic layer (the backend server), and (3) a data layer (the database). This classical design modularizes the application, allowing each layer to be developed, maintained, and scaled independently. In our implementation, the presentation layer is a React Native mobile application (deployable on both iOS and Android), the logic layer is a Node.js web service using the Express framework, and the data layer is a MongoDB document-oriented database. This technology stack constitutes a variant of the popular MERN architecture (MongoDB, Express, React, Node), enabling a unified JavaScript codebase across all tiers and efficient data exchange via JSON. Communication between the mobile app and server is facilitated exclusively through a RESTful API over HTTPS (secure HTTP), adhering to REST design principles. The client sends HTTP requests (to fetch alerts or submit a report) and receives JSON responses, while the server handles application logic and interacts with the database.

FIGURE 1. System architecture diagram



The mobile client communicates with the backend over the network, and the backend in turn reads from and writes to the database. In addition, the backend connects to external notification services (Firebase Cloud Messaging for Android and Apple Push Notification service for iOS) to deliver urgent alerts to user devices.

The React Native app component encompasses the user interface and client-side logic (state management, input handling, etc.). The Node.js/Express server component is organized into sub-modules for different functionalities (a *User* controller, an *Alerts* controller, an *Authentication* service, etc.), and it exposes a REST API interface to the client. The server also interfaces with the MongoDB database and with the external push notification services. This architecture promotes *loose coupling*; changes in one component or layer (for instance, switching the database engine) have minimal impact on others as long as the communication interfaces (API endpoints, data formats) remain consistent. Such decoupling improves maintainability and extensibility of the system. The design also inherently supports scalability and reliability: each tier can be scaled horizontally as needed (deploying multiple Node.js server instances behind a load balancer, or using MongoDB replication/sharding for large data volumes) and a failure in one server node or frontend instance will not collapse the entire system. Security considerations are addressed throughout the architecture, all client-server communications are encrypted (HTTPS) and token-based authenticated, sensitive data is encrypted or hashed before storage, and the database is configured with access control roles and backup replication policies for fault tolerance .

Functional Requirements (FR)

- FR-1 Emergency alerts (High): Push notifications with type, severity, area, and protective actions; audible + vibration; ≤5s delivery.
- FR-2 Geo-targeting (High): GPS-based delivery; saved places (home/work); target by radius or polygon.
- FR-3 Incident reporting (High): Quick form (category, text, auto-GPS, photo/video); routed to relevant authority; confirmation; optional contact.
- FR-4 Chatbot assistant (Medium): FAQ guidance from verified KB (multilingual); hands off to hotlines/live help when needed.
- FR-5 Interactive map (Medium): Live hazard zones, shelters/medical, user location; tap for details; updates as events evolve.
- FR-6 SOS / 112 (High): Prominent button; passes GPS + ID (if logged-in); fallback SMS with coords under poor connectivity.
- FR-7 Admin console (High): Secure web UI with MFA + RBAC, templates, and full audit trail (create/approve/send times, actor).
- FR-8 Alert history (Medium): List of received alerts with linked updates; filter by category.
- FR-9 Preferences (Low): Language, sounds, quiet hours; users cannot disable life-critical alerts.

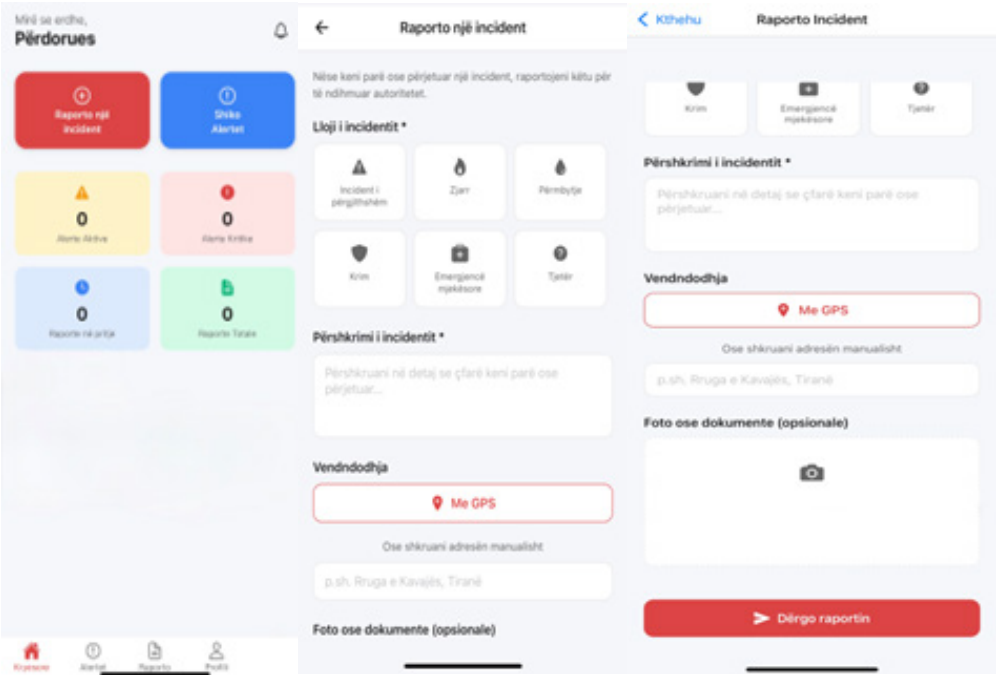
Non-Functional Requirements (NFR)

