CMPE 492 / SENG 492

Senior Design Project II

Test Plan Report

Burak Güçlü

Gizem Yüksel

İrem Özyurt

Zeynep Sude Bal

Advisor

Emin Kuğu

17.04.2025

Table of Contents

1. Introduction

1.1. Project and Operation

Natural disasters and unforeseen emergencies rank among the most significant dangers to human life, particularly in nations situated in active earthquake regions like Turkey. In these situations, it is crucial to swiftly assess the safety condition of disaster victims, guide emergency response teams to the correct locations, and provide the required assistance promptly.

Traditional disaster management systems can prove insufficient in post-disaster responses due to delays in information gathering and assessment, communication issues, and coordination challenges. At this moment, establishing a fast and flexible disaster management system by leveraging the advantages of modern technology has become an essential requirement to reduce fatalities and enhance societal resilience against disasters.

This project is an innovative solution developed to detect environmental risk factors in real time during and after a disaster, to determine security situations by communicating directly with disaster victims, and to instantly transmit this information to the relevant emergency units. The system continuously monitors environmental conditions through various sensors such as seismic, temperature, water level, based on the Internet of Things. When any danger is detected, the system automatically initiates a security verification process.

An instant notification is sent to individuals in the disaster area via the developed mobile application in the form of "Are you safe?" Users are expected to respond by selecting one of the options "I am safe" or "I am not safe". If no response is received within a certain period of time or the option "I am not safe" is selected, the system automatically alerts pre-defined emergency contacts and relevant official emergency units such as AFAD, fire department, health teams, with location information.

The mobile application also allows users to register emergency contact information (relatives, neighbors, etc.) in the system before a disaster, keep their personal profile information up to date, and quickly request assistance when needed. The web-based management panel allows emergency teams to visualize the security status of users in the disaster area on a map in real time, to monitor, identify risky areas, and to intervene in incidents as quickly as possible. In this way, it is aimed not only to ensure individual security, but also to carry out the overall disaster management process in a more coordinated, efficient, and rapid manner.

1.2. Importance of Testing

The reliability, precision, efficiency, and ease of use of this essential disaster management system are crucial. This system, aimed at crucial objectives like reducing loss of life and enhancing emergency responses, is anticipated to function effectively during actual disaster situations. Consequently, a detailed testing procedure is an essential component for the project's success.

The subsequent key matters will be addressed through a robust test plan and execution:

- **Precision and Dependability:** The system elements (IoT sensors, mobile app, web interface, communication framework) will be guaranteed to gather, process, and transfer precise data as intended. It is aimed to prevent incorrect or incomplete information flow and prevent incorrect guidance and delays.
- **Performance and Scalability:** The system will be tested to operate stably and quickly even in the event of a high number of users and simultaneous events. The system's durability and scalability will be secured against heavy communication traffic that may occur during disasters.
- **Usability and User Experience:** The user-friendly, understandable and easy-to-use nature of the mobile application and web panel will be tested. It will be ensured that disaster victims and emergency teams can use the system effectively even in stressful environments.
- **Security:** The confidentiality and security of user data, protection against unauthorized access and the system's resistance to cyber attacks will be tested. Sensitive information will be stored and transmitted securely.
- **Integration:** The seamless integration of different system components such as IoT sensors, mobile application, web panel, map services and communication infrastructure with each other and the ability to exchange data will be tested.
- **Error Identification and Resolution:** Throughout the testing phase, possible errors, shortcomings, and improvement opportunities in the system will be recognized, feedback will be given to the development team, and these issues will be addressed.

2. Features to Be Tested

This section details the main features of the developed disaster management system that will be comprehensively addressed during the testing process. Each feature is critical to understanding the basic functionality of the system and its role in meeting user needs. The tests aim to verify that these features work as expected, are reliable, and meet user requirements.

The main features that will be addressed primarily during the testing process and the basic functions of these features are listed below:

2.1. IoT Sensor Data Collection and Processing

It covers the system's ability to collect real-time data from various IoT sensors securely transmit this data to the central system and convert it into meaningful information.

Scope of Test:

- Verification of correct data acquisition from different sensor types.
- Control of reliability and timeliness of data transmission.
- Testing whether the received data is processed correctly and converted into meaningful alarm and risk levels.
- Examination of how the system behaves in the event of sensor failures or disconnections.

2.2. Security Verification Notifications and Response Mechanism

This includes the automatic sending of "Are you safe?" notifications to users in the disaster area when a danger is detected by the system and the mechanism for users to respond with "I am safe" or "I am not safe".

Scope of Test:

- Ensuring that notifications reach the correct users on time when a danger is detected.
- The mobile application displays notifications correctly and records user interaction (response) correctly.
- User responses ("I am safe", "I am not safe") are correctly transmitted and processed to the system.
- Verification that the system initiates the correct actions (informing emergency contacts and units) in case no response is received within a certain period of time.

2.3. Emergency Contact Notification

This includes the function of the system automatically informing predefined emergency contacts in case the user responds with "I am not safe" or does not respond within the specified period of time.

Scope of Test:

- Checking that emergency contact information is stored correctly in the system and accessible when needed.
- Checking that notification messages are delivered to the correct contact people on time.
- Checking that the content of notification messages is correct.

2.4. Notification to Emergency Units

Included is the function of the system to automatically notify the relevant official emergency units if the user responds with "I am not safe" or does not respond within the specified time.

Scope of Test:

- Checking that the contact information of emergency units is configured correctly in the system.
- Checking that notifications are delivered to the correct units on time.
- Checking that the content of notifications (location information, type of danger, etc.) is correct.

2.5. Mobile Application User Management

Covers basic mobile application functions such as users creating and updating profiles, adding/editing emergency contacts, and receiving security notifications.

Scope of Test:

- Verifying that user registration and login processes work smoothly.
- Testing that profile information is recorded and updated correctly.
- Ensuring that emergency contacts can be easily added, edited, and deleted.
- Testing that security notifications are displayed correctly on the mobile application and user interaction (response).

2.6. Web-Based Management Panel

It covers the web-based interface that allows emergency teams to monitor the security status of users in the disaster area in real time, view risk maps and manage incident response processes.

Test Scope:

- Verification of secure access to the management panel.
- Testing the correct and real-time display of user security statuses on the map.
- Correct display and interpretation of risk maps.
- Testing the functionality of incident management tools.

2.7. Communication Infrastructure and Durability

It covers the reliable and uninterrupted communication infrastructure between all components in the system.

Test Scope:

- Testing the continuity and reliability of communication under different network conditions.
- Controlling whether data transfer between system components is correct and timely.
- Examining the system's resilience (load testing) against possible communication congestion that may occur during disasters.

2.8. Positioning and Map Integration

It covers the ability to accurately determine the location information of users and events in the system and visualize it on the map. This is critical for locating disaster victims, identifying risky areas, and directing emergency teams.

Test Scope:

- Verifying accurate location data acquisition from mobile devices.
- Testing the reliability and precision of location data in different environments.
- Checking the correct integration and visualization of location data into map services.
- Ensuring the confidentiality and secure processing of location information.
- Testing the correct and timely sharing of location information with emergency units.
- Examination of how the system handles possible errors in positioning technologies.

2.9. Alarm and Warning Management

It covers the function of automatically triggering alarm and warning mechanisms upon dangers detected by the system or requests for assistance sent by the user and sending warnings to relevant people via different channels.

