REPUBLIC OF CAMEROON
PEACE-WORK-FATHERLAND
UNIVERSITY OF BUEA
BUEA, SOUTH-WEST REGION
P.O BOX 63.

RÉPUBLIQUE DU CAMEROUN
PAIX-TRAVAIL-PATRIE
UNIVERSITÉ DE BUEA
BUEA, RÉGION DU SUD-OUEST
B.P. 63.

FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING
SOFTWARE ENGINEERING
CEF440: Internet and Mobile Programming

SYSTEM REQUIREMENTS ANALYSIS FOR A DISASTER MANAGEMENT MOBILE APPLICATION

Presented by:

GROUP 2

NAMES	MATRICULE
QUINUEL TABOT NDIP-AGBOR	FE21A300
SIRRI THERESIA ANYE	FE21A306
NGONCHI RAMATOU YOLAND	FE21A157
CHE BLAISE NJI	FE21A260
LIMA CHARLES	FE21A225

Course Facilitator:
Dr. NKEMENI Valery, PhD

Date Issued:
12th May 2024

Table of Content

1. Abstract

This document outlines the system requirements analysis for a mobile disaster management application. Traditional disaster management methods often lack efficiency due to manual processes. This project proposes a mobile application to address these challenges by creating a comprehensive platform for disaster preparedness, response, and recovery.

This mobile application aims to leverage the power of mobile technology to improve communication, information sharing, and overall effectiveness in disaster management situations.

2. Introduction

Natural disasters and emergencies pose a constant threat to communities worldwide. Technology can play a vital role in mitigating their impact by empowering individuals and facilitating coordinated response efforts. This essay explores the technical requirements and tools for building a robust and user-friendly disaster management mobile application specifically designed as a native Android app.

3. Requirements Analysis For A Disaster Management Mobile Application

System requirements define the characteristics and functionalities a disaster management mobile application must possess to meet its objectives. These requirements can be categorized into different types:

- **Functional Requirements:** These define the specific actions the application should be able to perform. In our case, this includes features like real-time alerts, incident reporting, and access to resources.

- **Non-Functional Requirements:** These address overall qualities of the application, like performance, security, and usability. Examples include real-time delivery of alerts and ensuring the application is accessible to diverse users.

- **Technical Requirements:** These specify the technical aspects needed to develop the application, such as integrating with mapping services or implementing multi-language support.

- **Stakeholder and User Requirements:** These capture the specific needs of the application's intended users. Examples include customized alerts based on location and forums for community engagement.

3.1. Functional Requirements

1. **Real-time Alerts and Notifications:**

    - System should deliver critical information like warnings about impending disasters, evacuation orders, and safety instructions.
    - Allow for user preferences to customize alerts based on location severity (e.g., receive alerts only for major disasters or within a specific radius).
    - Multiple notification channels can be used (push notifications, text messages, in-app alerts).

2. **Incident Reporting and Assistance:**

    - Users can report incidents (e.g., flooding, fire damage, injuries) directly through the app.
    - Reporting should include options to capture details like location (using GPS), type of incident, photos/videos (if possible), and any additional relevant information.
    - The system should allow users to request assistance from specific emergency services (police, fire department, ambulance) based on the reported incident.

3. **Geospatial Data and Mapping Services:**

    - Integrate with real-time geospatial data services to display disaster-affected areas on interactive maps.
    - Maps should show key information like evacuation routes, shelter locations, flood zones, and critical infrastructure status.
    - Users' current location should be displayed on the map for better situational awareness.

4. **Community Engagement and Collaboration:**

- Integrate features like forums, chat rooms, and social media feeds to facilitate information sharing and communication among users and stakeholders involved in disaster response.
- Users can share updates, request help, offer assistance, and connect with others in their community during emergencies.
- Allow for anonymous posting if desired to encourage broader participation.

5. **Preparedness and Mitigation Resources:**

    - Provide a centralized repository of educational resources, guides, and checklists to help users prepare for different types of disasters.
    - Content should cover topics like creating emergency plans, assembling disaster kits, and practicing safety measures.
    - Resources can be downloadable for offline access.

6. **Emergency Contacts and Services:**

    - Offer a comprehensive directory of emergency contact information for various services (police, fire department, hospitals, poison control).
    - Allow users to easily call or message emergency services directly from the app.
    - Information should be categorized and searchable for quick access during emergencies.