- NFR-1 Performance: End-to-end delivery <5s (normal); handle ≥100k concurrent and scale to ~2.8M users; typical queries <2s.
- NFR-2 Reliability: 99.9% uptime; ≥99.5% device reach; redundancy + automatic failover.
- NFR-3 Security: TLS 1.3+ in transit; AES-256 at rest; MFA for admin; detailed audit logs; regular security testing.
- NFR-4 Scalability: Horizontal scale (servers, workers); DB sharding/ partitioning; efficient batched push delivery.
- NFR-5 Maintainability: Modular code, docs; automated unit/integration tests; containerized deployments.
- NFR-6 Usability: One–two taps to key actions (view alert, report, SOS); plain language; accessibility features (larger text, screen readers).
- NFR-7 Interoperability: CAP-formatted alerts; RESTful APIs; interfaces compatible with telecom CB/112 systems.

Mobile Frontend

The frontend is a cross-platform mobile application developed with React Native. This choice allows a single codebase to natively deploy on both iOS and Android devices, providing nearly native performance and a consistent user experience on each platform. The mobile app's user interface was designed to be clean and minimalistic, focusing on critical functions to be used under emergency conditions. It includes screens for viewing active emergency alerts (a list or map of alerts in the vicinity), submitting a new incident report, and viewing user profile/settings. The UI adheres to human-centered design guidelines for clarity and simplicity (ISO 9241-210:2019), with consistent navigation and large, clear interactive elements to accommodate usage under stress. For example, posting a new incident report is accomplished through a single form with fields for incident type, location (which can be auto-obtained via GPS), description, and an optional photo attachment. Figure 2 shows an example of the app's data submission interface and alert display screen in the prototype.

FIGURE 2. Homepage and Send Report page



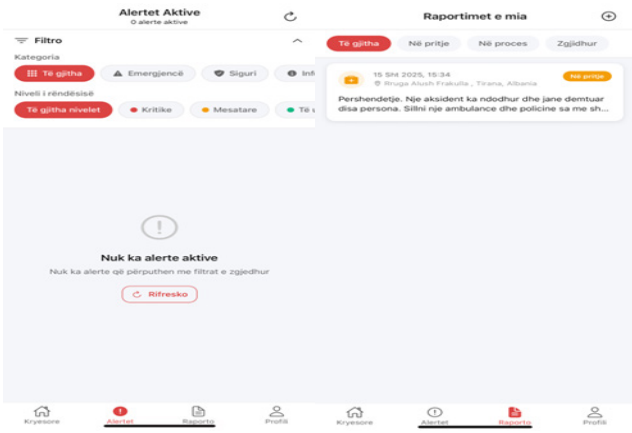
Also, to ensure responsiveness, the app manages local state (caching the latest alerts) and uses asynchronous API calls to the server.

