CMPE 491 / SENG 491

Senior Design Project I

High Level Design Report

Burak Güçlü

Gizem Yüksel

İrem Özyurt

Zeynep Sude Bal

Advisor

Emin Kuğu

27.12.2024

TABLE OF CONTENTS

1. Introduction

1.1 Purpose of the System

Natural disasters and emergencies have very high risks to human life and mostly occur when less expected, causing destruction of great magnitude. This is especially critical in places like Turkey, which lies on a geographical location that easily develops earthquakes. The main goal of the system is to provide an effective solution in real-time disaster management to rapidly communicate and verify the safety of people in times of disasters. It shall be ensured that the coordination between the emergency response teams and individuals is effective to save lives and reduce the impacts of disasters by using advanced IoT technologies along with mobile applications.

The IoT sensors will continuously monitor the environmental conditions of seismic wave sensors, temperature sensors, and water level sensors. In case of any potential disaster, it will instantly identify the risk and begin the safety verification processes by notifying the users and their emergency contacts. The combination of both mobile and web-based platforms enables a centralized approach for the management of responses to disasters effectively and makes the system very valuable in improving the safety of the public during disasters.

1.2 Design Goals

The design of the system aims to achieve the following objectives:

Real-Time Detection and Notification: The system is designed for prompt disaster scenarios with the use of sophisticated IoT sensor technologies that track real-time environmental data. In this way, seismic sensors, temperature, and water level sensors will allow immediate early detection of disaster scenarios. It sends notifications to users relevant to the detected scenario in a matter of seconds so that individuals in affected areas are well informed to take precautions on time.

User Safety Verification: One of the major intentions of the system is to retrieve and process the safety responses of the users from disaster-hit regions. The status of their safety can be reported in an easy way through the mobile application, where they confirm whether they are safe or require help. This information gets processed in real time to prioritize the emergency response effort.

Reliable Communication: It will make sure that communications from users, emergency contacts, and response teams are sustained without break under hostile conditions. The system attains this by deploying failover mechanisms and other modes of communication like SMS notifications to ensure continuous operations when partial failures occur within the system.

Data Security and Privacy: Safeguarding user information is essential to the design of the system. The system encrypts data in transmission and storage, including personal details and location information. The system also implements sensitive information protection against unauthorized use through the use of access control policies that grant access to authorized people, such as emergency responders.

Scalability and Accessibility: The system is designed to handle a huge number of concurrent users, especially in disaster events affecting large populations. The system also implements accessibility interfaces for a wide range of users, such as multilingual support and features for people with disabilities, to ensure wide usability during emergencies.

Integration with Emergency Services: The system allows for seamless integration with third-party emergency response services. The system permits the sharing of vital data, such as the location of users and their safety status, in a timely manner to enhance coordination between response teams for effective disaster management operations.

Adaptability: It has been designed in a modular and extensible way, bearing in mind that continuous improvement is necessary. Such architecture will easily allow adding new functionalities, such as integration with other sensor types or adapting to new disaster scenarios, in order to keep the system updated and effective over time.

1.3 Definitions, Acronyms and Abbreviations

IoT (Internet of Things): The technology of having physical devices communicate data with each other over the internet.

Emergency Alert System: A communication platform for the dissemination of vital information in disaster scenarios.

Disaster Response Teams: Teams responsible for the rapid assistance and intervention at the time of emergency.

Sensor Data: Collected from environmental sensors that detect potential disaster scenarios.

Timeout: A predefined period after which the system automatically flags the user as unsafe if that user does not respond.

Dynamic Models: Models that represent the dynamic behavior of systems, such as sequence or state diagrams.

1.4 Overview

This system has a comprehensive solution towards disaster response capability. In brief, this includes IoT sensors, mobile applications, and a web-based dashboard. The real-time monitoring of situations along with communicating any emergency through the above sensing, handheld mobile, and a Web-based dashboard is depicted: Continuous monitoring of sensors reporting anomalies indicative of the possible occurrence of a disaster automatically triggers alerts to be pushed onto users in that affected area via the mobile app. Non-responses or negative responses will trigger automated notifications to emergency contacts and response teams, complete with a user's location and details.

The mobile application provides full support for user profile management, including the possibility to attach emergency contacts and edit already provided information. The web application is the core, since it enables the Emergency Response Team to monitor the statuses of safety, access important information, and coordinate interventions with much efficiency. These parts together build a system that prioritizes users for their safety, cuts down response times, and collectively enhances disaster management processes. In the light of this, the adoption of innovative design and advanced technologies in this particular project is basically committed to saving lives and making society resilient against natural hazards.

2. Proposed Software Architecture

2.1 Overview

The proposed system architecture is meticulously designed to address the challenges associated with disaster management by ensuring rapid and efficient communication during emergencies. The system leverages IoT-based sensors, a centralized web application, and a mobile application to deliver an integrated solution. This architecture not only enhances real-time communication but also provides a robust platform for data processing, notification management, and user interaction.

The system adopts a **layered architecture**, which is structured to ensure modularity, scalability, fault tolerance, and maintainability. Each layer has clearly defined responsibilities, allowing for independent development, testing, and enhancement of specific functionalities. This modular approach ensures that changes or improvements in one part of the system do not adversely affect other components.

Key Layers of System Architecture

1. **Data Collection Layer:**
    a. This layer consists of IoT sensors deployed in disaster-prone areas, such as seismic, temperature, and water level sensors. These sensors continuously monitor environmental parameters to detect potential disasters like earthquakes, floods, or fires.
    b. **Responsibilities:**
        i. Collect raw environmental data in real time.
        ii. Perform basic threshold-based analysis to detect anomalies.
        iii. Transmit collected data securely to the central server for further processing.
    c. **Features:**
        i. The sensors are designed to operate under harsh environmental conditions, ensuring reliability during extreme weather or disasters.
        ii. Low-power consumption ensures continuous operation, even during prolonged emergencies.

2. **Processing and Analysis Layer:**
    a. This is the core of the system, powered by the **central server**, which includes both a web server and a database server. The central server is responsible for processing sensor data, validating disaster scenarios, and triggering appropriate alerts and notifications.
    b. Responsibilities:
        i. Aggregate and analyze data from IoT sensors.
        ii. Validate sensor alerts to distinguish between real disasters and false positives.
        iii. Identify the severity and scope of the disaster to determine the affected areas and users.
        iv. Communicate with the Notification Management Subsystem to send alerts to users and emergency teams.
    c. Features:
        i. Fault-tolerant design ensures uninterrupted operations even if one server fails.
        ii. Data encryption and access control mechanisms protect sensitive information during storage and transmission.
        iii. Real-time data processing minimizes delays in detecting and responding to disaster scenarios.
3. **User Interaction Layer:**
    a. This layer provides interfaces for users to interact with the system via **web** and **mobile applications**. It ensures that users can receive notifications, provide safety status updates, and access disaster-related information.
    b. Responsibilities:
        i. Display real-time disaster updates and safety prompts to users.
        ii. Collect user responses (e.g., “I am safe” or “I am not safe”) and relay them to the central server.
        iii. Allow users to manage their profiles, including adding emergency contacts and updating personal information.
    c. Features:
        i. The mobile application, built using React Native, supports cross-platform functionality, ensuring accessibility for both Android and iOS users.

ii. The web application, developed with React.js and Node.js, provides a comprehensive interface for administrators and users.

iii. The user-friendly design and multilingual support make the system accessible to a diverse range of users.

Key Architectural Principles

The architecture is guided by several core principles to ensure its efficiency and reliability:

1. **Modularity:**
   a. The system is divided into smaller, independent subsystems, such as IoT Sensor Subsystem, Data Center Subsystem, Notification Management Subsystem, Web Application Subsystem, Mobile Application Subsystem, and Emergency Response Subsystem.
   b. Each subsystem has specific responsibilities, which reduces complexity and improves maintainability.
2. **Scalability:**
   a. The architecture is designed to handle an increasing number of users, sensors, and disaster events without performance degradation.
   b. Cloud-based servers ensure that resources can be scaled up or down dynamically based on demand.
3. **Fault Tolerance:**
   a. Redundancy mechanisms are in place to ensure that the system remains operational even if some components fail. For example, backup servers and data replication ensure continuity during hardware or network failures.
4. **Real-Time Processing:**
   a. The system prioritizes speed to ensure timely disaster detection and response. Alerts are processed and delivered within seconds of a disaster being detected.
5. **Secure Data Management:**
   a. All data, including user profiles, sensor readings, and notifications, is encrypted both in transit and at rest. Access to sensitive data is restricted to authorized personnel through multi-factor authentication.