7. **Offline Capability and Data Sync:**

    - The app should function to some extent even without an internet connection.
    - Users should be able to access critical information like alerts, maps (pre-downloaded versions), and emergency resources offline.
    - The system should queue incident reports and other user-generated data when offline, automatically syncing it with the server when an internet connection is re-established.

8. **Multi-Language Support:**

    - The app interface and content should be available in multiple languages to cater to a diverse user base.
    - Allow users to easily switch between languages based on their preference.
    - Localization should consider cultural nuances to ensure clear and effective communication.

3.2. Non-Functional Requirements

1. **Privacy and Security Measures:**

- **Data Encryption:** Implement strong encryption algorithms to protect user data during transmission and storage. This includes sensitive information like location, incident details, and potentially even personal information used for registration (if applicable).

- **Authentication and Authorization:** Consider user authentication mechanisms (logins, passwords) to restrict access to sensitive functionalities or user data (optional depending on app design).

- **Data Minimization:** Collect only the data essential for app functionality. Avoid unnecessary data collection to minimize privacy concerns.

- **Transparent Privacy Policy:** Clearly outline the app's data collection practices, how user data is used, and user rights regarding their data.

2. **Real-time Delivery of Alerts and Notifications:**

- **Low Latency Communication:** Utilize efficient communication protocols and server infrastructure to minimize delays in delivering critical alerts and notifications. This is crucial for ensuring timely warnings during emergencies.

- **Push Notification Optimization:** Optimize push notification delivery to minimize reliance on a stable internet connection. Explore alternative methods like SMS or local network broadcasts for critical alerts in case of internet outages.
- **Notification Retries:** Implement mechanisms to retry sending notifications if there are initial delivery failures due to network issues.

3. Accuracy and Reliability of Geospatial Data:

- **Data Source Validation:** Establish partnerships with reliable sources for geospatial data, such as government agencies or reputable mapping services.

- **Data Refresh Mechanisms:** Implement processes to regularly update geospatial data, especially during emergencies when information about disaster zones and evacuation routes can change rapidly.

- **Data Accuracy Checks:** Integrate data validation techniques to identify and correct any potential inaccuracies in the geospatial data before it's displayed in the app.

4. **Accessibility of the Application:**

- **WCAG Compliance:** Strive for conformance with the Web Content Accessibility Guidelines (WCAG) to ensure the app is accessible to users with disabilities. This includes features like:

    - **Screen Reader Compatibility:** The app interface should be compatible with screen reader software for visually impaired users.
    - **Increased Colour Contrast:** Implement sufficient colour contrast between text and background elements for better readability by users with visual impairments.
    - **Keyboard Navigation:** Allow users to navigate the app interface using a keyboard instead of a touchscreen, catering to users with motor impairments.

3.3 Stakeholders and User Requirements

- **Stakeholders:** These are individuals or organizations that have a vested interest in the success of the application but may not directly use it themselves. They are impacted by the application's effectiveness and functionality.

- **Emergency Responders (Police, Fire Department, Ambulance):**

    - **Needs:** Real-time information on incidents, secure communication channels for coordination, efficient resource allocation based on user reports.
    - **Benefits:** Improved response times, better situational awareness, streamlined collaboration.

- **Government Agencies:**
    - **Needs:** Dissemination of critical information to the public, data collection for post-disaster analysis, centralized platform for resource management.
    - **Benefits:** Enhanced public safety and awareness, improved coordination between emergency services, informed decision-making for future preparedness efforts.

- **Non-profit Organizations and NGOs:**
    - **Needs:** Platform for sharing resources and connecting with affected communities, ability to coordinate volunteer efforts.
    - **Benefits:** Increased visibility and outreach, streamlined resource distribution, efficient deployment of volunteers.

Users (General Public): These are the individuals who will directly interact with the mobile application to access its features and functionalities.

- **Needs:** Timely and location-specific alerts, ability to report incidents and request assistance, access to emergency resources and evacuation information.
    - **Benefits:** Improved preparedness, increased sense of security, ability to connect with others during emergencies, access to critical resources during disasters.