Authentication on the mobile app is handled via JSON Web Tokens (JWT). Upon successful login, the server issues a signed JWT representing the user's identity and role. The app stores this token securely on the device (in the secure keychain storage) and attaches it to the Authorization header of subsequent requests. This approach eliminates the need to maintain sessions on the server and enables a stateless authentication model that scales well. Whenever the app is launched, it checks for a valid stored token to keep the user logged in across sessions. If the token is missing or expired, the user is prompted to log in again. In addition to login and registration interfaces, the app includes logic to handle incoming push notifications. When the user first installs or opens the app, it registers the device with the notification service (obtaining a device token) and sends this to the backend server. This enables the server to target that device for future alerts. If a critical alert is pushed by the server, the mobile OS will display it as a system notification (with a distinctive sound/vibration). Tapping the notification will automatically open the app and navigate to a detail screen showing the alert information and safety instructions. This push mechanism allows users to receive urgent warnings in near real-time even if the app is running in the background.

Backend and API

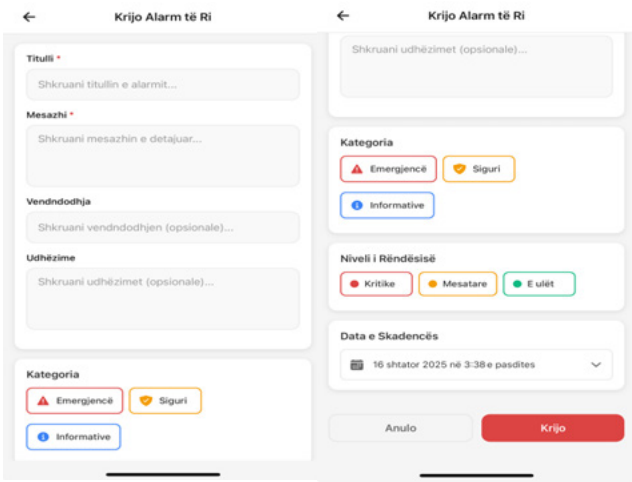
The backend of the system is implemented as a RESTful web service using Node.js with the Express framework. The server is structured according to a Model–View–Controller (MVC) pattern adapted for a web API context. The source code is divided into logical modules: routes (Express route definitions for each API endpoint), controllers (functions that handle requests and responses, encapsulating application logic), models (database schemas and data access using MongoDB via an Object-Document Mapper), and middleware components for cross-cutting concerns (authentication, logging). For each major resource in the system, an Express router is defined, for example, there are routes for user management, for emergency alerts, for incident reports, and for authentication. Each route maps HTTP endpoints (URLs and methods) to controller functions. For instance, the route POST /api/users/register invokes a controller that creates a new user account (after validating input and hashing the password), and POST /api/users/login verifies credentials and, on success, returns a signed JWT token to the client. Similarly, GET /api/alerts returns the list of active public alerts (for authenticated users), POST /api/reports allows a logged-in citizen to submit a new incident report (with details like type, description, and location), and POST /api/alerts (an endpoint restricted to authorized officials) allows an institutional user to issue a new emergency alert to the public.

FIGURE 3. Active alerts page and My reports page



Security and role enforcement are central in the backend design. It employs JWT-based authentication: clients must include the JWT token in the Authorization header of requests to protected endpoints. A custom Express middleware intercepts incoming requests to verify the token's validity and decode the user identity and role before the request reaches any controller. If the token is missing or invalid (expired or tampered), the request is rejected with an unauthorized error. Additionally, role-based authorization rules are implemented, for example, only users with an "admin" or "institution" role are permitted to invoke the alert issuance endpoint, preventing normal citizens from sending public alerts. This is achieved via another middleware that checks the authenticated user's role against the required privileges for certain routes.

FIGURE 4. Creating an alert as an Admin



In addition, passwords are never stored in plaintext in the database; the user registration controller hashes passwords (using a secure one-way hash function with salt) before saving, and login compares hashes to authenticate users. The server also logs important actions (such as alert creations, report submissions) in an audit log collection for accountability. Basic rate limiting is applied on public-facing endpoints to prevent abuse (to mitigate spam submission of fake incident reports).

Once an institutional user (or the system automatically) issues an emergency alert, the backend notifies all relevant users in real time via push notifications. The server is integrated with cloud messaging services, specifically, Firebase Cloud Messaging (FCM) for Android clients and Apple Push Notification service (APNs) for iOS. The system keeps track of each registered device's token (provided by the app). When a new alert is to be broadcast, the backend constructs a notification payload (including the alert title, message, and possibly a geographic target or category) and sends it through FCM/APNs to all devices or to devices in a specific area, as appropriate. User devices receive these as native push notifications accompanied by a distinctive alert sound even if the app is not active. This publish-subscribe design ensures fast and reliable dissemination of critical messages; leveraging the infrastructure of Google's and Apple's notification servers allows the system to scale to large numbers of recipients with minimal latency. For particularly urgent incidents reported by citizens, the backend can also perform an automatic escalation: for example, if a user report is classified as extremely critical (such as a major fire or explosion), the system can immediately generate a public alert based on that report's data and push it out to nearby users (after confirming validity). This feature provides a form of intelligent alerting, shortening the response time when every second counts.