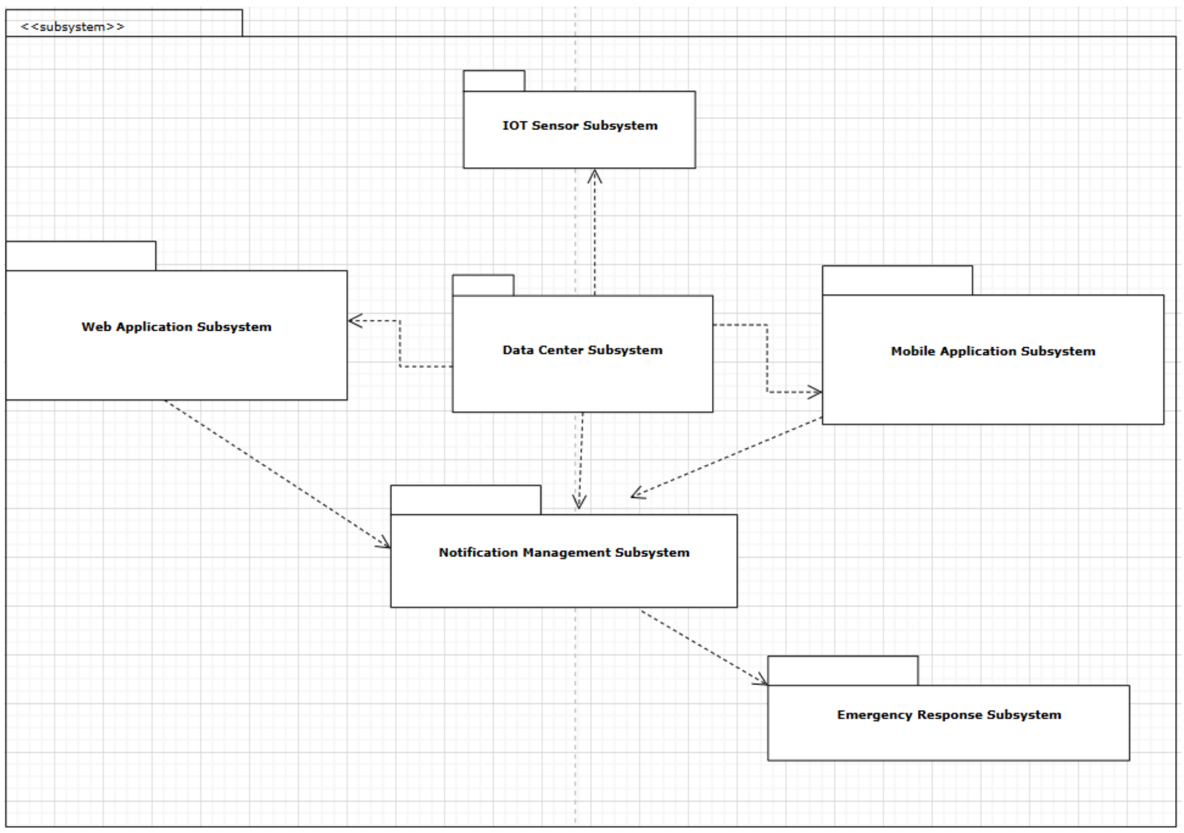
6. **User-Centric Design:**
    a. The system places a strong emphasis on usability, ensuring that users can easily interact with it during high-stress situations. Intuitive interfaces, minimal response times, and offline functionality enhance the overall user experience.

Operational Workflow

1. **Disaster Detection:**
    a. Sensors detect anomalies and transmit data to the central server in real time.
    b. The server analyzes the data, validates the disaster scenario, and determines its severity.
2. **Notification Delivery:**
    a. Validated alerts are sent to users in the affected region through the mobile application.
    b. Emergency teams are notified simultaneously, with detailed information about the disaster and impacted users.
3. **User Interaction:**
    a. Users respond to safety prompts, updating their status via the mobile application.
    b. The system aggregates user responses and updates the dashboard for administrators and emergency responders.
4. **Emergency Coordination:**
    a. Emergency response teams access real-time data through the system to plan and execute rescue operations.

By integrating these layers and adhering to these principles, the proposed architecture provides a robust framework for disaster management. It ensures that all stakeholders, including users, administrators, and emergency teams, can collaborate effectively to minimize the impact of disasters.

2.2 Subsystem Decomposition



<<subsystem>>
IOT Sensor Subsystem
Web Application Subsystem
Data Center Subsystem
Mobile Application Subsystem
Notification Management Subsystem
Emergency Response Subsystem

Figure 1 - System Decomposition

The diagram represents the decomposition of the system into distinct subsystems, each designed to handle specific functionalities. These subsystems work together to ensure the system achieves its goals of disaster detection, notification management, and response coordination. Below is a detailed explanation of each subsystem and their roles within the architecture:

1. IoT Sensor Subsystem

- **Purpose:**
  - The IoT Sensor Subsystem is responsible for collecting real-time environmental data from various sensors such as seismic, temperature, and water level sensors.
- **Functionality:**
  - Monitors environmental conditions and detects anomalies that could indicate a disaster.
  - Sends data to the **Data Center Subsystem** for processing.

- **Interaction:**
    - Data is transmitted to the **Data Center Subsystem** via secure protocols for further analysis.

2. Data Center Subsystem

- **Purpose:**
    - Serves as the central hub for data storage, analysis, and system-wide decision-making.
- **Functionality:**
    - Processes raw data received from the **IoT Sensor Subsystem** to validate disaster scenarios.
    - Provides processed data to the **Notification Management Subsystem** for user and emergency team alerts.
    - Facilitates data exchange with the **Web Application Subsystem** and **Mobile Application Subsystem**.
- **Interaction:**
    - Receives data from the **IoT Sensor Subsystem** and communicates processed outputs to the **Notification Management Subsystem**.
    - Allows the **Web Application Subsystem** and **Mobile Application Subsystem** to access user and system data for interaction and updates.

3. Notification Management Subsystem

- **Purpose:**
    - Manages the delivery of alerts and notifications to users and emergency response teams.
- **Functionality:**
    - Sends real-time notifications to the **Mobile Application Subsystem**, such as “Are you safe?” prompts.
    - Forwards critical updates and location information to the **Emergency Response Subsystem**.

- **Interaction:**
  - Receives disaster alerts and status updates from the **Data Center Subsystem**.
  - Sends notifications to users via the **Mobile Application Subsystem** and emergency information to the **Emergency Response Subsystem**.

4. Mobile Application Subsystem

- **Purpose:**
  - Provides a user-friendly interface for individuals to interact with the system during disaster scenarios.
- **Functionality:**
  - Allows users to receive disaster alerts, respond with their safety status, and manage profiles and emergency contacts.
  - Displays real-time updates and disaster information.
- **Interaction:**
  - Receives notifications from the **Notification Management Subsystem**.
  - Sends user responses and updates to the **Data Center Subsystem**.

5. Web Application Subsystem

- **Purpose:**
  - Provides a centralized platform for administrators and users to monitor and manage disaster events and user information.
- **Functionality:**
  - Enables viewing of system status, user responses, and disaster scenarios.
  - Supports manual triggers for notifications and disaster management.
- **Interaction:**
  - Communicates with the **Data Center Subsystem** to retrieve and update user and system data.

6. Emergency Response Subsystem

- **Purpose:**
  - Supports emergency teams by providing critical information for quick and effective disaster response.
- **Functionality:**
  - Receives detailed reports, including affected users' information, locations, and disaster severity levels.
  - Coordinates with external systems to organize emergency responses.
- **Interaction:**
  - Receives notifications and updates from the **Notification Management Subsystem**.
  - Provides feedback on disaster resolutions to the **Notification Management Subsystem** if required.

Subsystem Interactions

The system relies on the seamless interaction of these subsystems to perform effectively:

1. **IoT Sensor Subsystem** provides raw data to the **Data Center Subsystem**.
2. The **Data Center Subsystem** processes this data and identifies potential disasters.
3. Disaster alerts are routed to the **Notification Management Subsystem**, which:
   a. Sends alerts to users via the **Mobile Application Subsystem**.
   b. Sends critical updates to the **Emergency Response Subsystem**.
4. Users interact with the system through the **Mobile Application Subsystem** or the **Web Application Subsystem**, providing feedback or managing their profiles.
5. The **Emergency Response Subsystem** utilizes the information for rapid response planning and execution.

2.3 Hardware/Software Mapping

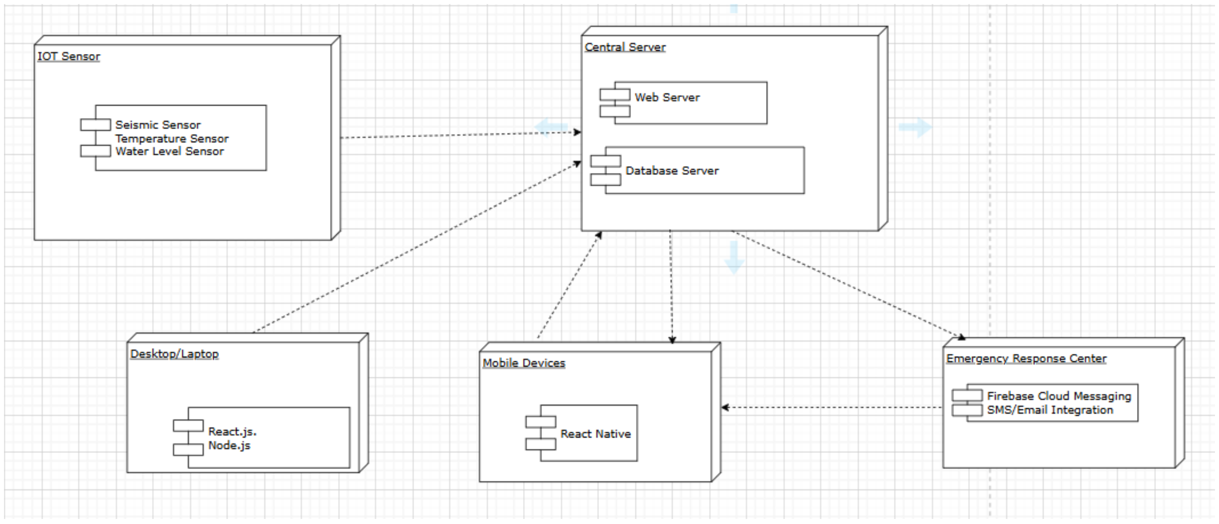


Figure 2 - Deployment Diagram

This diagram illustrates the hardware and software components of the system and their interactions. Each component is designed to meet the functional requirements of the system, supporting critical operations such as data collection, processing, user interaction, and emergency response.