- **Additional Considerations:**
    - **Accessibility:** Ensure the application caters to users with disabilities (visual impairments, hearing impairments, etc.) through features like text-to-speech functionality or closed captions.
    - **Multilingual Support:** The application should be available in multiple languages to reach a diverse user base.

3.4 Technical Requirements

Developing a feature-rich and effective disaster management app necessitates a clear understanding of the technical needs. Here's a breakdown of key requirements:

- **Front-End Development:**
    - **User Interface (UI) Design:** Figma, a web-based design tool, allows for collaborative UI design, prototyping, and creating a user-friendly and intuitive interface for the application.
    - **Native App:** Developing a native Android app using tools like Android Studio ensures optimal performance, access to native device features (GPS, camera) and a seamless user experience tailored for the Android platform.
- **Back-End Development:**
    - **React Native:** This JavaScript framework enables building native-looking mobile apps using reusable components, streamlining development and potentially reducing maintenance costs.
    - **Node.js:** This server-side JavaScript runtime environment facilitates building scalable and real-time functionalities like real-time alerts and communication features.
- **Database:**
    - **MongoDB:** This NoSQL document database offers flexibility and scalability for storing various types of data, including incident reports, user information, and resource locations.
- **Additional Considerations:**
    - **Real-time Communication:** Integrate real-time messaging libraries like Socket.IO or Pusher to enable real-time communication features like chat rooms or push notifications.
    - **Offline Functionality:** Implement mechanisms like caching and local storage to allow users to access critical information (e.g., alerts, maps) even without an internet connection.
    - **Security:** Prioritize robust security measures like data encryption, user authentication, and secure API access to protect user data and system integrity.

3.5 Requirement Gathering Analysis

Developing a successful disaster management mobile application hinges on understanding the needs of its users and stakeholders. This document outlines a comprehensive approach to requirement gathering for this crucial technology.

Requirement Gathering Techniques:

We employed various techniques to gather requirements from stakeholders and users:

- **Stakeholder Workshops:** Facilitating discussions to identify critical needs, functionalities, and success metrics.
- **User Interviews and Surveys:** Gathering user feedback on their expectations from the application and potential pain points during disaster scenarios.
- **Focus Groups:** Conducting moderated discussions with diverse user groups to explore needs, preferences, and accessibility considerations.
- **Scenario Mapping:** Simulating potential disaster situations and brainstorm functionalities that would be helpful in those scenarios.
- **Competitive Analysis:** Reviewing existing disaster management applications to identify best practices and potential areas for improvement.

Documenting Requirements:

Capture all gathered requirements in a clear and organized manner. This involved:

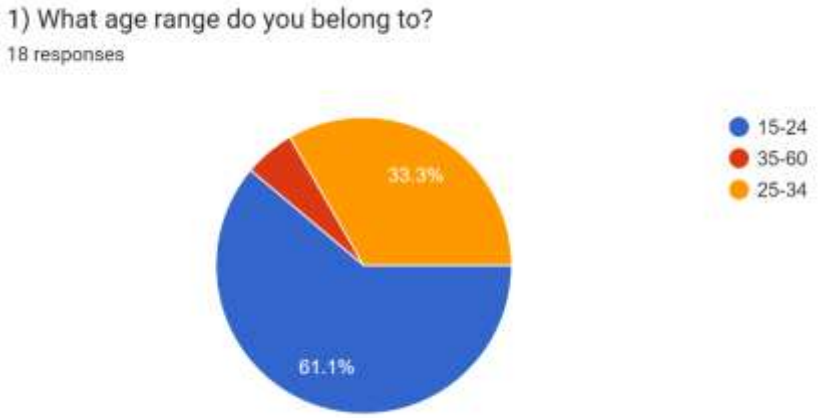
- **Use Cases:** Describe specific user interactions with the application to achieve particular goals.
- **User Stories:** Briefly describe functionalities from the user's perspective (e.g., "As a resident, I want to receive real-time alerts about approaching storms").
- **Functional Requirements:** Define the specific actions the application should be able to perform (e.g., display interactive maps, allow users to report incidents).
- **Non-Functional Requirements:** Specify performance, usability, security, and accessibility considerations.