Internally, the backend uses the Mongoose ODM (Object Data Modeling library for MongoDB) to interact with the database. Data schema definitions (models) are defined for each main entity, for example, a User model, Alert model, Report model, etc. These models enforce schema constraints (field types, required fields, validations) at the application level and provide convenient methods for database CRUD operations. Controllers use these model classes to query or update the database. By using a non-blocking, event-driven runtime (Node.js) and asynchronous I/O, the server can handle many concurrent requests efficiently, which is essential under high-load scenarios (during a widespread emergency when many clients may connect simultaneously).

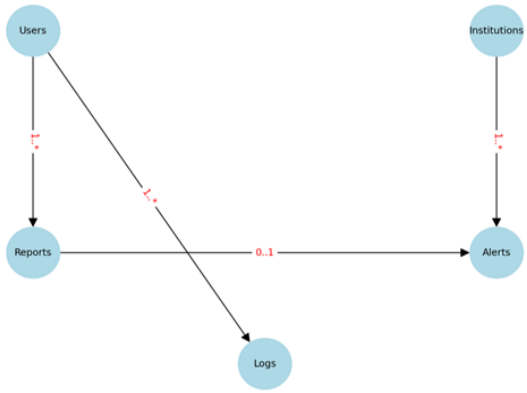
Database Design

The system's data is stored in a MongoDB NoSQL database. MongoDB was chosen for its flexibility in handling dynamic data structures and its high throughput for read/write operations on large datasets. The database contains several collections

corresponding to the core entities of the application: Users, Institutions, Alerts, Reports, and Logs. Figure 5 illustrates the simplified entity-relationship schema of these collections and their relationships. Each User document stores a user's information, including a unique user ID, name, email, a hashed password, role ("citizen" or "institution"), registration date, and status (active or suspended). Regular citizens have the "user" role, whereas authorized agency officials have an "institution" role (optionally linked to an entry in the Institutions collection). The Institutions collection contains entries for official agencies (Civil Emergency Directorate, State Police, Fire Department), with fields for institution name, type, jurisdiction/region, and contact details. This allows the system to associate certain user accounts with their respective institutions and to tag alerts or reports with the responsible agency.

The Alerts collection stores the emergency alerts issued to the public. Each Alert document includes an alert ID, a title (short description of the emergency, for example "Earthquake in Tirana"), a detailed message, a category (term for the emergency type: weather, earthquake, fire, security, etc.), a timestamp for when the alert was issued, an optional location (pinned coordinates or region), and a reference to the issuer (which could be an institution or admin user ID). Public alerts may also link to a specific report if the alert was generated as a response to a user-reported incident (creating a traceable connection between a citizen report and a follow-up public warning). The Reports collection contains incident reports submitted by end-users. A Report document captures details such as a report ID, the type of incident (*accident*, *fire*, *crime*), the description provided by the user, the timestamp of reporting, the location (GPS coordinates or an approximate address) of the incident, the userId of the reporter, and a status field. The status can be updated by authorities ("pending", "verified", "resolved") to reflect the progress of handling the incident. There is a one-to-many relationship between Users and Reports (each user may submit multiple reports) and reports can be associated with institutions via an assignedTo field (denoting which agency is handling the report). The Logs collection is used to track and audit actions in the system. Each log entry includes a log ID, an optional userId (if the action is tied to a specific user), an action description ("User Login", "Report Submitted", "Alert Issued"), a timestamp, and perhaps additional details such as the IP address or related record ID. These logs facilitate monitoring and can be analyzed to detect any misuse or to generate usage statistics.