1. IoT Sensor Subsystem

- **Hardware:**
  - Seismic Sensor
  - Temperature Sensor
  - Water Level Sensor
- **Purpose:**
  - Collect environmental data and detect potential disaster conditions.
- **Data Flow:**
  - Collected data is securely transmitted to the **Central Server** for processing.

2. Central Server

- **Hardware:**
  - Web Server
  - Database Server

- **Software:**
  - The web server processes sensor data and manages communication with user interfaces.
  - The database server stores user information, sensor data, and incident logs.
- **Purpose:**
  - Analyze incoming sensor data, validate disaster scenarios, and trigger notifications.
- **Communication:**
  - Receives data from the **IoT Sensor Subsystem**.
  - Interacts with **Desktop/Laptop** and **Mobile Devices** subsystems.

3. Mobile Devices

- **Hardware:**
  - Smartphones and tablets (Android/iOS).
- **Software:**
  - React Native: Cross-platform mobile application development.
- **Purpose:**
  - Allow users to receive disaster notifications and provide safety responses.
  - Manage user profiles and emergency contact details.
- **Communication:**
  - Notifications are received from the **Central Server**.
  - Responses and profile updates are sent back to the server.

4. Desktop/Laptop

- **Hardware:**
  - Desktop and laptop computers.
- **Software:**
  - React.js: Web application user interface.
  - Node.js: Backend logic and API integration.
- **Purpose:**
  - Enable users to manage disaster scenarios and safety statuses via a web-based platform.

- **Communication:**
    - Exchanges data with the **Central Server**.

5. Emergency Response Center

- **Hardware:**
    - Servers and communication devices.
- **Software:**
    - Firebase Cloud Messaging: Efficient notification distribution.
    - SMS/Email Integration: Communication with emergency teams.
- **Purpose:**
    - Receive user safety statuses and disaster site information to coordinate rapid responses.
- **Communication:**
    - Receives information from the **Central Server** and sends out notifications.

Overall Flow

- **IoT sensors** collect environmental data and send it to the central server.
- The **Central Server** processes sensor data, triggers notifications, and provides information to users and emergency response teams.
- Users interact with the system via **mobile devices** or **web applications** to manage profiles and respond to alerts.
- The **Emergency Response Center** organizes response efforts and maintains communication with the system.

This structure demonstrates how the hardware and software components work together to optimize disaster management and response.

2.4 Persistent Data Management

Persistent data management is a cornerstone of the system's ability to store, retrieve, and secure critical information efficiently, especially during disaster scenarios. The system employs a hybrid database architecture combining relational and real-time databases to meet the demands of performance, scalability, and data integrity. The primary relational database, PostgreSQL, serves as the backbone for storing structured data, including user profiles, emergency contacts, sensor metadata, and disaster response records. This relational model allows for effective organization and rapid querying through indexed and normalized tables. Complementing this is Firebase, a real-time database platform that ensures instantaneous synchronization of safety responses and notifications across the system's central server, mobile applications, and web interfaces.

First in the sequence of the data lifecycle, comes ingestion. IoT sensors send their read times in real-time to Firebase, which in turn propagates on all the components bound to it and logs into PostgreSQL for storage concurrently. The responses of the users to the safety prompt are also collected and then analyzed continuously for the system to have correct statuses on them, up to date. It provides periodic backups stored at geographically distributed locations. The backup systems automatically restore themselves on redundant servers in case of system failures, assuring that there is minimum downtime and no loss of vital information.

Security plays an important role in managing persistent data. All the data in transit is encrypted with the use of SSL/TLS protocols, while at rest it is encrypted with AES-256. Moreover, additional encryption of sensitive information related to the user's location and their emergency contacts adds even more protection to their privacy. Role-based access is based on mechanisms of access control. User and administrator accounts are further secured by multi-factor authentication. Audit trails create a very granular log of who accessed or changed the data, therefore offering full transparency and adherence to regulatory compliance.

Another critical consideration is scalability for pertaining to the approach adopted by the system toward managing its data. Whenever the demand goes up significantly higher-than for example situations of general disasters-the systems dynamically scale up their PostgreSQL installation in accordance with such demand spurts, simply by adding replicas for increased reads and thereby load-balancing and fanning out incoming queries into several database instances. With these together, a robust, safe, and sufficiently capable infrastructure pertaining to data management challenges which a given disaster scenario can pose upon the system becomes achievable.

2.5 Access Control and Security

In a critical system such as an emergency response application, robust access control and security measures are critical. We aim to apply secure software development principles in our applications. Unauthorized access can lead to data breaches, system outages and potentially hinder emergency response efforts.

User Authentication:

Multi-factor authentication (MFA): We have decided to implement MFA for all user accounts, including administrators and potential citizens using the app. This may include a combination of the following:

Something you know: Passwords (strong, regularly changed)

Something you own: Mobile device, security tokens (we will use captcha)

Something you are: Biometric authentication (if applicable and appropriate)

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a verification method used to determine whether the user entering a website is a real human or a bot. CAPTCHA requires solving a word or visual puzzle, usually consisting of broken letters. While humans usually perform this task with ease, it is quite difficult for computers to solve such visual or linguistic complexities. In this way, it is possible to distinguish the difference between human and computer.

User Authorisation

Role-Based Access Control (RBAC): We plan to improve security by defining different roles with different levels of access to system functions and data. There are currently 3 different user types in our project. Admin-User

Administrators: Full system access, including user management, data analysis and system configuration.

Citizens: Limited access, primarily to submit emergency requests and receive notifications.

Principle of Least Privilege: We aim to implement one of the most important principles of secure software development by giving users only the minimum privileges necessary to fulfil their tasks.