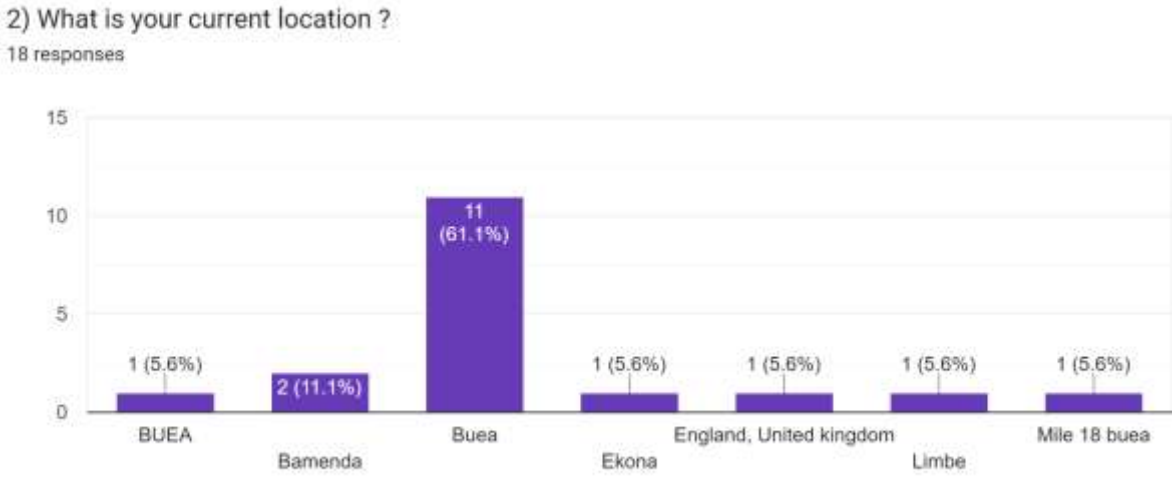
Prioritization:

Not all requirements have equal importance and for that reason we will be prioritising them in the following order: **Usability, Reliability, Performance, Scalability and Security**. So we employed strategies like survey forms, interviews, and research .

Survey Form Analysis:

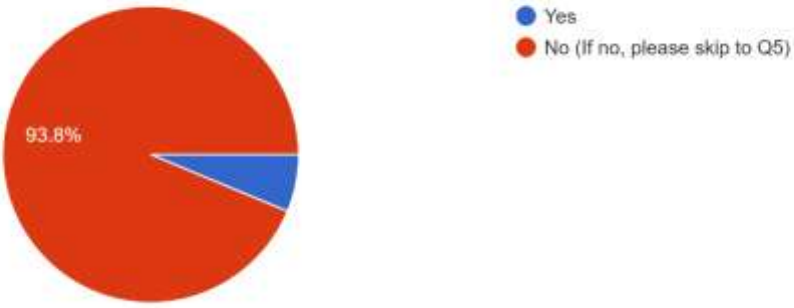
We sent out a survey with 8 questions with over 20 responses using Google Forms which made is quite easy to collect and analysis statistics. The questions were asked on the survey asked for things like age group, disaster experience and possible features for implementation as well as suggestions. The given set of questions prioritised the mindset of users to better understand their needs and wants. As seen below:





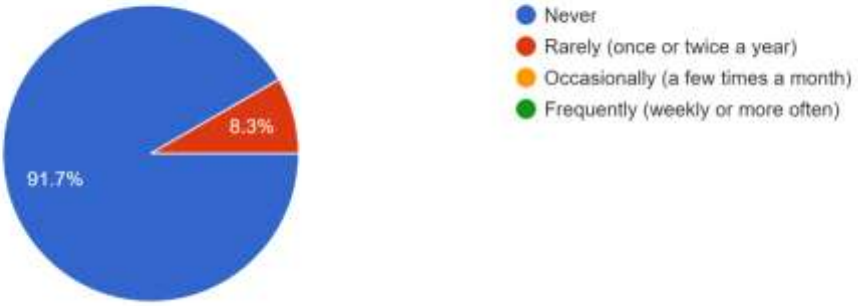
3) Have you ever used a disaster management mobile application?