Test Scope:

- Verifying the correct operation of alarm triggering mechanisms for different danger scenarios and threshold values.
- Checking that the content of warning messages is correct and informative according to the type and location of the danger.
- Checking that warnings are delivered timely and reliably via different communication channels (mobile notification, SMS, voice alarm).
- Checking that warning prioritization mechanisms operate correctly.
- Testing whether users can configure their alert preferences.
- Examining how false alarms are handled by the system and how feedback is provided to the user.
- Ensuring that alarm and warning logs are recorded correctly and can be viewed from the management panel.

3. Test Methodology

Different testing approaches to be applied in order to ensure the quality and reliability level of the developed disaster management system are discussed in this section. These approaches aim to evaluate the various features of the system from a holistic perspective and to confirm whether it complies with the defined specifications and whether it shows the targeted performance.

3.1. Unit Testing

Unit testing, which is implemented in the early stages of the software development cycle, while coding is being done or immediately after, is a technique that is critical to the robustness of the application. This approach targets the most basic building blocks of the application that make sense and function on their own, namely code blocks such as functions, methods or classes. It is essential that these components are considered and checked individually and independently of each other; in other words, when a function is being tested, the status of other functions or classes it depends on should not affect this test. This early intervention allows possible errors to be detected and fixed at the beginning of the development process, when the cost of fixing them is lower and the impact is limited. In this way, a solid foundation is created by ensuring that each part works correctly in isolation before moving on to more complex integration stages.

3.2. Integration Testing

Integration testing is the testing of interoperability and correctness of data flow between different software modules or components that interact with each other by bringing them together. These tests are performed after the unit tests are successfully completed and aim to detect potential problems at the integration points of different units. In the disaster management system project, integration scenarios between different components such as correctly transmitting data from IoT sensors to the back-end system, transmitting security information received from the user of the mobile application to the server and displaying it on the web panel will be tested.

3.3. System Testing

System testing is a testing phase that holistically evaluates the compliance of the system formed by all integrated components with the specified functional and non-functional requirements (performance, security, usability, etc.). These tests are designed to understand the general behavior of the system and its reactions in different scenarios. Within the scope of the project, end-to-end scenarios such as reaching users with security notifications triggered by the start of

data flow from sensors in the event of a disaster, and sending notifications to emergency units and contacts according to user responses will be covered within the scope of system testing.

3.4. Performance Testing

Performance testing aims to measure and evaluate how the system performs under a certain load. These tests are performed to determine the response time, stability, scalability and resource usage of the system. In critical applications such as disaster management systems, it is possible for many users to access the system simultaneously and exchange data. Therefore, performance testing is of great importance to verify that the system performs acceptable even under high user load. Within the scope of this project, the system's durability and performance will be tested by creating intense user and data traffic over simulated disaster scenarios.

3.5. Usability Testing

The purpose of usability tests is to measure how comfortably, efficiently and satisfactorily a specified user base can use the system. These tests, usually conducted with real or simulated users, focus on whether the system's interface and interactions can be easily understood by the user.

3.6. Security Testing

The main goal of testing system security is to identify potential vulnerabilities, security gaps, and resistance to threats. These tests are indispensable for preventing security risks such as unauthorized access, unauthorized modification of data, and service interruptions. Especially in systems that contain both sensitive user data and manage emergency interventions, such as disaster management, it is imperative that security tests are performed with great care and attention. The security status of the system will be evaluated thoroughly using various methods such as penetration tests, vulnerability scans, and code analysis.

3.7. User Acceptance Testing (UAT)

User Acceptance Testing is a testing phase that aims to verify whether the developed system meets the real needs and expectations of end users. These tests are usually performed by testing the system in a real or near-real environment by end users after the development process is completed. The purpose of User Acceptance Testing is to confirm the system's compliance with business requirements and its acceptability by users. Within the scope of this project, the functionality and ease of use of the system will be evaluated through scenarios determined by target user groups (e.g. AFAD personnel, volunteers).

3.8. Compatibility Testing

Compatibility testing aims to evaluate whether the system works as expected in different hardware, software, operating system and network environments. The disaster management system must work smoothly on different mobile devices (Android, iOS), different web browsers and different network connection speeds. The performance and functionality of the system in these different environments will be verified with compatibility tests.

3.9. Sanity Testing

Sanity testing is a focused type of testing that is done after a bug is fixed or a change is made to the system to check if the fix or change works as expected and does not negatively affect other parts of the system. The aim is to make sure that the changes made do not break the system further. Within the scope of this project, after any bug is fixed, the relevant functionality and other areas associated with it will be quickly verified with sanity tests.

4. Test Environment

The test environment provides an emulation of the real operating environment under which the Post-Disaster Communication System will be deployed. A stable and controlled test environment is necessary to facilitate analysis of system behavior under a range of use-case scenarios, environment conditions, and network conditions.

4.1. Hardware Environment

- **IoT Sensor Simulators:**
  - Water level, temperature, and seismic sensor simulations were used to simulate disaster scenarios.
  - Edge-level signal generating dummy devices based on Arduino/Raspberry Pi.
- **Mobile Devices:**
  - Android Phones
  - iPhones
- **Desktop Machines (for Web App Testing):**
  - Windows 10/11 and macOS Ventura environments
  - Browsers: Chrome, Firefox, Safari, and Edge (latest 2 versions)
- **Server Infrastructure:**
  - Firebase Cloud Firestore (NoSQL database)
  - PostgreSQL instance hosted via Heroku for structured data logging
  - Node.js backend hosted via Firebase Functions

4.2. Software Environment

- **Mobile Application Stack:**
  - Flutter for cross-platform development
  - Firebase SDK for real-time database operations and authentication
  - Android Studio and Xcode for emulation and deployment
- **Web Application Stack:**
  - React.js frontend
  - Node.js backend with Express.js
  - RESTful API services for mobile/web integration

- **Testing Tools:**
  - **Postman** – API validation and load testing
  - **Selenium WebDriver** – Automated UI testing for the Web Portal
  - **Jest & Mocha** – Unit and integration testing of components
  - **JMeter** – Load testing of server endpoints
  - **OWASP** – Web application penetration testing
  - **Firebase Emulator Suite** – Local test environment for real-time data sync

4.3. Network Conditions

- **Real-World Scenarios Simulated:**
  - Signal strength ranging from weak to strong for 4G/5G mobile data
  - Simulation of a high-traffic network (latency injection, packet loss)

4.4. Environmental Configurations

- **Deployment:**
  - CI/CD with GitLab CI for build-test-deploy cycle
  - Docker containers used for isolating test server and mock services
- **User Types Simulated:**
  - End Users (Citizens)
  - Emergency Personnel
  - System Administrators

5. Test Schedule

Test Phase	Responsible Person(s)	Start Date	End Date	Description
Unit Testing	Burak Güçlü, İrem Özyurt	15.04.2025	20.04.2025	Ensure correctness of individual components such as backend functions, Flutter modules, and sensor data handlers.
Integration Testing	Burak Güçlü, Gizem Yüksel	21.04.2025	25.04.2025	Verify data flow and interaction between modules like IoT sensor → backend → mobile/web application.
System Testing	Gizem Yüksel	26.04.2025	30.04.2025	Run end-to-end disaster scenarios including emergency alert and response cascade.
Performance Testing	Burak Güçlü	01.05.2025	04.05.2025	Simulate concurrent user loads and evaluate system under stress to ensure high availability.
Usability Testing	İrem Özyurt, Zeynep Sude Bal	02.05.2025	06.05.2025	Assess ease-of-use, clarity of UI, and efficiency of workflows under user stress.
Security Testing	İrem Özyurt	05.05.2025	10.05.2025	Conduct penetration testing, session hijacking, data protection, and OWASP guideline checks.
UAT	AFAD Volunteers	11.05.2025	15.05.2025	Real users (e.g., first responders) evaluate system usability and functionality in near-real settings.
Compatibility Testing	Zeynep Sude Bal	12.05.2025	14.05.2025	Test mobile/web interface on different OS (Android/iOS, macOS/Windows), browsers, and networks.
Sanity Testing	Gizem Yüksel, İrem Özyurt	Throughout	Throughout	Quick regression checks after each bug fix or module update to ensure stability.