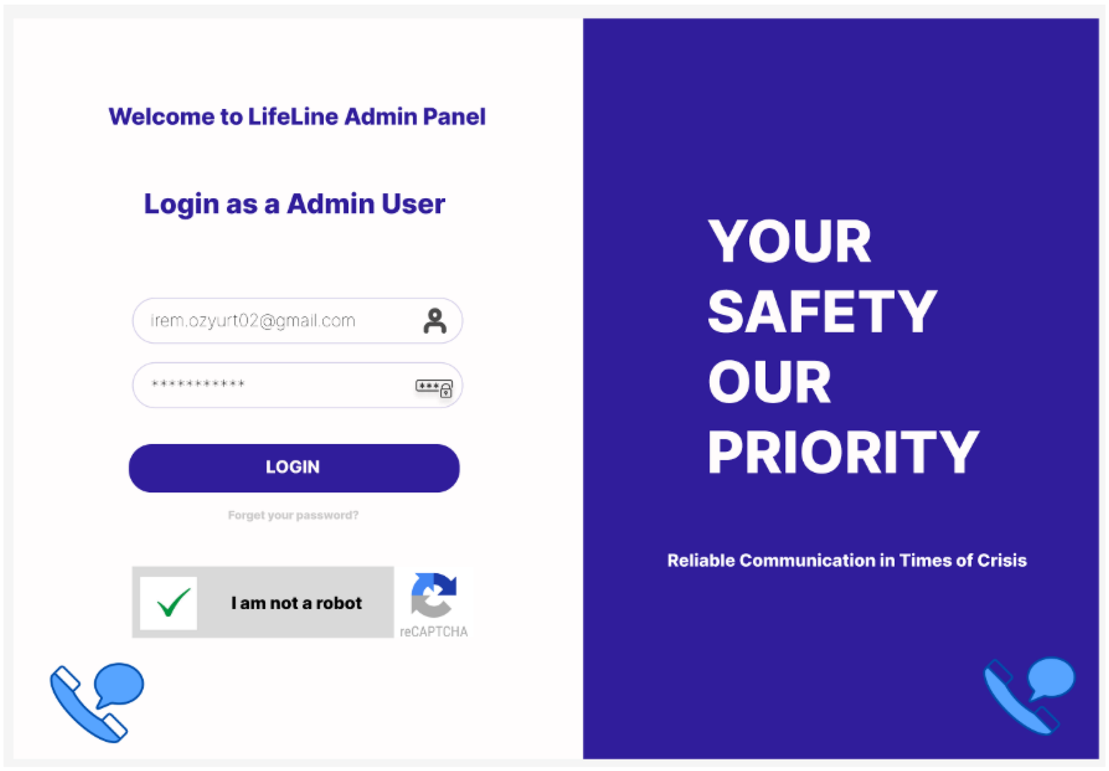


Figure 3 - CAPTCHA Design 1

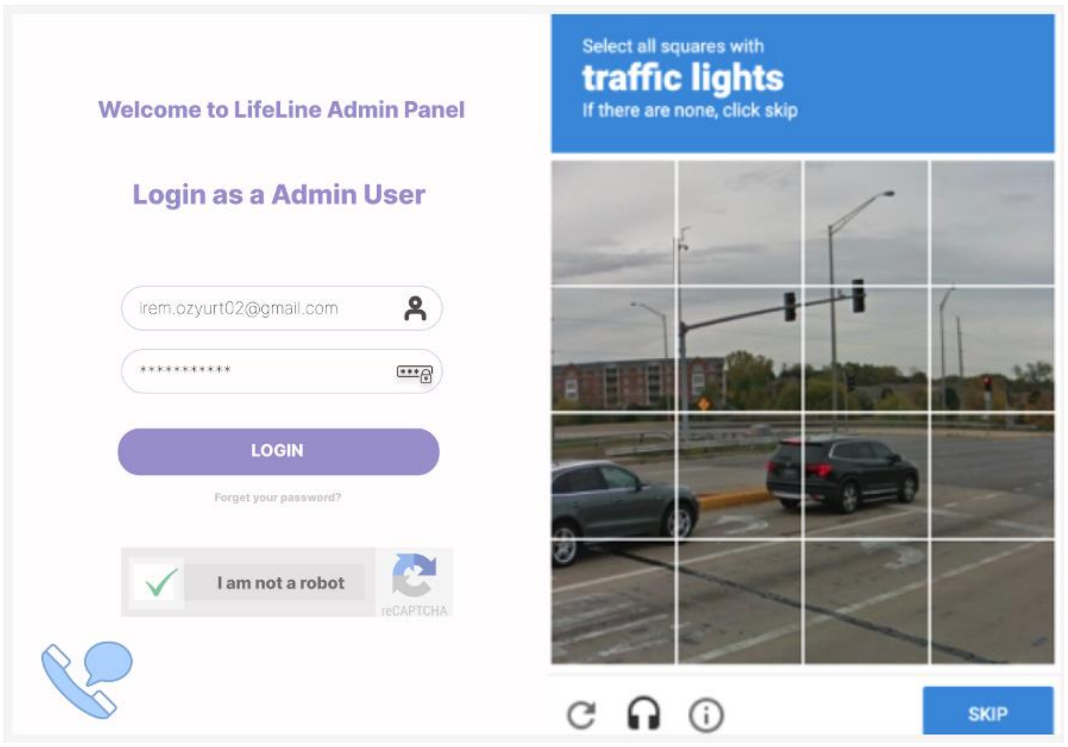
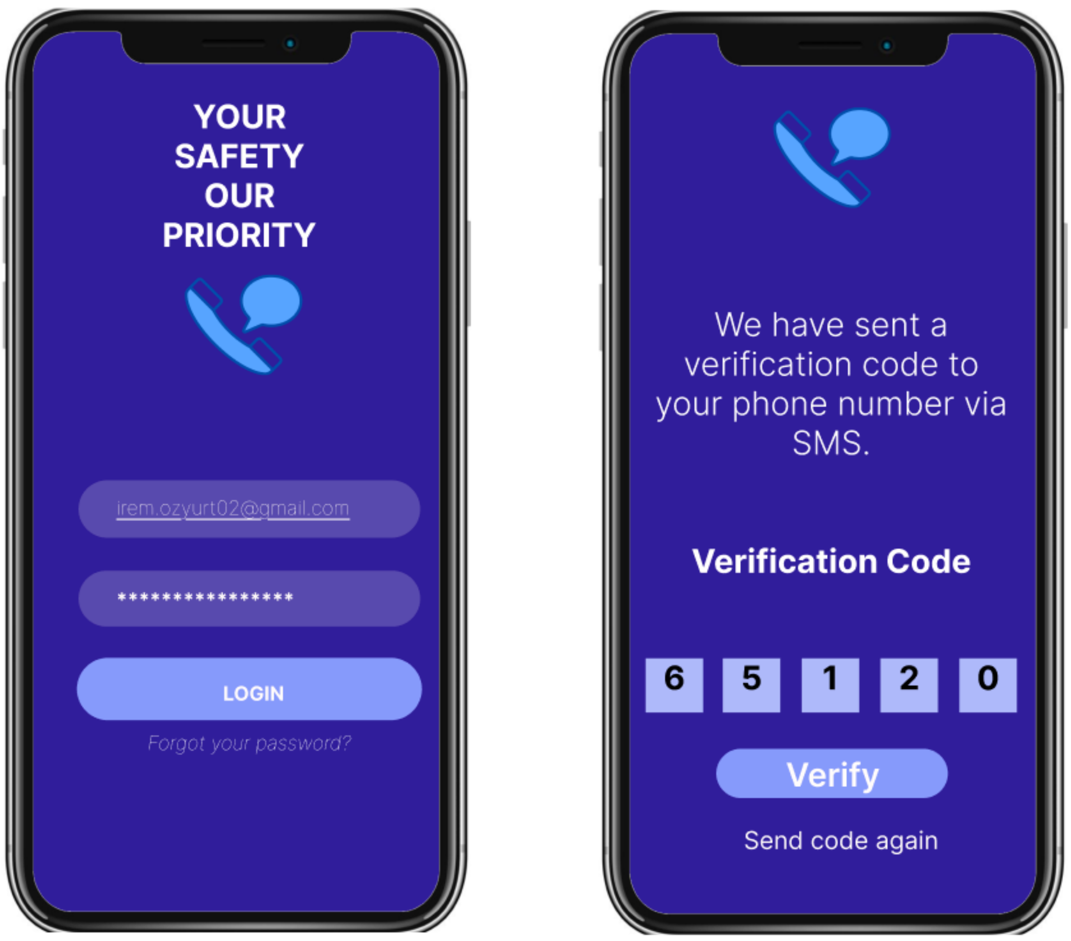


Figure 4 - CAPTCHA Design 2



We have sent a verification code to your phone number via SMS.

Figure 5 - 2FA Design

How to do 2-Factor Authentication (2FA) in Mobile Application?

2-factor authentication (2FA) is a very important step to maximize user security in our mobile applications. 2FA requires users to verify their identity using not only their password, but also another verification factor (for example, SMS sent to a phone number, code in a verification application, or biometric data).