16 responses



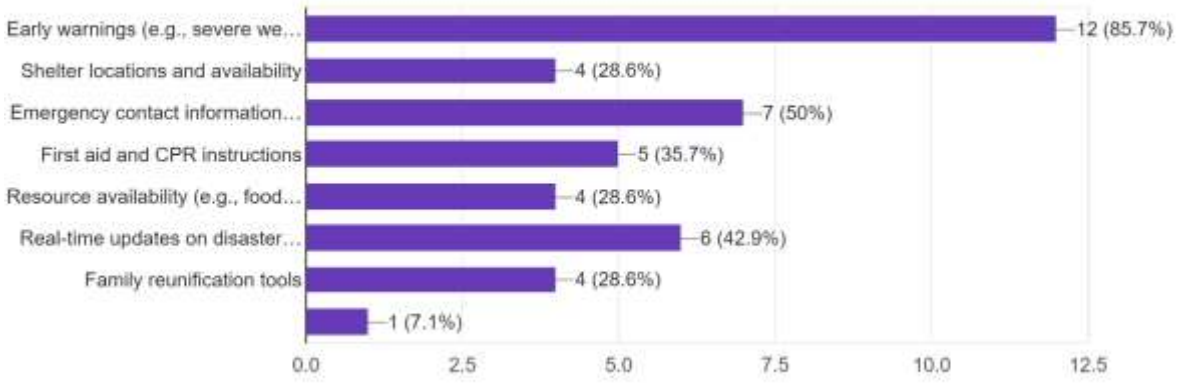
4) How often do you use a disaster management app?

12 responses



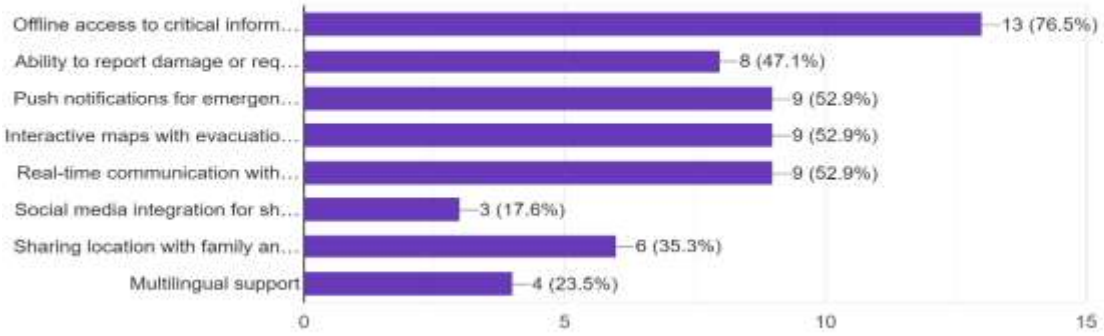
5) What disaster information is most important to you in a mobile app

14 responses



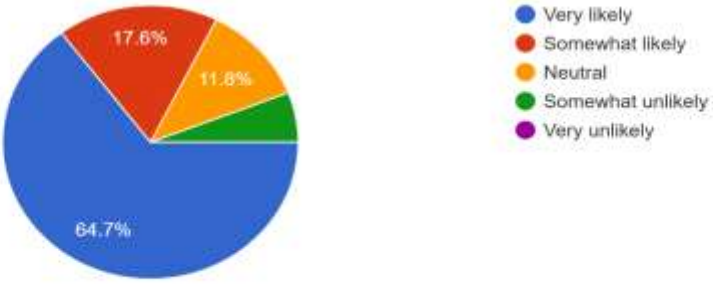
6) What features would be most helpful during a disaster?

17 responses



7) How likely are you to recommend our app to others?

17 responses



8) What suggestions do you think will be necessary in a disaster management mobile application ?

10 responses

Have a geospatial facilities

Climate change report section

Aid to dose affected

First aid

The app should be efficient and reliable

.

I don't know since I have never used one

A way to know how fast and how far the disaster is going

It should be free

Research Analysis:

Developing a disaster management mobile application requires understanding user needs and existing solutions. Leveraging internet research and interviews to gather valuable insights:

Internet Research:

- **Academic Papers and Reports:** Search for research papers and reports on disaster management mobile applications. These provides insights into user needs, successful functionalities, and common challenges. Look for reputable sources like academic journals, government agencies (e.g., FEMA in the US), or disaster response organizations.