The test schedule outlines the timeline of all planned testing activities that will be conducted throughout the lifecycle of the post-disaster communication system. Each phase has been strategically placed within the project calendar to ensure continuous verification and validation of system components. The distribution of responsibilities among team members ensures that

specific technical and functional areas are thoroughly tested by individuals with relevant expertise.

For instance, unit testing is scheduled early to capture basic functionality issues, while integration and system testing follow to validate interactions between modules and the system as a whole. Performance and security testing are planned in anticipation of high-stress scenarios such as natural disasters, where reliability and data protection become critical. Lastly, user acceptance testing ensures that the system is both functional and intuitive for emergency personnel and real users in disaster settings. This systematic approach allows for early bug detection, structured improvement, and optimal project progression.

6. Control Procedures

Strong control processes ensure the integrity, traceability, and consistency of all test activities conducted throughout the Post-Disaster Communication System's lifecycle. They manage how test assets are developed, tracked, updated, and validated and how issues and risks are managed. They are essential to providing a reliable and resilient system that satisfies safety-critical requirements.

6.1. Test Case Management

Test case management will be handled by version-controlled and organized documentation tools such as Google Sheets, GitLab Wiki, or dedicated software such as TestRail. Every test case is assigned a unique identifier and resides in a test case repository.

- Each test case record will include:
  - Test Case ID (e.g., TC-MOB-01)
  - Test Title
  - Test Objective / Description
  - Test Preconditions
  - Test Input Data
  - Test Steps
  - Expected Result
  - Actual Result
  - Pass/Fail Status
  - Tester Name
  - Test Date
  - Linked Requirement ID

Test cases will be reviewed and validated by the QA Lead. After review, any modifications to the test cases require change control approval.

6.2. Defect Tracking and Bug Management

All errors or inconsistencies found during testing will be recorded and monitored with traceability to test case IDs and system requirements using Jira or GitLab Issues.

- Each defect report will include:
  - Bug ID
  - Summary Title
  - Severity Level (Blocker, Critical, Major, Minor, Trivial)
  - Priority (High/Medium/Low)
  - Reproducibility Steps
  - Affected Component
  - Expected vs. Actual Behavior
  - Environment (Device, OS, Browser, Network condition)
  - Status (Open, In Progress, In Review, Resolved, Verified, Closed)
  - Assignee
  - Attachments (screenshots/logs)

Meetings for bug triage will take place every two to three days during periods of high testing and once a week otherwise. Issue aging reports will be examined and forwarded to the course advisor if significant flaws are not fixed after the predetermined amount of time.

6.3. Version and Configuration Management

All system components are subject to the following versioning rules to guarantee that testing is conducted against consistent and controlled builds:

- A Build ID will be given to every build.
- Each environment (dev, test, staging) will be separately configured and tagged.
- Version control will be handled via Git; every release needs to be approved and marked in the repository.
- Test datasets, environment variables, and configuration files are kept in versioned, safe, and documented folders.

There will be a Baseline Environment Configuration Document that contains the following:

- OS versions and device specifications
- Versions of the database and backend
- Versions of API endpoints
- Firebase emulator state (for offline testing)

6.4. Change Management Procedure

A systematic change management process will manage any system functionality, infrastructure, or user-facing behavior modification after baseline approval.

Steps in the change control workflow:

- Submission of a change request (by a stakeholder, developer, or QA)
- Analysis of Impact (by testing and development teams)
- Decision on Approval/Deferment (by project lead)
- Update to Applicable Artifacts (codebase, requirements, and test cases)
- Testing for regression in every module that could be impacted
- Change documentation in the project's test and change logs

Every modification will be recorded in a Change Log Register along with the timestamp, accountable parties, modules impacted, and post-change testing status.

6.5. Test Exit Criteria

The following exit criteria will be used to determine that the testing process has been successfully completed and the system is ready for release:

All Critical Test Case Passes: All high-priority test cases covering the core functionality of the system must be 100% successfully passed.

Acceptable Level of Remaining Bugs: None of the remaining open bugs must be at the “Blocker” or “Critical” level; only ‘Minor’ or “Trivial” bugs that do not seriously impact the user experience will be considered acceptable.

Comprehensive Regression Tests: Regression tests must confirm that the new changes do not disrupt the existing system operation and that all modules work as before.

Meeting Performance Criteria: The system must meet specified performance thresholds (e.g. response time < 2 seconds, 95% uptime).

Passing Security Tests: Security tests, including compliance with OWASP guidance, should result in no vulnerabilities or exploits.

Completion of Documentation: Test cases, case results, bug reports and change logs must be fully documented.

6.6. Communication and Reporting

Transparent and continuous communication throughout the testing process is critical to the success of the project. The communication and reporting procedures to be implemented in this context are as follows:

Weekly Status Reports: Weekly reports will be generated by the test leader and will include the following information:

- Types of tests performed
- Test cases passed/failed
- Number of bugs found and their severity

Live Tracking via GitLab/Jira: All test progress, bug logs and test cases will be tracked instantly by the team via GitLab Issues or Jira. Prioritization will be determined with tagging and filtering systems.

Triage Meetings: These meetings, which will be held twice a week in case of critical defect excess, and weekly in normal periods, will ensure that defects are prioritized and assigned to responsible people.

Final Test Report: At the end of the testing process, a comprehensive final report will be prepared by the QA team including the overall quality assessment of the system. This report will include test results, success rates, remaining risks and recommendations.

7. Roles and Responsibilities

Role	Responsibilities	Assigned Person(s)
QA Lead	Coordinates all QA activities, manages test cases, reviews reports, and ensures compliance with quality standards.	Gizem Yüksel, Burak Güçlü, İrem Özyurt, Zeynep Sude Bal
Mobile App Tester	Executes tests on Flutter mobile app including UI/UX feedback, notification mechanisms, and error handling.	İrem Özyurt
Web Panel Tester	Ensures stability and functionality of the admin panel, real-time map updates, and alert display.	Zeynep Sude Bal
Backend Tester	Validates all REST API endpoints, database logging, sensor integration and backend performance.	Gizem Yüksel
IoT Simulation Engineer	Creates disaster scenarios via dummy sensors, manages edge devices (e.g. Arduino/Pi).	Burak Güçlü
Developer Support	Assists test team by debugging, updating code, and implementing urgent fixes.	All Developers
UAT Coordinator	Organizes user testing sessions with AFAD and processes feedback into actionable tasks.	QA + AFAD Representative
Academic Advisor	Provides guidance on test strategy alignment with research goals and ensures methodological rigor.	Dr. Emin Kuğu

This section delineates the individual roles and responsibilities within the testing process, promoting accountability and transparency. By assigning specific tasks to designated members of the development and QA team, the report ensures that all dimensions of the system—including mobile interface, web panel, sensor integration, and backend logic—are covered comprehensively.