FIGURE 5. ERD diagram for Users, Reports, Logs, Alerts, Institutions



In designing the database schema, normalization was balanced with performance. MongoDB's document model allows related data to be embedded within a single document if it is frequently accessed together, while other relationships are maintained via references (identifiers linking documents). For example, the geolocation details of a report (latitude/longitude) are stored as an embedded sub-document within the Report document for quick access and completeness. Meanwhile, links between users and their reports, or alerts and their issuing institution, are kept as references (IDs) rather than nested documents, since those entities are managed separately and using references avoids duplication and inconsistency. This hybrid approach achieves a flexible, semi-structured schema optimized for the app's query patterns. The database is also configured for fault tolerance and scalability. MongoDB is deployed in a replica set configuration: the primary node handles reads/writes, while secondary nodes maintain copies of the data, providing redundancy if the primary fails. This replication also enables scaling read operations across multiple nodes. Access to the database is protected by authentication credentials and role-based permissions, and all network traffic between the backend server and database is encrypted. In summary, the database design supports the application's need to efficiently store and retrieve emergency data, maintain data integrity (through relational links between collections), and scale to accommodate growing numbers of users and reports.

Results and discussion

Preliminary Evaluation

A preliminary evaluation of the prototype was conducted focusing on usability and stakeholder feedback. Usability testing was performed with 10 volunteer end-users (students and professionals) who were asked to install the app and perform

a set of core tasks (such as registering an account, reporting a sample incident, and responding to a test alert). After completing the tasks, participants filled out the System Usability Scale (SUS) questionnaire. The results were very positive: the average SUS score was 84 out of 100 (standard deviation ~5), with individual scores ranging from 78 to 90. This exceeds the typical SUS benchmark of ~68 (considered "average" usability); a score above 80 is generally characterized as indicating excellent usability. All participants were able to successfully complete the scenario tasks, and their subjective feedback was that the app was intuitive and the workflow (from receiving an alert to taking recommended actions) was clear. Some users noted that the interface felt "simple and clean," which is desirable for an emergency app. A few minor suggestions were made, such as providing an offline mode for viewing downloaded alerts or using more distinct alert notification sounds, which can be addressed in future iterations.

In addition to end-users, input from domain experts was gathered in relevant public safety fields. Interviews were conducted with three experts: a civil emergencies officer, a senior police officer, and a medical emergency doctor. Overall, the experts strongly supported the concept of the application and its potential benefits. The following are their opinions given in the interviews:

The civil emergency expert noted that a platform enabling real-time public warnings "could be an extremely valuable addition to the current emergency alert system," emphasizing that rapid information dissemination "can save lives, especially in natural disasters" and that location-targeted alerts are something that "currently is missing in many cases" (Civil Protection representative, paraphrased).

The police expert highlighted the app's utility for urgent law enforcement situations, such as ongoing violent incidents or AMBER Alert-type child abduction cases, as well as its use for quickly notifying citizens of roadblocks or evacuation orders. He remarked that *"this app could help us spread critical notifications much faster than traditional means like press conferences or TV broadcasts,"* underscoring the immediacy of push alerts.

The medical emergency expert similarly stressed that in public health crises or mass casualty events, *"accurate and timely information is the first remedy… This app can disseminate life-saving instructions within seconds to thousands of people."* Such feedback indicates that stakeholders see the prototype as addressing real needs in the current emergency communication ecosystem.

Despite the encouraging results, the evaluation also revealed important limitations. First, the usability test was limited to a small sample and conducted in a controlled setting; users were not under true emergency stress. Thus, the results, while indicative, are not conclusive of performance in a real crisis. Second, the system has not yet been tested under heavy load or in a wide-area deployment. Questions remain about how the infrastructure will handle tens or hundreds of thousands of concurrent users and rapid influxes of reports. Third, the success

of the platform depends on broad adoption and trust both by the public and by government agencies. The experts pointed out that integrating the app with official emergency operations would require formal agreements, training, and public awareness campaigns. Some also noted the challenge of filtering out false or duplicate reports from citizens to avoid information overload. These preliminary findings will guide the next steps of development, focusing on enhancing the system's robustness and preparedness for real world deployment.

Discussion and future work

The development and early evaluation of this emergency alert application demonstrate the feasibility and potential of such a system in strengthening public safety. The high usability scores and positive user feedback suggest that even non-technical users can navigate the app and respond to alerts effectively. Likewise, the enthusiastic responses from field experts indicate a strong demand for the capabilities provided by the system. If implemented at scale, the application could significantly improve emergency response workflows. For instance, it would enable authorities to gather structured, real-time incident data directly from citizens, leading to more informed and faster decision-making. It would also facilitate better inter-agency coordination: all relevant agencies (police, fire, medical, etc.) can gain a shared operational picture of an incident as information flows into the unified platform, addressing the common problem of siloed communications and leading to a more synchronized response. Furthermore, the app can serve as an official channel for urgent public communications, complementing existing methods. Traditional mass-alert channels like sirens, radio/TV broadcasts, SMS cell broadcasts would not be replaced, but augmented by this interactive mobile medium. By reaching users directly and allowing two-way interaction, the system can fill gaps (such as delivering rich context or receiving crowd-sourced updates) that one-way legacy channels cannot.