- **Existing Disaster Management Apps:** Analysing existing disaster management applications available on app stores.
    - Identifying features they offer (e.g., real-time alerts, incident reporting, resource maps).
    - Reading user reviews to understand user pain points and areas for improvement.
    - Exploring competitor apps to identify potential gaps your application can address.

- **Industry Reports and News Articles:** Look for industry reports and news articles about disaster management trends and technological advancements. These can highlight emerging technologies like real-time data analytics or wearable device integration that could enhance your app.

Interviews:

- **User Interviews:** Interviewing potential users of the disaster management application, focusing on individuals residing in disaster-prone areas.
    - Learning about their concerns and information needs during emergencies.
    - Identifying what functionalities they would find most valuable and how they typically access information during disasters (e.g., smartphones, internet access).
    - Including people with disabilities to ensure the app is accessible to everyone.

Integrated Analysis of Requirements for Disaster Management Mobile App

Developing a disaster management mobile application necessitates a comprehensive analysis of various requirement categories to ensure an effective and user-centric solution. Here's an integrated analysis combining functionalities, technical considerations, stakeholder needs, and user experience:

Functional Requirements Analysis:

- **Strengths:** Core functionalities like real-time alerts, incident reporting, and resource access empower individuals and support coordinated response efforts. Interactive

maps and customizable alerts enhance user experience and situational awareness. Forums and social media integration foster community engagement and information sharing.

- **Weaknesses:** Complexity of features might impact usability, especially for non-tech-savvy users. It's crucial to prioritize functionalities based on critical needs and conduct usability testing to ensure ease of use. Disaster-specific considerations may require additional features tailored to the prevalent types of disasters in the target region.

Non-Functional Requirements Analysis:

- **Strengths:** Emphasis on data security, privacy, and real-time information delivery is crucial for user trust and effective response. Accessibility features like text-to-speech and larger fonts cater to users with disabilities, promoting inclusivity. Offline functionality with data synchronization ensures app usability even in areas with limited internet connectivity.

- **Weaknesses:** Balancing performance with resource limitations on mobile devices can be challenging. Defining clear metrics for "real-time" delivery and data accuracy can be complex. Performance optimization techniques and robust server infrastructure are essential to address these challenges.

Stakeholder and User Requirements Analysis:

- **Stakeholders:** Emergency responders benefit from real-time incident reports, secure communication channels, and efficient resource allocation based on user data. Government agencies can leverage the app for public information dissemination, data collection for post-disaster analysis, and centralized resource management. NGOs can use the platform for resource sharing, volunteer coordination, and increased visibility.

- **Users:** The application empowers users with timely and location-specific alerts, the ability to report incidents and request assistance, and access to emergency resources and evacuation information. Accessibility features and multilingual support ensure the app caters to a diverse user base. Effective stakeholder and user engagement through workshops, surveys, and focus groups is crucial to understand their specific needs and priorities.

Technical Requirements Analysis:

- **Development Tools:** Leveraging tools like Figma for UI design, React Native for native Android app development, Node.js for back-end functionalities, and MongoDB for a flexible database facilitates efficient development and scalability.

5. Conclusion

Building a disaster management mobile application requires careful consideration of technical needs, user experience, and data management principles. By leveraging tools like Figma, React Native, Node.js, and MongoDB, developers can create a robust and user-friendly application. A well-defined CAP analysis strategy helps prioritize consistency, availability, and partition tolerance based on the specific requirements of the application in a disaster scenario. This combination of technology and planning can empower individuals and communities to be better prepared, informed, and connected during emergencies, fostering a more resilient future.

6. References

1) Enhanced 911 http://library.ema.gov.au/LIBERO/WebOpac.cls

2) SMS Alerts http://www.egov4dev.org/mgovapplic.htm

3) SMS emergency broadcasting http://www.egov4dev.org/mgovapplic.htm

4) Mobile devices to government authorities www.jsce-int.org/Report/report/flood_euro.pdf

5)The value of GSM in a natural disaster situation. http://www.gsmworld.com/news/press_2000/press_releases_43.shtml

6) New emergency response system helps fight disasters (EGERIS) http://www.innovations-report.com/html/reports/communication_media/report-29094.html

7)Disaster Emergency Management Information System (AFAYBIS) http://www.isprs.org/istanbul2004/comm4/papers/339.pdf