The allocation of duties aligns with team members' skill sets, ensuring effective utilization of resources. For example, mobile and web testers focus on front-end behavior, usability, and responsiveness, while backend responsibilities are assigned to those proficient in server logic and API validation. Furthermore, the involvement of an academic advisor helps maintain methodological rigor and objectivity. The structure also facilitates better collaboration, issue tracking, and streamlined decision-making during the testing process.

8. Risks

Risk	Potential Impact	Mitigation / Solution
Lack of access to physical sensors	Delays in hardware-specific bug discovery; unreliable simulation fidelity	Use Raspberry Pi/Arduino with calibrated sensor mocks and edge scripts.
Device/browser fragmentation	Inconsistent behavior across platforms may affect emergency delivery	Target top 80% market share devices for both Android/iOS and cross-browser.
Overload during disaster simulation	Crash or delays under simultaneous access	Benchmark load thresholds; apply horizontal scaling via Firebase Functions.
Late detection of critical bugs	Failure in emergency flow could endanger user trust	Adopt CI/CD with auto-regression test suites for each build.
Unstructured UAT feedback	Inapplicable or missed usability insights	Provide structured feedback forms and guidance to users before testing.
Security misconfigurations	System compromise or user data breach	Conduct end-to-end OWASP checklist and implement role-based access control (RBAC).
Inefficient team communication	Misunderstood bug reports, duplicate efforts	Centralize test tracking via GitLab Issues and triage meetings.

In critical systems such as post-disaster communication platforms, identifying and managing risks is a fundamental part of the testing strategy. The risk matrix presented in this section serves not only as a proactive diagnostic tool but also as a strategic safeguard to ensure system resilience under adverse conditions. Given the life-saving nature of this project, any disruption in functionality, accessibility, or data integrity could have serious consequences. Therefore, an early and systematic evaluation of potential risks helps the team to mitigate failures before they manifest in real disaster scenarios.

The risks identified encompass a wide range of technical and operational concerns. For example, access to real IoT sensor hardware may be limited due to logistical or budget constraints, which could impact the accuracy of real-world simulation tests. Similarly, the diversity of mobile devices and operating systems presents a risk of inconsistent user experiences or software failures across platforms. These issues are especially problematic in emergency scenarios where immediate and reliable user feedback is crucial.

Another critical concern involves system overload due to simultaneous access by thousands of users during a disaster. This scenario could lead to crashes or severe delays, hampering emergency response. The risk table anticipates this by recommending scalability testing and performance optimization. Moreover, late detection of bugs—especially those related to safety-critical functions like alert notifications and location sharing—can undermine the reliability of the system. Thus, regression and sanity tests are scheduled throughout the development lifecycle to ensure stability.

Security risks are also highlighted as a high-priority concern. Unauthorized access, data leaks, or tampering with alert mechanisms could not only violate user privacy but also compromise public safety. To address this, the team will conduct security testing using industry standards like the OWASP framework and implement robust access control protocols.

Organizational risks, such as poor communication between developers and testers or delayed test reporting, are likewise acknowledged. These challenges can lead to inefficiencies, duplication of work, or unaddressed vulnerabilities. Regular triage meetings, centralized issue tracking systems (e.g., GitLab or Jira), and automated reporting pipelines are suggested as mitigation methods.

Overall, this risk management strategy ensures that the team is well-prepared to handle both expected and unforeseen complications. It provides a safety net not just for the development process, but also for the end users who will depend on the system during emergencies. This proactive approach enhances the reliability, trust, and operational continuity of the platform, which are indispensable in disaster-response scenarios.

9. Test Scenarios and Test Cases

Scenario #1: IoT Sensor Data Flow and Alarm Generation

Verify that real-time data from various IoT sensors (earthquake, water level, temperature) is correctly received by the system, securely transmitted to the central system, and transformed into meaningful information (alarms and risk levels). The test will determine whether the system automatically generates an alarm when sensor data exceeds predefined threshold values.

Scenario #2: Sending "Are You Safe?" Notifications and Processing User Responses

Verify that "Are You Safe?" notifications are sent to users in the disaster zone when a danger is detected, and that user responses ("I am Safe" or "I am Not Safe") are correctly received and processed by the system. This test will check if the user's response mechanism works correctly and how the system behaves when no response is received within a specified time.

Scenario #3: Sending Notifications to Emergency Contacts

Verify that the system sends accurate and timely notifications to the user’s predefined emergency contacts when the user responds "I am Not Safe" or does not respond within a specified time. The test will verify whether the notification content (including location information) is correct and whether the notifications reach the emergency contacts.

Scenario #4: Sending Notifications to Emergency Units

Verify that the system sends accurate and timely notifications to relevant official emergency units (e.g., AFAD, fire departments, health teams) when the user responds "I am Not Safe" or does not respond within a specified time. The test will check if the notification content (including location information, danger type, etc.) is correct and complete and whether the notifications reach the emergency units.

Scenario #5: Mobile Application User Registration and Login Operations

Verify that users can register and log in smoothly via the mobile application. The test will check whether user information (name, email, password, etc.) is correctly processed and whether users can access the system.

Scenario #6: Mobile Application Profile Management

Verify that users can correctly save, view, and update their personal profile information (name, surname, contact details, etc.) through the mobile application. The test will check if the profile information is correctly stored in the database and displayed correctly in the user interface.

Scenario #7: Mobile Application Emergency Contact Management

Verify that users can easily add, update, and delete emergency contacts through the mobile application. The test will check if emergency contact information is correctly stored and whether the user interface is user-friendly.

Scenario #8: Secure Access to Web Management Panel

Verify that authorized emergency teams can access the web management panel through a secure authentication mechanism. The test will check if access control for different user roles works correctly and if unauthorized access attempts are blocked.

Scenario #9: Displaying User Security Status on Web Management Panel

Verify that user security statuses (Safe, Not Safe, No Response) in the disaster area are displayed in real-time on the map in the web management panel. The test will check whether user location and security statuses are updated correctly and timely on the map.

Scenario #10: Displaying Risk Maps and Event Management in Web Management Panel

Verify that risk maps generated from IoT sensor data are displayed correctly on the web management panel and that emergency teams can use event management tools (e.g., directing response teams). The test will evaluate the accuracy of risk maps, clarity, and functionality of event management tools.

Scenario #11: Communication Resilience in Different Network Conditions

Verify that the system can maintain reliable and uninterrupted communication between all components under various network conditions (strong signal, weak signal, high traffic, delay, packet loss, etc.). The test will simulate conditions such as data loss, delay, and connection drops to observe how communication performance is affected.

Scenario #12: Accurate Location Detection and Map Integration

Verify that the location information obtained from mobile devices is accurately and reliably detected and integrated into map services to be visualized correctly in both the mobile application and web management panel. The test will check location accuracy in different environments (open space, indoor, etc.) and whether location information is securely processed.