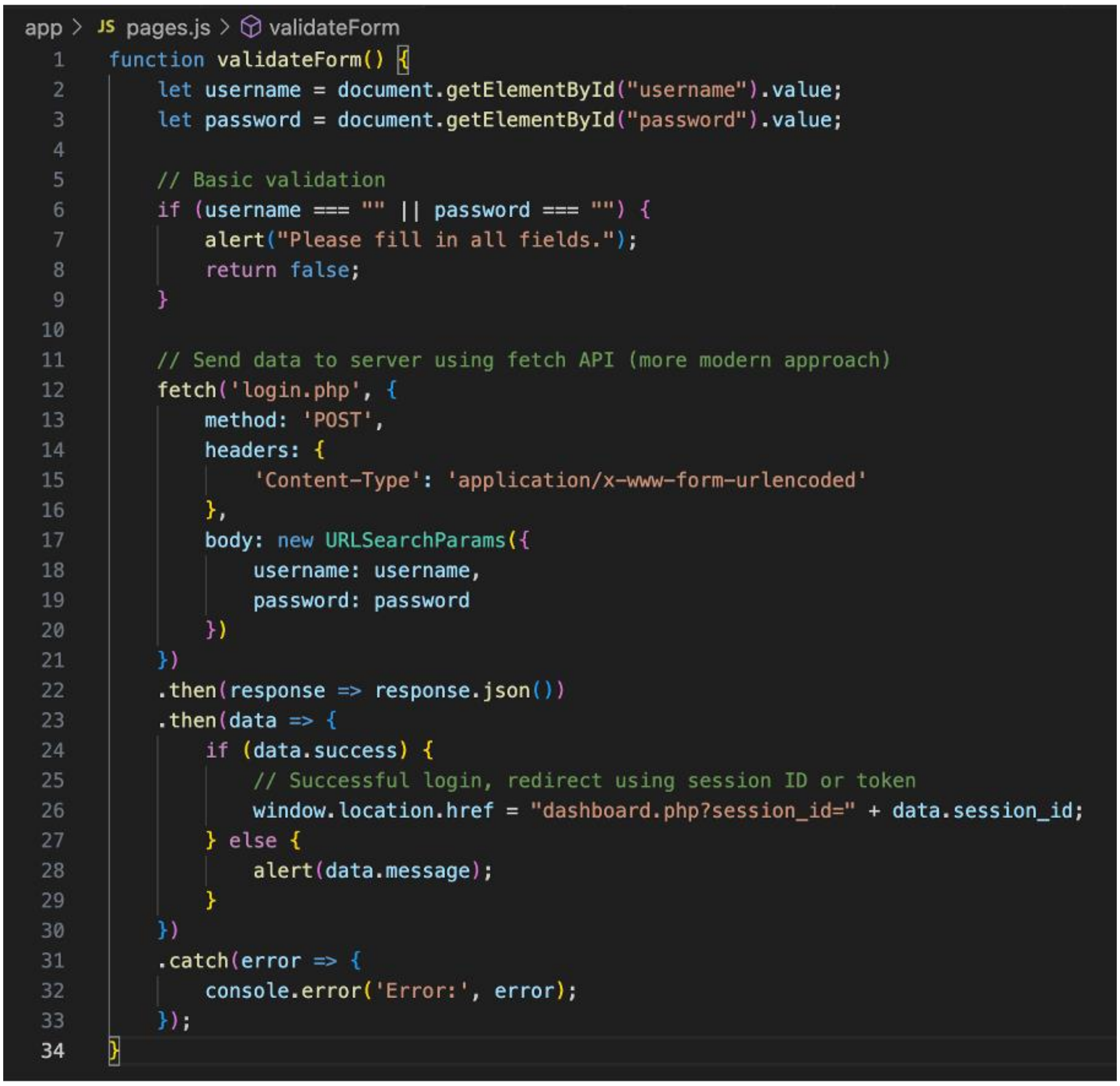
Advantages of 2FA

Security: Prevents unauthorized access to the account even in case of password theft.

User Experience: Allows users to feel more comfortable with account security.

Application Reliability: Increases the reliability of the application and allows you to reach more users.

We are doing input validation to prevent Broken Access Control and Injection from OWASP Top 10 vulnerabilities, and we aim to perform user_id validation to prevent login by changing the url.



```javascript
function validateForm() {
    let username = document.getElementById("username").value;
    let password = document.getElementById("password").value;

    // Basic validation
    if (username === "" || password === "") {
        alert("Please fill in all fields.");
        return false;
    }

    // Send data to server using fetch API (more modern approach)
    fetch('login.php', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded'
        },
        body: new URLSearchParams({
            username: username,
            password: password
        })
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            // Successful login, redirect using session ID or token
            window.location.href = "dashboard.php?session_id=" + data.session_id;
        } else {
            alert(data.message);
        }
    })
    .catch(error => {
        console.error('Error:', error);
    });
}
```

Figure 6 - Example Secure Code for Possible Vulnerabilities

2.6 Global Software Control

2.6.1 System Architecture and Components

The system aims to communicate and manage the situation quickly and effectively after a disaster. Firstly, sensors (devices that detect situations such as earthquakes, floods and fires) continuously monitor environmental data. When a critical situation is detected, the sensors transmit this data directly to the Firebase database. The Notification Service, which is integrated into the data in Firebase, analyses the incoming data and, if certain thresholds are exceeded, sends a notification to mobile application users with the question ‘Are you safe?’. Users respond to this notification via the mobile application (React Native) and the information they enter is saved in Firebase via the Node.js-based API server.

Meanwhile, on the administrative side of the system, administrators (admin) monitor both the data from the sensors and the responses of the users in real time using the React-based web interface. The admin panel receives data from the API server and monitors it in a visual and understandable way. In a critical situation, the administrator directs the relevant units (e.g. ambulance, fire brigade, AFAD) based on user responses and sensor data

The system works separated into frontend and backend layers. The frontend contains the mobile application and web interface, while the backend contains the Firebase database and Node.js based API server. Sending notifications and controlling threshold values are managed by a special Notification Service in the backend. The instant processing of sensor data and the monitoring of user feedback by the admin ensures that the system provides a fast and reliable communication infrastructure.

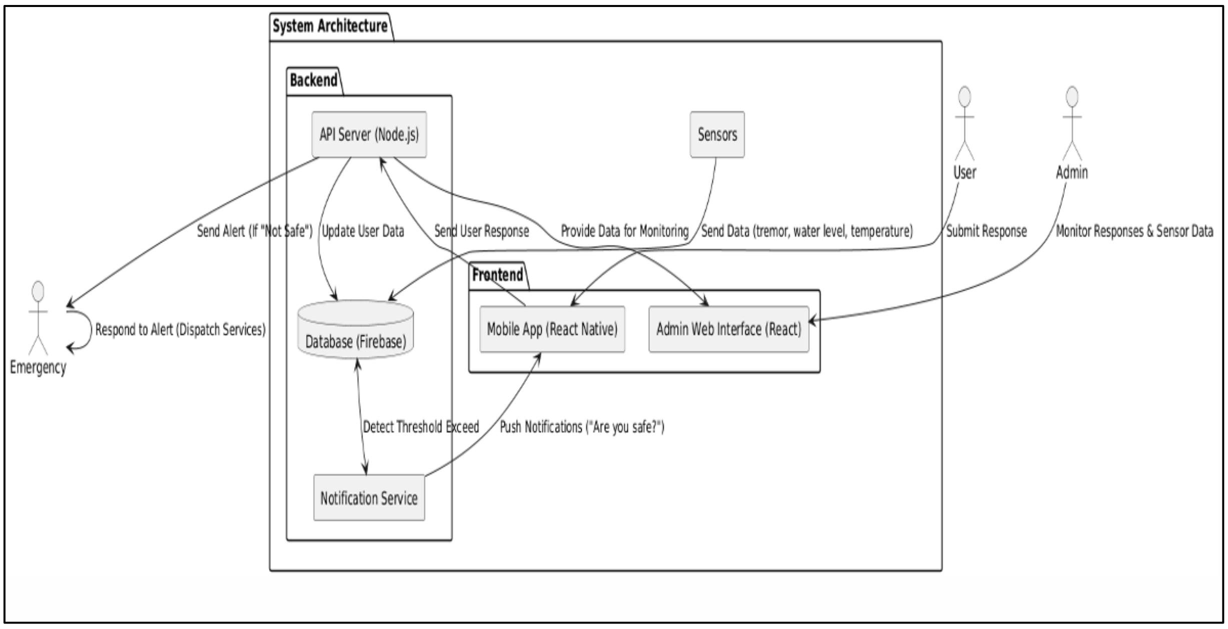

System Architecture
Backend
API Server (Node.js)
Sensors
User
Admin
Send Alert (If "Not Safe")
Update User Data
Send User Response
Provide Data for Monitoring
Send Data (tremor, water level, temperature)
Submit Response
Monitor Responses & Sensor Data
Frontend
Emergency
Respond to Alert (Dispatch Services)
Database (Firebase)
Mobile App (React Native)
Admin Web Interface (React)
Detect Threshold Exceed
Push Notifications ("Are you safe?")
Notification Service

Figure 7 - System Architecture

2.6.2 Control Flow

Sensors and Algorithms

In the first step, sensors monitor the surrounding environment. For situations such as earthquakes, floods or fires, predefined thresholds are analysed. If the sensor data exceeds this threshold, the event is triggered and the sensors notify the event detection algorithms.

Event Detection and Mobile Notification:

Event detection algorithms verify the existence of a disaster situation by analysing the data from sensors. As a result of the verification, a notification is sent to the relevant mobile application according to the type of event (earthquake, flood, fire). This notification includes a message explaining what happened and a button where users can select their security status (for example: ‘I am safe’ or ‘I am not safe’).

User Responses:

Users using the mobile application report their status through the options in the notification. Responses are received by the mobile application backend and stored in the Firebase database. The security status of each user matches this data.

Monitoring Responses in the Web Application:

Users' responses are monitored in real time by the system's web application. This web application is an interface designed only for administrators (admin). Administrators can see all responses from users and identify which areas have security problems.

Sending Notifications to Related Units in Critical Situations:

When ‘I am not safe’ responses are received from users, the system processes these responses and sends the details of the relevant incident to emergency units (for example: fire brigade, ambulance, AFAD). In addition, information is sent via SMS to the emergency contacts previously specified by the user.

Closure and Follow-up:

After all these processes are completed, the system is continuously monitored by administrators. When the situation is under control or the disaster situation is over, the system closes the processes and switches to monitoring mode until a new event is detected.