Despite these benefits, several challenges and areas for future work remain. A key priority is to move beyond the prototype and conduct extensive field testing. The plan is to deploy the app in a pilot program with a larger and more diverse user base, possibly in collaboration with a local municipality or civil protection unit. This would provide insight into user adoption rates and reveal usage issues in a real-world context. It is also critical to perform stress-testing and ensure the system can handle high volumes of traffic. Emergency scenarios can lead to sudden spikes in usage (thousands of people reporting the same earthquake or seeking information), so the backend and network infrastructure must be robust. Techniques such as cloud auto-scaling, load balancing, and database query optimization will be explored to guarantee reliability. Ensuring a high uptime and

low latency is paramount, for example, aiming for at least "three nines" (99.9%) availability in line with best practices for critical systems.

Another future direction is to enhance the app's intelligence and integration. The current prototype implements a basic rule-based alert classification (automatically elevating certain critical reports). In the future, machine learning models could be incorporated to analyze incoming reports or social media feeds to detect emerging crises faster, though any AI components would need thorough validation to avoid false alarms. It's needed also to recognize the importance of official integration: to be truly effective, the platform should be integrated with national emergency infrastructures (the 112-emergency call system or existing public warning systems). Achieving this will require close collaboration with government agencies and may involve adhering to common alerting protocols or data standards. On the organizational side, formal agreements and clearly defined operating procedures would be needed for authorities to confidently use the app during actual emergencies. The issue of public trust will also be a focus, much work will be put on communication strategies to ensure that citizens understand the app's purpose and that alerts sent via the app are authoritative. This includes cybersecurity hardening to prevent unauthorized or false alerts, learning from incidents like the false missile alert in Hawaii (2018) which underscored the damage a faulty alert can cause to public trust (FEMA, 2019).

Lastly, there are several feature improvements planned. These include multi-language support (important in Albania's context to reach ethnic minorities and tourists), accessibility enhancements for users with disabilities, and perhaps a web-based dashboard for emergency operators to manage alerts on a larger screen. Also, an aim is to incorporate a feedback mechanism so that after an alert or incident, users can receive confirmations or all-clear messages and provide feedback on the event's outcome. In summary, future work will address scaling up the system, tightening its integration with official workflows, and refining its features based on stakeholder input.

Conclusions

This paper presented the design and development of a mobile application for public safety and emergency alerts in Albania and discussed its preliminary evaluation. The research was motivated by the absence of a dedicated, centralized emergency communication platform in Albania's public safety landscape. By implementing a React Native mobile client and a Node.js/Express backend with a MongoDB database, was realized a functional prototype capable of delivering real-time emergency notifications and collecting citizen reports. The system's architecture

emphasizes scalability, security, and interoperability with established technologies (RESTful APIs, push notification services).

Initial preliminary testing results are promising, the application demonstrated a high level of usability (SUS score 84/100) and received encouraging feedback from both end-users and public safety experts. These findings suggest that the proposed solution is not only technically viable but also addresses real user needs in crisis situations. The police, emergency management, and medical experts that were consulted foresee the app strengthening the speed and effectiveness of public warnings and incident response. However, we emphasize that these results are preliminary. The prototype has yet to face the demands of a large-scale deployment or a live emergency. Additional development and validation are required to ensure the system's reliability, security, and integration into the broader emergency response ecosystem.

The development of this emergency alert app represents an important step toward modernizing crisis communication and public engagement in Albania. The system offers a new digital tool in the toolkit of disaster management, one that can empower citizens and authorities alike through instant, two-way information sharing. With further refinement and official support, such an application could significantly enhance public safety, helping to protect lives and property when emergencies occur. I intend to continue this work by collaborating with national institutions to pilot the system in real-world settings, implement the improvements identified, and ultimately move closer to deploying a fully operational emergency notification network for Albania.