Scenario #13: Alarm and Warning Mechanism Functionality

Verify that alarm and warning mechanisms are automatically triggered based on detected hazards or user-submitted help requests, and that warnings are sent to relevant individuals through various communication channels (mobile notification, SMS, audible alarm). The test will verify whether alarm thresholds are correctly set for different hazard scenarios, if warning messages are appropriate, and if warnings are delivered on time.

The table starting from next page will be showing different test cases for the scenarios mentioned above.

Table 1 – Test Cases

Test Case ID	Test Title	Test Objective	Preconditions	Test Data Input	Test Steps	Expected Result	Related Requirement ID
TC-IoT-001-001	Verification of data acquisition below the earthquake threshold	To verify that the system receives data from the earthquake sensor at regular intervals and at normal levels	- IoT sensor simulator must be running - Earthquake sensor should send data below threshold - System must be active	Seismic activity at Richter level 2.0	1. Ensure the simulator is sending data 2. Monitor sensor data on the web panel 3. Observe data for 5 minutes	- Data should be received and displayed regularly - Data should match simulator values - No alarm should be generated	FR-IoT-001
TC-IoT-001-002	Verification of alarm generation for data above the earthquake threshold	To ensure the system automatically generates an alarm when high-level data is received from the earthquake sensor	- IoT sensor simulator must be running - Earthquake sensor should send data at alarm level (Richter 6.0) - System must be active	Seismic data at alarm level	1. Start sending data from simulator 2. Monitor alarm status on web panel 3. Check for mobile notifications	- High-level data should be visible on web panel	FR-IoT-001, FR-ALM-001
TC-IoT-001-003	Verification of flood alarm for data above critical water level	To ensure the system generates a flood alarm when critical water level data is received from the sensor	- IoT sensor simulator must be running - Water level sensor should send data above the critical threshold - System must be active	Critical water level data	1. Start sending critical level data from simulator 2. Monitor web panel for water level 3. Check if flood alarm is generated and notification is sent	- Critical water level should be displayed - Flood alarm should be triggered - Notification should be sent to related parties	FR-IoT-001, FR-ALM-002
TC-NOT-002-001	Notification Delivery upon Alarm Trigger	To confirm that the system sends a timely "Are you safe?" notification to users when a hazard (e.g., simulated earthquake) is detected.	- System is operational - Users have the mobile app installed and are logged in - A hazard has been triggered	Simulated earthquake alarm	1. Trigger an earthquake alarm via simulator 2. Check mobile devices of affected users	Users should receive an "Are you safe?" notification with a timestamp matching the alarm trigger time	FR-NOT-001

TC-NOT-002-002	Processing of "I'm Safe" Response	To verify that when a user responds with "I'm safe," the response is accurately recorded in the system and reflected in the web admin panel.	- System is operational - User has received the notification	User selects "I'm safe"	1. Confirm user submits "I'm safe" response 2. Open the web admin panel	The user's safety status is updated to "Safe" in the admin panel, along with the correct timestamp	FR-NOT-002
TC-NOT-002-003	Processing of "I'm Not Safe" Response & Emergency Notification Trigger	To confirm that when a user responds with "I'm not safe," the system updates the safety status and automatically notifies the user's emergency contacts and relevant authorities.	- System is operational - User has received the notification - User selects "I'm not safe" - Emergency contacts and authorities are registered in the system	User selects "I'm not safe"	1. Confirm user submits "I'm not safe" response 2. Open the web admin panel 3. Navigate to user safety status view 4. Verify messages are sent to emergency contacts and authorities via registered channels (e.g., email, SMS)	The user's status is updated to "Not Safe" in the system, and emergency contacts and authorities receive a notification including the user's location and timestamp	FR-NOT-002, FR-EMC-001, FR-OFF-001
TC-EMC-003-001	Notification to Emergency Contact after "Not Safe" Response	To verify that when a user selects "Not Safe" in response to the "Are you safe?" prompt, the system automatically sends a notification (e.g., SMS) to the user's registered emergency contact.	- The system is operational - "Are you safe?" prompt has been sent to the user - The user selects "Not Safe" via the mobile app - At least one emergency contact is registered (e.g., phone number)	User selects "Not Safe"	1. Confirm that the user selects "Not Safe" in the mobile app 2. Check the emergency contact's communication channel (e.g., SMS inbox)	A notification containing the "Not Safe" status and the user's location is sent to the emergency contact. The timestamp of the notification is correctly recorded.	FR-EMC-001
TC-EMC-003-002	Notification to Emergency Contact after No Response	To verify that if the user does not respond to the "Are you safe?" prompt within a predefined time (e.g., 60 seconds), the system automatically sends a notification to the registered emergency contact.	- The system is operational - "Are you safe?" prompt has been sent - The user does not respond within the predefined time	No response from the user	1. Confirm that the user does not respond 2. Wait for the predefined time 3. Check the emergency contact's	A notification indicating the user did not respond, including the user's location, is sent to the emergency contact. The timestamp of the notification is correctly recorded.	FR-EMC-002

			- At least one emergency contact is registered		communication channel		
TC-EMC-003-003	Notification Sent to Multiple Emergency Contacts	To verify that if the user has more than one emergency contact, the system sends notifications to all contacts in case of a “Not Safe” response or no response.	- The system is operational - “Are you safe?” prompt has been sent	“Not Safe” response or no response from the user	1. Confirm that the user responds “Not Safe” or the predefined time has passed without a response 2. Check all emergency contacts’ communication channels	All registered emergency contacts receive a notification containing the user’s status and location. All notification timestamps are correctly recorded.	FR-EMC-001
TC-OFF-004-001	Notification Sent to Emergency Unit After “Not Safe” Response	When the user responds “Not Safe” to the "Are you safe?" prompt, the system must automatically notify the predefined emergency unit (e.g., via API call to AFAD).	- System must be operational. - “Are you safe?” prompt must have been sent. - User selects “Not Safe” in the mobile app. - Emergency contact details (e.g., API endpoint) must be properly configured.	User response: "Not Safe"	1. Ensure the user selects "Not Safe" in the mobile app. 2. Open the web admin panel and check system logs for notification dispatch. 3. If possible, verify the content and receipt of the notification using a mock API endpoint.	- A log entry should confirm that the notification was sent. - If using a simulated receiver, the notification content (user ID, location, danger type) must be correct. - Timestamp of the notification must be correctly recorded.	FR-OFF-001
TC-OFF-004-002	Notification Sent After No Response	If the user does not respond to the “Are you safe?” prompt within a predefined time, the system must automatically notify the relevant emergency unit.	- System must be operational. - “Are you safe?” prompt must have been sent. - The user fails to respond within the defined time. - Emergency unit contact details must be correctly configured.	No response from user	1. Ensure no response from the user and wait for timeout. 2. Open the web admin panel and check system logs for notification dispatch. 3. If possible, verify the content and receipt of the	- A log entry should confirm that the notification was sent. - If using a simulated receiver, the notification content must be correct. - Timestamp of the notification must be correctly recorded.	FR-OFF-001