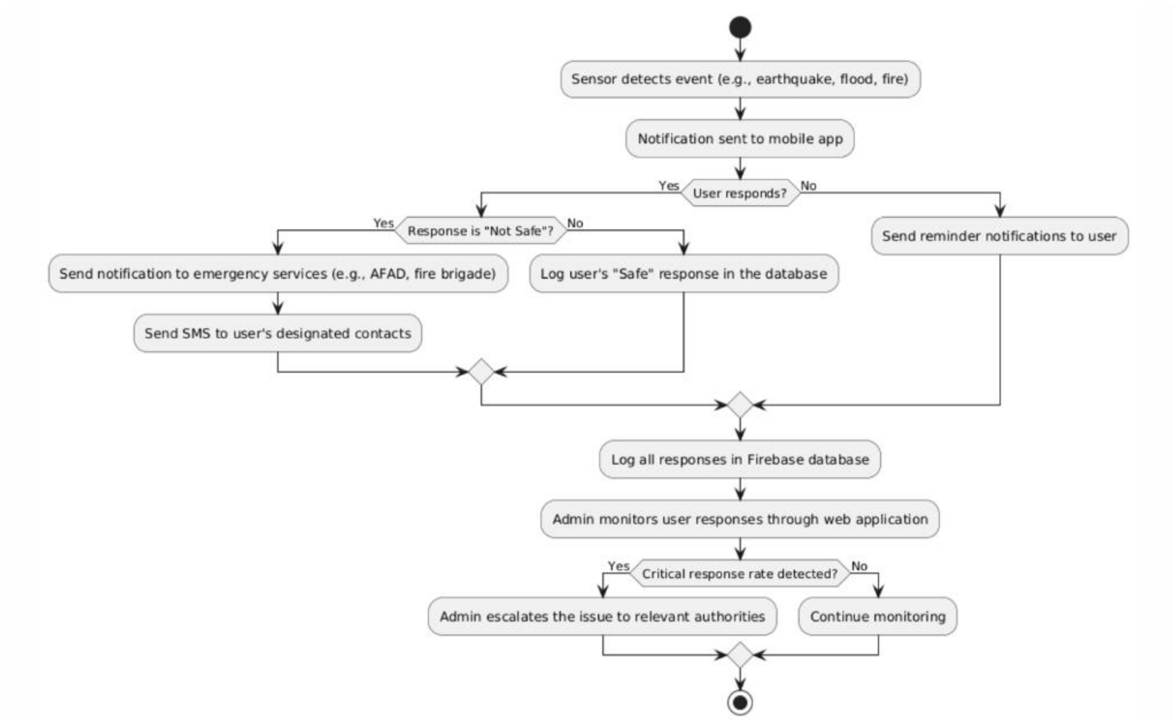

Sensor detects event (e.g., earthquake, flood, fire)
Notification sent to mobile app
Yes
User responds?
No
Yes
Response is "Not Safe"?
No
Send reminder notifications to user
Send notification to emergency services (e.g., AFAD, fire brigade)
Log user's "Safe" response in the database
Send SMS to user's designated contacts
Log all responses in Firebase database
Admin monitors user responses through web application
Yes
Critical response rate detected?
No
Admin escalates the issue to relevant authorities
Continue monitoring

Figure 8 - Control Flow

2.6.3 Data Flow

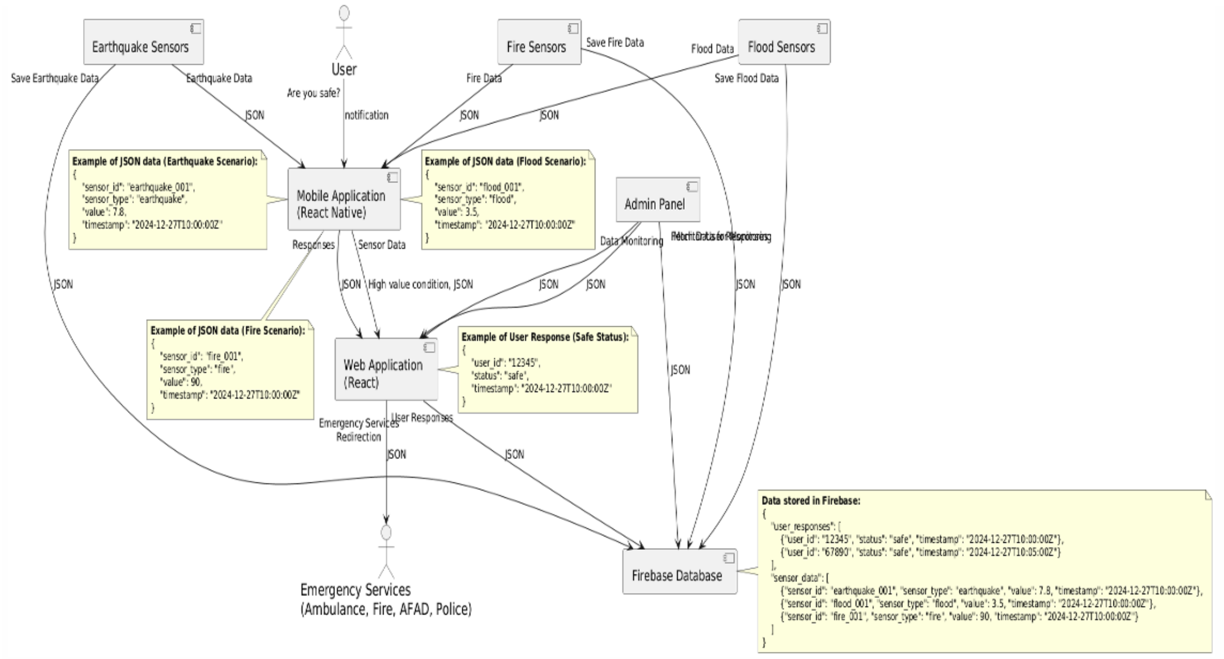
Responses from the User: Users respond to notifications received on the mobile application. Responses are transmitted to the Web application in JSON format and saved in the Firebase database. These responses include states such as ‘I am safe’.

Sensor Data Collection: Sensors detect values that exceed a certain threshold and this data is transmitted to the Web application via the mobile application. The web application receives this data and saves it in the Firebase database.

Emergency Routing: The web application analyses critical data from sensors with ‘I am not safe’ responses received from the user and transmits it to emergency teams.

Admin Monitoring:

Admin can monitor all user responses and sensor data through the Web application. The admin panel receives this data in JSON format and displays the trust status of users, sensor data and emergency routing when necessary.



Example of JSON data (Earthquake Scenario):

```
{
  "sensor_id": "earthquake_001",
  "sensor_type": "earthquake",
  "value": 7.8,
  "timestamp": "2024-12-27T10:00:00Z"
}
```

Example of JSON data (Flood Scenario):

```
{
  "sensor_id": "flood_001",
  "sensor_type": "flood",
  "value": 3.5,
  "timestamp": "2024-12-27T10:00:00Z"
}
```

Example of JSON data (Fire Scenario):

```
{
  "sensor_id": "fire_001",
  "sensor_type": "fire",
  "value": 90,
  "timestamp": "2024-12-27T10:00:00Z"
}
```

Example of User Response (Safe Status):

```
{
  "user_id": "12345",
  "status": "safe",
  "timestamp": "2024-12-27T10:00:00Z"
}
```

Data stored in Firebase:

```
{
  "user_responses": [
    {"user_id": "12345", "status": "safe", "timestamp": "2024-12-27T10:00:00Z"},
    {"user_id": "67890", "status": "safe", "timestamp": "2024-12-27T10:05:00Z"}
  ],
  "sensor_data": [
    {"sensor_id": "earthquake_001", "sensor_type": "earthquake", "value": 7.8, "timestamp": "2024-12-27T10:00:00Z"},
    {"sensor_id": "flood_001", "sensor_type": "flood", "value": 3.5, "timestamp": "2024-12-27T10:00:00Z"},
    {"sensor_id": "fire_001", "sensor_type": "fire", "value": 90, "timestamp": "2024-12-27T10:00:00Z"}
  ]
}
```

Figure 9 - Data Flow

2.7 Boundary Conditions

The design of this system considers numerous boundary conditions for reliability and continuity in the case of different operations and failures. In the startup process, sensors in IoT devices connect with the central server and check their operational readiness through diagnosis. At the same time, the server will boot up the main memory with all the vital data, such as user profiles, emergency contact lists, and disaster scenarios for immediate monitoring and processing purposes. Once these preparatory steps are done, real-time data collection and monitoring start, where the system is ready to detect an event and automatically respond to such a disaster event.

The controlled shutdown is one in which normal or scheduled maintenance is used to prevent loss of data and handle transitions within the system. Cleanup of active notifications pending and then all the transactions related to the data, and finally execution of backup operations running, are done to preserve the integrity of data. During emergency shutdowns or partial system failures, the system enters a degraded mode where core functionalities, such as user notifications and data logging, remain operational while non-critical features are temporarily suspended.

Failure management mechanisms are integral to the system's resilience. For example, the failure of a sensor due to redundancy means that some neighbor sensors will provide compensatory but overlapping data coverage. Besides that, the system creates alerts to intervene manually where necessary. In case there is a connectivity failure, the mobile applications are enabled to store user responses locally, which will be synchronized with the central server upon restoration. Failover configurations manage server failures by automatically performing a switchover of operations to redundant servers with no interruption of the service. Real-time replication between primary and secondary active servers minimizes data losses from such transitions.

The system is designed to handle high-load scenarios effectively. Large-scale disasters automatically scale up with extra computational and storage resources, so as to maintain performance. Similarly, the notifications and user responses are sorted in priority order about the severity of situations and proximity to the disaster sites for effective and focused response. The performance monitoring tool at regular intervals traces system metrics for any bottleneck possibilities and thereby takes proactive actions in their mitigation.

The extreme cases, like unresponsive users, are handled through escalation protocols. If a user can't respond to the set of safety prompts within their time, the system does an auto-notification to that user's emergency contact as well as the authorities. With location-based escalation, the responses are effectively deployed in regions of greatest need. Of course, in cases of wide-scale sensor failures, manual overwrites enable operators to input data regarding a disaster and trigger notifications appropriate for the situation. Besides, the system handles parallel disasters-for example, earthquakes followed by flooding-and alerts them in order of their criticality and according to the needs of user safety. These comprehensive strategies have implications for ensuring that the system remains robust and adaptive in order to maintain critical operations, even under challenging conditions.