References

BBC News. (2018). France withdraws SAIP emergency alert app after malfunctions. BBC News. (Archived news report).

BBC News. (2022). UK emergency alerts system. BBC News. Retrieved from BBC News website: https://www.bbc.com/news/uk-64371490

Bangor, A., Kortum, P. T., & Miller, J. T. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. Journal of Usability Studies, 4(3), 114–123.

Bass, L., Clements, P., & Kazman, R. (2013). Software Architecture in Practice (3rd ed.). Addison-Wesley.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2), 77–101.

Brooke, J. (2013). SUS: A retrospective. Journal of Usability Studies, 8(2), 29–40.

Bundesamt für Bevölkerungsschutz und Katastrophenhilfe (BBK). (2022). Warnsysteme in Deutschland (NINA, KATWARN, Cell Broadcast). Retrieved from https://www.bbk.bund.de/

Bundesnetzagentur. (2022). Technical directive: DE-Alert cell broadcast system. Retrieved from

https://www.bundesnetzagentur.de/SharedDocs/Pressemitteilungen/EN/2022/20220223_CellBroadcast.html

Cantelon, M., Harter, M., Holowaychuk, T., & Rajlich, N. (2014). Node.js in Action. Manning Publications.
Chodorow, K. (2013). MongoDB: The Definitive Guide (2nd ed.). O'Reilly Media.
Davies, R. (2022). Albania floods: Thousands isolated in Shkodër and Lezhë. Disaster Watch Journal, 7(2), 8–12.
European Commission. (2018). Directive (EU) 2018/1972 of the European Parliament and of the Council establishing the European Electronic Communications Code (EECC). Official Journal of the European Union, L321, 36–214.
European Commission. (2022). Report on mobile public warning system implementation in the EU. Brussels: European Commission.
European-Mediterranean Seismological Centre (EMSC). (2019). LastQuake usage metrics after the 2019 Albania earthquake. Retrieved from https://www.emsc-csem.org/
Federal Communications Commission (FCC). (2019). Emergency Alert System (EAS) overview. Retrieved from https://www.fcc.gov/general/emergency-alert-system-eas
Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures (Doctoral dissertation, University of California, Irvine).
FR-Alert. (2022). FR-Alert: France's new nationwide emergency alert system [Press release]. Paris: Ministère de l'Intérieur (France).
Guterres, A. (2022). Every person protected: Global Early Warning Initiative. United Nations. Retrieved from https://www.un.org/early-warnings/
Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly, 28(1), 75–105.
Institute of Statistics (INSTAT). (2020). Albania Earthquake 2019 – Key Statistics Report. Tirana: INSTAT.
International Organization for Standardization (ISO). (2019). ISO 9241-210:2019 Ergonomics of human-system interaction—Human-centred design for interactive systems. ISO.
Johnson, R. B., & Onwuegbuzie, A. J. (2004). Mixed methods research: A research paradigm whose time has come. Educational Researcher, 33(7), 14–26.
Jones, M. B., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT) (RFC 7519). Internet Engineering Task Force (IETF).
Jones, K., & Patel, N. (2020). Ensuring 99.9% uptime: Best practices for high-availability systems. Journal of Cloud Engineering, 5(4), 10–18.
Nwajana, A. O. (2025). Public safety mobile applications in West Africa. African Journal of Mobile Systems, 9(1), 95–110.
Policia e Shtetit. (2022). "Komisariati Dixhital" – Raport mbi përdorimin e aplikacionit dixhital të policisë. Tirana: Albanian State Police
React Native. (2023). Secure storage and keychain usage in React Native. React Native Documentation. Retrieved from https://reactnative.dev/docs/security#secure-storage
Regulation (EU) 2016/679. (2016). General Data Protection Regulation (GDPR). Official Journal of the European Union, L119, 1–88.
U.S. Federal Emergency Management Agency (FEMA). (2019). Hawaii false missile alert: After-action report. Washington, DC: FEMA.
U.S. Federal Emergency Management Agency (FEMA). (2021). Integrated Public Alert & Warning System Annual Performance Report, 2021. Washington, DC: FEMA.
U.S. Federal Emergency Management Agency (FEMA). (2023). Nationwide Emergency Alert System test results and improvements. Washington, DC: FEMA.
World Meteorological Organization (WMO). (2021). Atlas of mortality and economic losses from weather, climate and water extremes (1970–2019). Geneva: WMO.