					notification using a mock API endpoint.		
TC-OFF-004-003	Correct Notification Based on Hazard Type	The system must send the notification to the correct emergency unit based on the detected hazard type (e.g., earthquake → AFAD, flood → fire brigade).	- System must be operational. - IoT hazard simulators must be active. - Emergency units' contact information must be correctly set for each hazard. - Simulated earthquake and flood alerts must be triggered.	Simulated earthquake and flood alerts	1. Trigger an earthquake alert in the system. 2. Open the web admin panel and check system logs for which unit was notified. 3. Trigger a flood alert in the system. 4. Check the system logs again. 5. If possible, verify notifications were sent to the correct units using mock endpoints.	- Earthquake alert should result in a notification to AFAD (or configured unit). - Flood alert should result in a notification to the fire brigade (or configured unit). - Notification contents must be accurate.	FR-OFF-002
TC-MOB-005-001	Successful User Registration with Valid Information	Ensure users can successfully register with valid email, password, etc.	Mobile app should be installed on the device. User should have a valid email address and a strong password. Device must have an internet connection.	Valid email address and password.	1. Open the mobile app. 2. Click on "Register". 3. Enter valid email address. 4. Enter valid password and confirm it. 5. Accept terms and privacy policy (if applicable). 6. Click "Register".	User should see a message confirming successful registration. User should be logged in automatically or redirected to the login screen. User data (email, etc.) should be stored correctly.	FR-UMG-001
TC-MOB-005-002	Registration Attempt with Invalid Email Format	Verify that the system shows an appropriate error message when an invalid email is entered.	Mobile app should be installed on the device. Device must have an internet connection.	Invalid email address (e.g., "user", "user@", "user@mail").	1. Open the mobile app. 2. Click on "Register". 3. Enter an invalid email address.	The system should display an error message like "Please enter a valid email address". Registration should not be completed.	FR-UMG-001

					4. Enter valid password and confirm it. 5. Click "Register".		
TC-MOB-005-003	Registration Attempt with Already Registered Email	Verify that the system shows an appropriate error message when a registered email is used.	Mobile app should be installed on the device. User must have an existing account with the email. Device must have an internet connection.	Registered email address and valid password.	1. Open the mobile app. 2. Click on "Register". 3. Enter an already registered email address. 4. Enter valid password and confirm it. 5. Click "Register".	The system should display an error message like "This email address is already registered". Registration should not be completed.	FR-UMG-001
TC-MOB-005-004	Successful Login with Valid Credentials	Ensure that registered users can log in with the correct email and password.	Mobile app should be installed on the device. User should have a valid account. Device must have an internet connection.	Registered email address and password.	1. Open the mobile app. 2. Click on "Login". 3. Enter registered email address. 4. Enter correct password. 5. Click "Login".	User should successfully log in and be redirected to the main screen.	FR-UMG-002
TC-MOB-005-005	Login Attempt with Invalid Password	Verify that the system shows an appropriate error message when an incorrect password is entered.	Mobile app should be installed on the device. User should have a valid account. Device must have an internet connection.	Registered email address and incorrect password.	1. Open the mobile app. 2. Click on "Login". 3. Enter registered email address. 4. Enter incorrect password. 5. Click "Login".	The system should display an error message like "Incorrect email or password". Login should fail.	FR-UMG-002
TC-MOB-006-001	Successful Display of Profile Information	Verifying that the user's profile information (entered during registration or updated previously) is displayed correctly in the mobile app.	The mobile app must be installed on the user's device. The user must be logged in. The user must have saved profile information (e.g., first name, last name).	None (Existing profile information)	1. Open the mobile app and log in. 2. Navigate to Profile or Account Settings. 3. Find the section displaying personal	The user's previously saved or updated profile information should be displayed correctly.	FR-UMG-003

					information (name, email, etc.).		
TC-MOB-006-002	Successful Update of Profile Information	Verifying that users can update their profile information (e.g., phone number) via the mobile application.	The mobile app must be installed on the user's device. The user must be logged in. The user must be able to access the profile update section.	A new phone number	1. Open the mobile app and log in. 2. Navigate to Profile or Account Settings. 3. Click on the option to edit profile information. 4. Enter a new phone number. 5. Click Save or Update.	The user should be shown a message confirming successful update. The new phone number should be displayed correctly.	FR-UMG-004
TC-MOB-006-003	Attempt to Update Profile Information with Invalid Formats	Verifying that the system shows an error message when the user attempts to update profile information with invalid formats (e.g., incorrect phone number).	The mobile app must be installed on the user's device. The user must be logged in. The user must be able to access the profile update section.	Invalid phone number format (e.g., letters or a short number)	1. Open the mobile app and log in. 2. Navigate to Profile or Account Settings. 3. Click on the option to edit profile information. 4. Enter an invalid phone number. 5. Click Save or Update.	The system should display an error message such as "Please enter a valid phone number." The profile should not be updated.	FR-UMG-004
TC-MOB-007-001	Successful Emergency Contact Addition	Verify that the user can successfully add an emergency contact with valid details (name, surname, phone number).	The mobile application must be installed, the user must be logged in, and the user must have access to the emergency contacts management section.	Valid name, surname, and phone number	1. Open the mobile application and log in. 2. Go to the Profile or Account Settings section. 3. Find and click the Emergency Contacts management option.	The user should receive a message confirming the successful addition of the emergency contact. The contact should appear in the emergency contacts list with the correct details.	FR-EMC-002

					4. Click the "Add New Emergency Contact" button. 5. Enter the name, surname, and phone number. 6. Click the "Save" button.		
TC-MOB-007-002	Attempt to Add Emergency Contact with Invalid Phone Number	Verify that the system displays an appropriate error message when the user tries to add an emergency contact with an invalid phone number.	The mobile application must be installed, the user must be logged in, and the user must have access to the emergency contacts management section.	Invalid phone number (missing digits or letters)	1. Open the mobile application and log in. 2. Go to the Profile or Account Settings section. 3. Find and click the Emergency Contacts management option. 4. Attempt to add an emergency contact with an invalid phone number. 5. Click the "Save" button.	The system should display an error message like "Please enter a valid phone number". The contact should not be added.	FR-EMC-002
TC-MOB-007-003	Successful Emergency Contact Edit	Verify that the user can successfully edit the details (e.g., phone number) of an existing emergency contact.	The mobile application must be installed, the user must be logged in, and there should be at least one emergency contact to edit.	New phone number for an existing emergency contact	1. Open the mobile application and log in. 2. Go to the Profile or Account Settings section. 3. Find and click the Emergency Contacts management option. 4. Find the contact to be edited and	The user should receive a message confirming the successful update of the emergency contact's details. The new phone number should be displayed correctly in the contacts list.	FR-EMC-003

					click the "Edit" option. 5. Enter the new phone number. 6. Click the "Save" button.		
TC-MOB-007-004	Successful Emergency Contact Deletion	Verify that the user can successfully delete an existing emergency contact from the mobile application.	The mobile application must be installed, the user must be logged in, and there should be at least one emergency contact to delete.	Selected emergency contact to be deleted	1. Open the mobile application and log in. 2. Go to the Profile or Account Settings section. 3. Find and click the Emergency Contacts management option. 4. Find the contact to be deleted and click the "Delete" option (confirm if necessary).	The user should receive a message confirming the successful deletion of the emergency contact. The contact should no longer appear in the emergency contacts list.	FR-EMC-004
TC-WEB-008-001	Successful Login with Valid Username and Password	Verify that an authorized user can successfully log into the web management panel with valid credentials.	The web management panel must be accessible (URL known). A test user account with valid credentials must be created. A web browser (e.g., Chrome, Firefox) must be installed in the test environment.	Valid username and password	1. Open a web browser and enter the web management panel URL. 2. Enter the valid username in the username field. 3. Enter the valid password in the password field. 4. Click the login button.	The user should successfully log into the web management panel and see the main control or dashboard.	FR-WAP-001
TC-WEB-008-002	Login Attempt with Invalid Username	Verify that the system shows an appropriate error message when trying to log into the web	The web management panel must be accessible.	Invalid username and any password	1. Open a web browser and enter the web	The system should display an error message like "Invalid username or	FR-WAP-001