3. Subsystem Services

In the Post-Disaster Communication project, subsystems are designed to ensure user safety, establish efficient communication, and provide rapid response in disaster situations. These subsystems work in an integrated manner, optimizing the overall performance of the system and improving the user experience. The main purpose of the subsystems is to provide real-time data flow, quickly detect and evaluate events, and initiate appropriate actions accordingly. Utilizing innovative technologies such as IoT sensors, mobile applications, and web-based management platforms, this system fills the communication gaps that arise in disaster scenarios. Data from sensors are processed on a central server and the situation is analyzed. This information is then transmitted to mobile application users and relevant emergency teams. The system creates a large-scale and comprehensive communication network, encompassing not only the physical infrastructure but also software-based solutions. Priority has been given to critical design goals such as performance, reliability, and scalability in the processes of collecting, processing, and transmitting data. This system has a strong infrastructure supported by the latest technologies to ensure user privacy and data security.

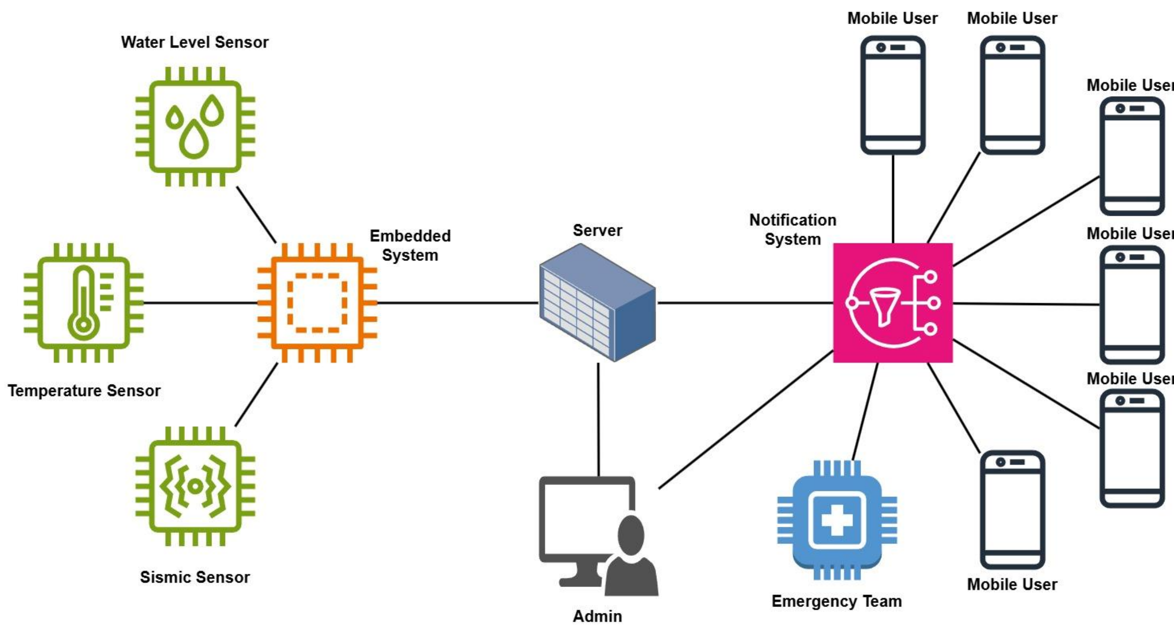
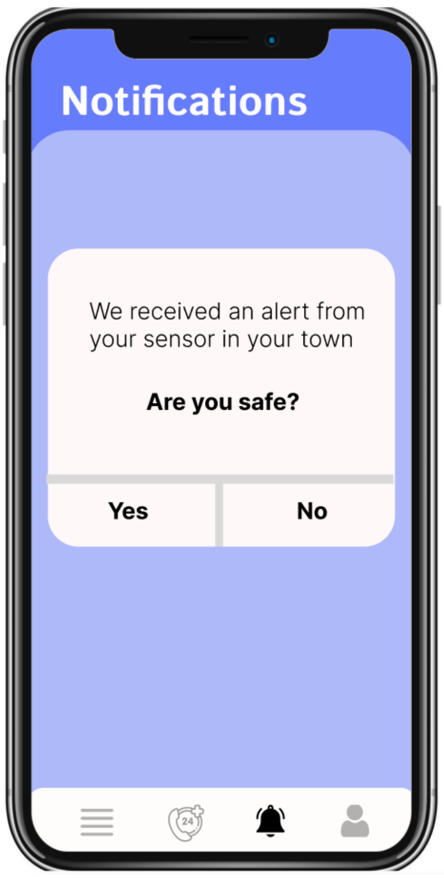


Figure 10 - System General Architecture Diagram

3.1 Mobile Application Subsystem

The mobile application provides a platform that allows users to update their safety status and quickly get the help they need. Users can create and log in to their accounts, edit their profile information, and manage emergency contacts. The application sends users "Are you safe?" notifications in the event of a disaster and collects their responses. These responses are transmitted to the central system and evaluated in real time. If users do not feel safe, the system automatically sends information to the relevant emergency teams and the people specified by the user. In addition, the application can send periodic notifications so that users can regularly update their safety status. The user experience is optimized to ensure fast and effective action even in stressful situations. The design is adapted to work across devices and supports Android and iOS platforms. It offers features such as high contrast modes and screen reader compatibility in terms of accessibility. The mobile application is a powerful tool that can be used to make post-event relief organizations more effective.


Notifications

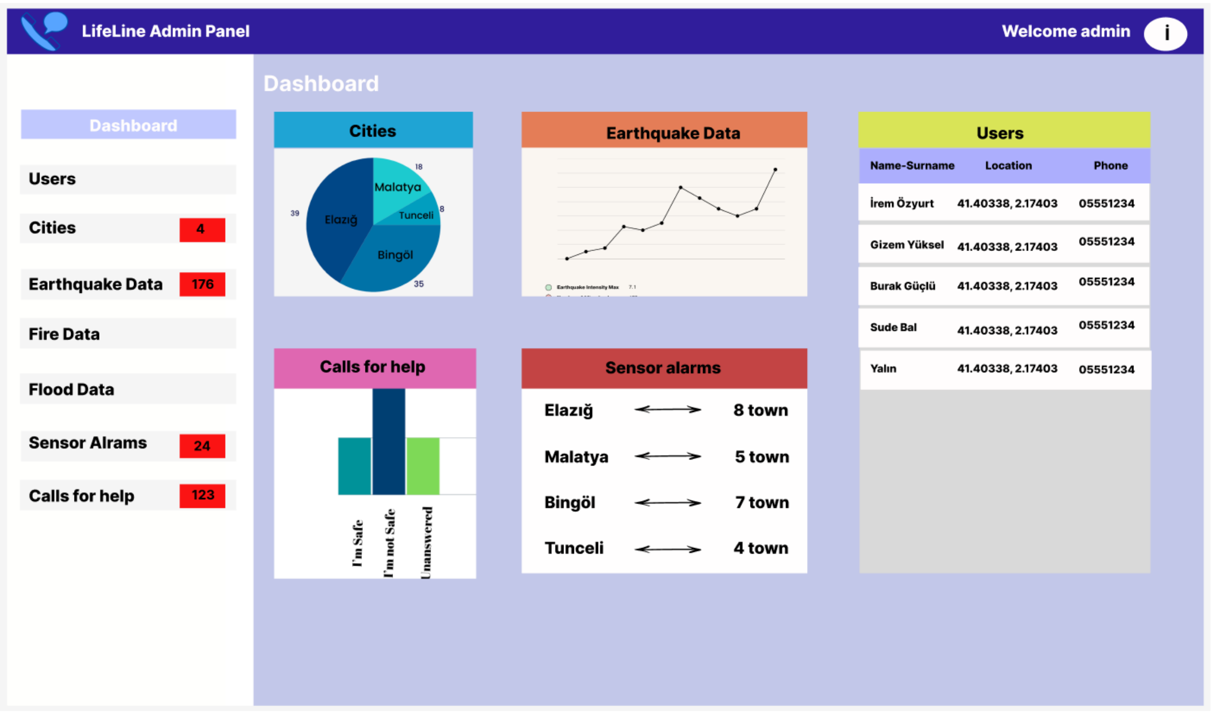
We received an alert from your sensor in your town

Are you safe?

Figure 11 - Screenshot of the Mobile Application Interface

3.2 Web Application Subsystem

The web application plays a key role in disaster management and acts as the central management panel of the system. This platform collects and evaluates all data from sensors. Operators can monitor incoming notifications in real time and track the safety status of users. The web application analyzes user responses and identifies critical situations that need to be quickly directed to the relevant units. It provides a dashboard that visualizes the general status of users and sensors. A map-based interface provides geographical monitoring of both users and disaster-affected areas. In addition, the web application stores data on past events and generates reports. These reports can be used to evaluate and improve system performance. The web application also allows authorized administrators to manage user accounts and sensor statuses. The platform is supported by security measures; it allows access only to authorized users and all data flows are encrypted. This application serves as a center that accelerates disaster management processes and makes them more effective.