		management panel with an invalid username.			management panel URL. 2. Enter an invalid username in the username field. 3. Enter any password in the password field. 4. Click the login button.	password". The login attempt should fail.	
TC-WEB-008-003	Login Attempt with Invalid Password	Verify that the system shows an appropriate error message when trying to log into the web management panel with a valid username and invalid password.	The web management panel must be accessible. The valid username must be known.	Valid username and invalid password	1. Open a web browser and enter the web management panel URL. 2. Enter the valid username in the username field. 3. Enter an invalid password in the password field. 4. Click the login button.	The system should display an error message like "Invalid username or password". The login attempt should fail.	FR-WAP-001
TC-WEB-008-004	Automatic Logout After Session Timeout	Verify that the system automatically logs out the user after a certain period of inactivity.	The web management panel must be accessed with valid credentials. A session timeout period (e.g., 15 minutes) must be set.	N/A	1. Log into the web management panel with valid credentials. 2. Remain inactive for the specified session timeout period. 3. After the timeout, click any button or refresh the page.	The user should be automatically logged out and redirected to the login page.	FR-WAP-002
TC-WEB-009-001	Users' Locations on the Map	Verify that active users' locations in the disaster area are correctly marked on the map in the web management panel.	The authorized user must be logged into the web management panel. At least one	None (Existing user location data will be used).	1. Go to the map viewing section in the web	Active users' locations in the disaster area should be displayed on the map with distinct markers	FR-WAP-003 (e.g., Displaying user locations

			active user in the disaster area must have location information.		management panel. 2. Display the disaster area on the map.	(e.g., a dot or icon). The marker locations should match the actual locations of the users.	on the map requirement)
TC-WEB-009-002	User Security Status Indicated by Colors on the Map	Verify that users' security statuses ("Safe", "Not Safe", "No Response") are indicated with different colors on the map.	The authorized user must be logged into the web management panel. There must be at least three users with different security statuses (Safe, Not Safe, No Response) in the disaster area.	Users with different security statuses.	1. Go to the map viewing section in the web management panel. 2. Display the disaster area on the map. 3. Observe the colors of the user markers on the map.	Users with "Safe" status should be marked with one color (e.g., green). Users with "Not Safe" status should be marked with a different color (e.g., red). Users with "No Response" status should be marked with another color (e.g., yellow or gray). The colors should be consistent with the system's definitions.	FR-WAP-004 (e.g., Displaying user security statuses with colors requirement)
TC-WEB-009-003	Displaying User Details When Clicking on a User Marker	Verify that the user's details (name, security status, location information, etc.) are displayed when a user marker on the map is clicked.	The authorized user must be logged into the web management panel. There must be at least one user marker on the map.	Clicking on a user marker.	1. Go to the map viewing section in the web management panel. 2. Click on any user marker on the map.	A window or panel should open displaying the user's details (name, surname, security status, last known location, response time, etc.). The information should be accurate and up-to-date.	FR-WAP-005 (e.g., Displaying user details requirement)
TC-WEB-010-001	Display of Earthquake Risk Map	Emergency response teams should be able to visually monitor earthquake risk areas via the map.	- Authorized user logged into the web management panel. - Data from earthquake sensors above a certain threshold must be available (can be simulated).	Simulated earthquake sensor data.	1. Go to the Risk Maps section on the web management panel. 2. Select the Earthquake Risk Map option.	An earthquake risk map should be displayed. Risk levels (e.g., low, medium, high) should be indicated with different colors or intensities. Risky areas should be marked at the correct locations.	FR-WAP-006
TC-WEB-010-002	Display of Water Level Risk Map	Emergency response teams should be able to visually monitor flood risk areas via the map.	- Authorized user logged into the web management panel.	Simulated water level sensor data.	1. Go to the Risk Maps section on the web	A flood risk map should be displayed. Risk levels (e.g., normal, rising, critical) should be	FR-WAP-006

			- Data from water level sensors above a certain threshold must be available (can be simulated).		management panel.	indicated with different colors or intensities. Risk-prone areas (e.g., riverbanks, lowlands) should be correctly marked.	
TC-WEB-010-003	Display of Incident Response Teams on Map	Emergency coordinators should be able to track the real-time locations of response teams.	- Authorized user logged into the web management panel. - At least one incident response team must be active and have location data in the system (can be simulated).	Simulated incident response team location data.	1. Go to the Map View section of the web panel. 2. Ensure the Incident Response Teams layer is active.	The locations of active incident response teams should be marked on the map with distinct icons. Icon locations should match the actual locations of the teams.	FR-WAP-007
TC-WEB-010-004	Sending Instructions to Incident Response Team (Simulation)	Emergency coordinators should be able to send assignment or guidance instructions to incident response teams.	- Authorized user logged into the web management panel. - At least one incident response team icon visible on the map.	Select a team and input an instruction message.	1. Go to the Map View section of the web panel. 2. Click on a response team icon on the map. 3. In the popup, click "Send Instruction" or similar. 4. Write an instruction (e.g., "Reach the user and assess the situation"). 5. Click the Send button.	System should display a confirmation that the instruction has been sent. (In simulation) The instruction should be visible on the mobile app or a simulated receiver system. The instruction and timestamp should be logged in the system.	FR-WAP-008
TC-COM-011-001	Data Transmission Under Strong Network Connection	The system should perform fast and reliable data exchange under ideal network conditions.	The mobile app must be running on the user's device with a strong Wi-Fi or mobile data connection. The system must be operational.	A “I’m Safe” response sent from the mobile app.	1. Send the "I’m Safe" response via the mobile app. 2. Open the web management panel and check the user	The user's "I'm Safe" response should be updated instantly or within a very short time on the web panel. No data loss or delay should occur.	FR-COM-001 (e.g., Reliable data transmission requirement)

					safety status section.		
TC-COM-011-002	Data Transmission and Delay Tolerance Under Weak Network Connection	The system should be able to transmit user responses and other critical data even under weak network conditions with acceptable delay tolerance.	The mobile app must be running on the user's device with a weak Wi-Fi or mobile data connection (network conditions may be simulated). The system must be operational.	A “I’m Not Safe” response sent from the mobile app.	1. Send the "I’m Not Safe" response via the mobile app. 2. Open the web management panel and check the user safety status section. 3. Record the response appearance time on the web panel.	The user's "I’m Not Safe" response should appear on the web panel within the defined maximum delay time. No data should be lost.	FR-COM-002 (e.g., Operation under weak network conditions)
TC-COM-011-003	System Performance Under High Network Traffic	The system should withstand high user activity and data traffic levels, such as during a simulated disaster event.	The system must be operational. A load testing tool capable of simulating multiple concurrent users and sensor data must be available.	Concurrent “Are you safe?” responses and sensor data from many users.	1. Use a load testing tool to simulate many users and data transmissions concurrently. 2. Monitor the web management panel's response time and overall performance.	The system should not crash or slow down excessively. The web panel should respond within an acceptable timeframe. No data should be lost.	FR-PER-001 (e.g., Performance requirements)
TC-LOC-012-001	Accurate Location Detection on Mobile App (GPS)	The app should accurately determine the user's real-world location.	- Mobile app must be installed and user must be logged in - GPS must be enabled - User should be in an open area (for better GPS signal)	None (location detection is expected)	1. Open the mobile app and navigate to the map view (if available) 2. Check the location marker or info 3. Verify the displayed location against user's actual location	The app should show the user’s location that reasonably matches the actual position	FR-LOC-001
TC-LOC-	Accurate Location Detection on Mobile App (Wi-Fi/Cellular)	The app should use alternative methods to determine location when GPS is weak or disabled.	- Mobile app must be installed and user must be logged in	None (location	1. Open the mobile app and go to the map view	The app should show a location that closely matches the user's actual	FR-LOC-002

012-002			- GPS should be turned off, but Wi-Fi or cellular data must be enabled - User may be indoors or in a weak GPS area	detection is expected)	2. Check the current location info 3. Compare the shown location with the user's actual position	location, with slightly lower accuracy than GPS	
TC-LOC-012-003	Accurate User Location Display on Web Management Panel	Emergency teams should be able to track mobile users' locations on the web panel.	- A logged-in user with appropriate access must be on the web panel - At least one mobile user with active location sharing must exist	Mobile user location data	1. Ensure the mobile user's location is active and optionally change their location 2. Go to the map section in the web panel 3. Verify the user's location on the map	The user's location should be accurately marked on the web map and match the latest mobile location data	FR-WAP-003, FR-LOC-003
TC-LOC-012-004	Disabling Location Sharing	Users should be able to stop sharing their location whenever they choose.	- Mobile app must be installed and user must be logged in - Location sharing must be enabled initially	User request to disable location sharing	1. Locate and disable the location sharing option in the app 2. Go to the web panel's map section 3. Check whether the user's location is still displayed or updated	The user's location should no longer update or be hidden on the web map (depending on system design)	FR-LOC-004
TC-ALM-013-001	Earthquake Alarm Triggering and Mobile Notification	The system should detect real-time hazards based on sensor data and send timely alerts.	- System must be operational - Mobile apps installed and logged in - Earthquake sensor simulator should send critical data	Simulated earthquake data above critical threshold	1. Send critical-level earthquake data using the sensor simulator 2. Check mobile devices of affected users	- Mobile devices should show a notification like "Earthquake Alarm" - Active earthquake alarm record should appear in the web panel - Alarm trigger time must be correct	FR-ALM-001, FR-NOT-001

					3. Check the alarm section in the web admin panel		
TC-ALM-013-002	Manual Help Request by User and Notification Sending	Users should be able to request help manually even without sensor data.	- System must be operational - Mobile app installed and logged in - Emergency contacts and units registered in system	Manual help request via mobile app	1. Open mobile app and click “Request Help” or similar button 2. Check web admin panel alarm section 3. Verify communication channels of emergency contacts and units	- A new help request record should appear in the web panel - Notifications with location should be sent to emergency contacts and units	FR-ALM-002, FR-EMC-001, FR-OFF-001
TC-ALM-013-003	Different Alert Messages for Different Threat Levels	The content of alerts should reflect the severity of the hazard.	- System must be operational - Mobile apps installed and logged in - Earthquake sensor simulator can send different thresholds	Simulated light, moderate, and severe earthquake data	1. Send light, moderate, and severe earthquake data respectively 2. Check mobile notifications after each input	- Light: Message like “Light Tremor Detected” - Moderate: Message like “Moderate Earthquake! Move to Safety” - Severe: Message like “Severe Earthquake! Take Cover Immediately and Seek Help if Needed”	FR-ALM-003
TC-ALM-013-004	Sending Alerts Through Multiple Communication Channels	Using multiple channels increases the chance that users receive the alert.	- System must be operational - Mobile apps installed and logged in - User contact info (phone, email) registered - Alarm (e.g., earthquake) must be triggered	Simulated earthquake alarm	1. Trigger an earthquake alarm in the system	- Mobile notifications should be sent - SMS alerts should be received (if configured) - Email alerts should be received (if configured) - Message content should be consistent across channels	FR-ALM-004

10. Glossary

Term	Definition
IoT (Internet of Things)	A network of physical objects ("things") embedded with sensors, software, and other technologies that enable them to collect and exchange data.
Sensor	A device that detects or measures a physical property (like temperature, pressure, light, or motion) and records, indicates, or otherwise responds to it.
Mobile Application	A software application designed to run on mobile devices such as smartphones or tablets.
Web-based Management Panel	A user interface accessible through a web browser that allows administrators or authorized users to manage and monitor a system.
Real-time	Occurring immediately or without any noticeable delay.
Notification	A message or alert displayed on a device to inform the user about something.
Emergency Contacts	Individuals designated by a user to be notified in case of an emergency.
Server	A computer or system that provides resources, data, services, or programs to other computers, known as clients, over a network.
UI (User Interface)	The means by which a user interacts with a computer system, application, or device.
Scalability	The ability of a system to handle an increasing amount of work or its potential to be enlarged in order to accommodate growth.
Vulnerability	A weakness or flaw in a system that could be exploited by a threat actor, leading to unauthorized access or other harmful outcomes.
Regression Testing	A type of software testing performed to confirm that recent program or code changes have not affected existing functionality. It ensures that previously working parts of the application still function correctly after modifications.

11. References

[1] Atlassian. (2025). Free Test Plan Template | Confluence. Retrieved from https://www.atlassian.com/software/confluence/resources/guides/how-to/test-plan

[2] BrowserStack. (n.d.). Performance Testing: Types, Metrics and How to. Retrieved from https://www.browserstack.com/guide/performance-testing

[3] Fowler, M. (2003). Unit Test. Retrieved from https://martinfowler.com/bliki/UnitTest.html

[4] International Software Testing Qualifications Board. (2025). ISTQB - The leading global provider of certification for software testing professionals. Retrieved from https://www.istqb.org/

[5] Nielsen Norman Group. (2025). Usability Testing. Retrieved from https://www.nngroup.com/

[6] Open Web Application Security Project. (2025). OWASP. Retrieved from https://owasp.org/

[7] Ray, P. P., Dash, D., & De, D. (2017). Testing of IoT systems: Challenges and research directions. International Journal of Pervasive Computing and Communications, 13(2), 130-159.

[8] Alsmadi, I., & Magelssen, M. (2014). A systematic review of testing techniques for mobile applications. Information and Software Technology, 56(11), 1298-1317.

[9] Patton, R. (2019). Software testing. Pearson Education.

[10] Jennex, M. E. (2010). Emergency response information systems: The effects of usability and training on performance. International Journal of Information Systems for Crisis Response and Management, 2(3), 33-47.

[11] TestDevLab. (2024, September 25). Integration Testing 101: Methods, Challenges & Best Practices. Retrieved from https://www.testdevlab.com/blog/integration-testing-101

[12] Shneiderman, B., & Plaisant, C. (2016). Designing the user interface: Strategies for effective human-computer interaction (6th ed.). Pearson.