Users	
Cities	4
Earthquake Data	176
Fire Data	
Flood Data	
Sensor Alrams	24
Calls for help	123

Sensor alarms

Elazığ	⟷	8 town
Malatya	⟷	5 town
Bingöl	⟷	7 town
Tunceli	⟷	4 town

Users

Name-Surname	Location	Phone
İrem Özyurt	41.40338, 2.17403	05551234
Gizem Yüksel	41.40338, 2.17403	05551234
Burak Güçlü	41.40338, 2.17403	05551234
Sude Bal	41.40338, 2.17403	05551234
Yalın	41.40338, 2.17403	05551234

Figure 12 - Web Application Dashboard Screenshot

3.3 IoT Sensor Subsystem

IoT sensors form the basis of the system's real-time data collection and analysis capabilities. Devices such as seismic waves, temperature, and water level sensors continuously monitor environmental changes to determine the potential for disaster. These sensors are strategically placed in disaster areas and detect abnormal conditions that exceed certain thresholds. Detected abnormalities are transmitted from the sensors to the central system via Arduino or similar microcontrollers. The system evaluates the incoming data and initiates appropriate actions after verifying its accuracy. The sensors are designed to withstand harsh environmental conditions. For example, temperature sensors are resistant to high temperatures during fires, and water level sensors are resistant to long-term humidity during floods. These sensors require regular maintenance and calibration; otherwise, they may produce false positive or negative results. The sensor subsystem enables early intervention in incidents, reducing loss of life and increasing the overall disaster preparedness of society.

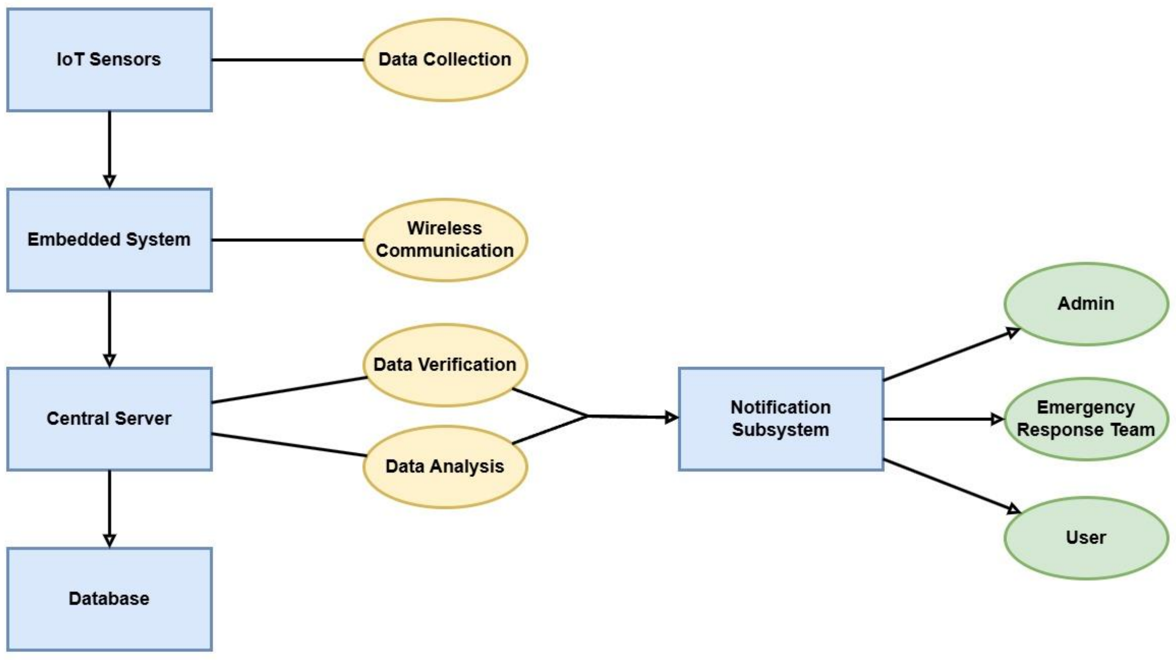


Figure 13 – Data Flow Diagram Showing Data Transfer of IoT Sensors

3.4 Notification Subsystem

The notification subsystem performs the function of informing users and emergency teams in the event of a disaster. After analyzing the data from the sensors, this system sends notifications directly to users' devices. Critical alerts reach users by bypassing the "Do Not Disturb" modes of the devices. The system collects users' responses, records them in the central database and informs emergency teams when necessary. If the user does not respond, the system repeats the alerts after a certain period of time and informs the user's relatives. The notification subsystem is an effective tool that allows not only individuals but also emergency response teams to intervene in incidents in a timely manner. The rapid and reliable delivery of notifications is a critical factor affecting the overall success of the system. The infrastructure of this system is designed to be scalable so that it can operate smoothly even in high traffic conditions.

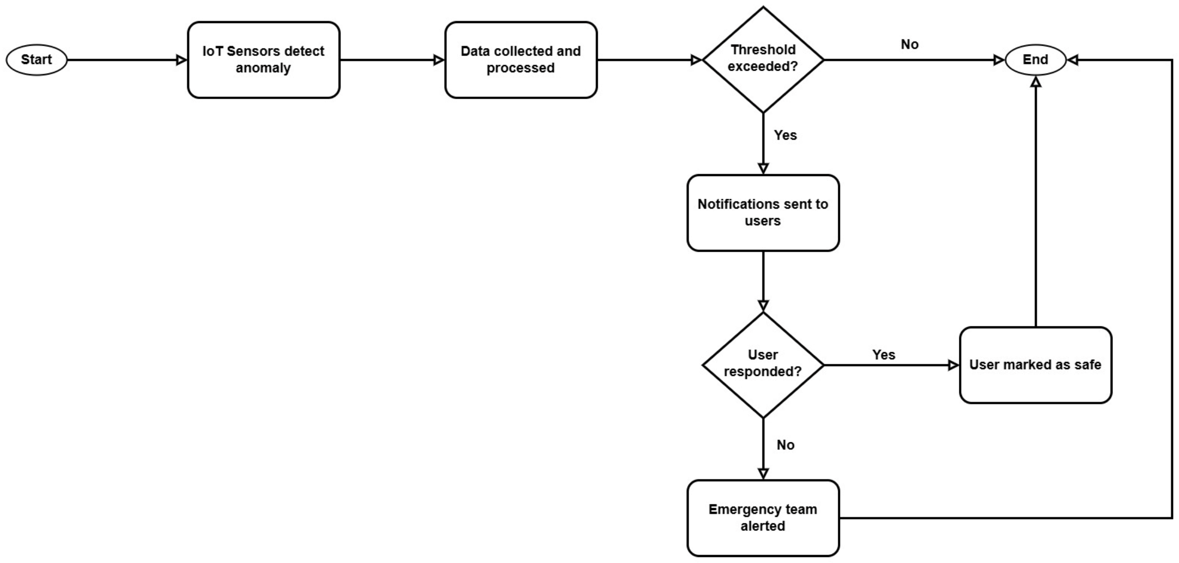


Figure 14 – Process Diagram Showing the Notification Flow and How Notifications Reach the User

4. Glossary

Term	Definition
IoT (Internet of Things)	The technology that allows physical devices to connect to the internet to collect and share data.
Emergency Alert System	The system is used to quickly share information and send warnings during natural disasters.
Sensor Data	Data obtained from the physical environment.
User Interface	Visual and functional components through which users interact with a system.
Access Control and Security	The access control policies and security measures of the system are explained.
Global Software Control	The overall control flow and processing logic of the system is explained.
Boundary Conditions	It specifies how the system behaves in unusual situations or boundary conditions
Subsystem Decomposition	Describes how to decompose the system into smaller, manageable subsystems or modules.

5. References

[1] Chandrakumar, C., Prasanna, R., Stephens, M., & Tan, M. L. (2022). Earthquake early warning systems based on low-cost ground motion sensors: A systematic literature review. Frontiers in Sensors, 3, 1020202.

[2] Chen, F. H., Shieh, H. L., & Tu, J. F. (2023). Development of Earthquake Detection and Warning System Based on Sensors. Sensors & Materials, 35.

[3] Karacı, A. (2018). IoT-based earthquake warning system development and evaluation. Mugla Journal of Science and Technology, 4(2), 156-161.

[4] Ray, N. K., & Turuk, A. K. (2017). A framework for post-disaster communication using wireless ad hoc networks. Integration, 58, 274-285.

[5] Deepak, G. C., Ladas, A., Sambo, Y. A., Pervaiz, H., Politis, C., & Imran, M. A. (2019). An overview of post-disaster emergency communication systems in the future networks. IEEE Wireless Communications, 26(6), 132-139.

[6] Matracia, M., Saeed, N., Kishk, M. A., & Alouini, M. S. (2022). Post-disaster communications: Enabling technologies, architectures, and open challenges. IEEE Open Journal of the Communications Society, 3, 1177-1205.