

**FACULTY ENGINEERING**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGINEERING, SE**

**CEF440: Internet and Mobile Programming**



**Presented by:**

| BOBGALA HARRISON DIMA | FE21A151 |
| --- | --- |
| KAMCHE YANN ARNAUD | **FE21A208** |
| MESSI II INNOCENT ROSTAND | **FE21A236** |
| NYANDO ONONGWENE | **FE21A290** |
| NKWI CYRIL AKINIMBOM | **FE21A281** |



**Table of Content**

[**1. User Interface Design 2**](#_299iefckpwku)

[1.1. Design Process 2](#_6flidnb810lw)

[1.2. Visual Design 3](#_91snks687u9y)

[1.3. Interaction Design 6](#_woqinauuefyt)

[1.4. Accessibility 7](#_igouyf8moqxp)

[**2. Functionalities 8**](#_493shep21mdc)

[2.1. Core Features 8](#_y2qnamja59e)

[2.2. Service Integration 8](#_4ttuj77xeprj)

[2.3. Offline Functionality 8](#_orjwz2ff0wlj)

[**3. Implementation Details 8**](#_3rdcrjn)

[3.1. Tech Stack 8](#_bp64r2mordcn)

[3.2. Architecture 8](#_k9ideq6dscyc)

[3.3. Code Structure 8](#_bj3kdn2mvmti)

[3.4. Version Control 8](#_t3nmonoh2at8)

[**4. Testing and Quality Assurance 8**](#_b5814h4h9h0r)

[4.1. Usability Testing 8](#_ddp0cm266s91)

[4.2. Functional Testing of the User Interface using Jest 9](#_xjvv6682wc1n)

[**5. Deployment and Maintenance 12**](#_cx4ntnt3p1gt)

[5.1. Deployment Process 12](#_qale5l4d1gr5)

[5.2. Maintenance Plan 13](#_7tgq4l1jtpwl)

[5.3. User Support 14](#_edpy6raojo44)

# **User Interface Design**

User Interface (UI) design is a crucial aspect of any mobile application, especially for a disaster management app where usability and accessibility can make a significant difference in emergency situations. A well-designed UI enhances user experience, making it easier for users to navigate through the app, access important information quickly, and take necessary actions during critical times. In this section, we will discuss the implementation of UI design principles for the disaster management mobile app, focusing on layout, visual design, interaction design, and accessibility.

## 

## Design Process

A robust iterative design process was followed to ensure the disaster management app meets the needs of its users effectively. The design process involved multiple iterations based on user feedback and usability testing. This iterative approach ensured that the app's UI evolved to meet user expectations and preferences. The process involved the following steps:

**1)Research and Analysis:** Comprehensive research was conducted to understand user needs, behaviors, and pain points related to disaster management. This included studying existing apps, conducting user interviews, and analyzing feedback.

**2)User Persona Development:** Based on research findings, user personas were created to represent different user groups and their specific requirements. This helped in designing a user-centric interface tailored to the needs of diverse users.

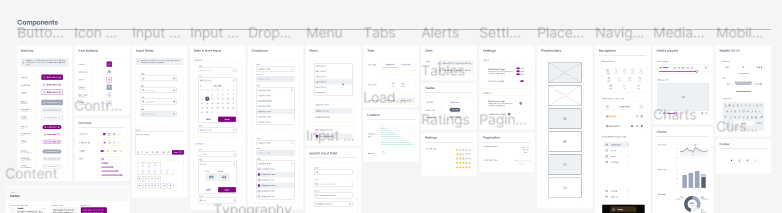
**3)Wireframing:** Low-fidelity wireframes were created to visualize the app's layout and navigation structure using paper sketches and shapes in figma. This facilitated early feedback and iteration..

**4)Prototyping:** High-fidelity prototypes were developed to simulate the app's functionality and gather user feedback before implementation on figma. This made the product ready for user testing before implementation.

**5)Testing and Feedback:** Collaboration among designers, developers, and stakeholders played a crucial role in refining the UI design. Regular feedback sessions and design reviews helped in identifying areas for improvement and making necessary adjustments.

Also,a design system was implemented to maintain consistency and coherence across the app's UI elements, *see Fig1.1* . This included standardized typography, colors, icons, and components, ensuring a cohesive visual identity and user experience.

By following a structured design process, the disaster management app was developed with a user-centered approach, prioritizing usability, accessibility, and functionality.



*Fig1.1: Design System consisting of re-usable components for structured design.*

## Visual Design

Visual design plays a crucial role in creating a cohesive and engaging user interface for the disaster management app. It encompasses the aesthetic aspects of the app, including color scheme, typography, icons, and other visual elements. Here's how we approached our visual design:

**Color Scheme**

The color scheme was carefully selected to evoke the right emotions and enhance usability:

* **Primary Colors:** 800080 (Purple) and 2ABFB3 (Teal) were chosen for their calming and trustworthy qualities.
* **Accent Colors:** Red was used sparingly for emergency buttons and critical alerts to ensure they draw immediate attention.
* **Background Colors:** Light gray or white backgrounds were employed to maintain readability and provide a clean aesthetic.
* Grayscale Colors: These a variants of black and gray which are on the text

*See Fig 1.2*

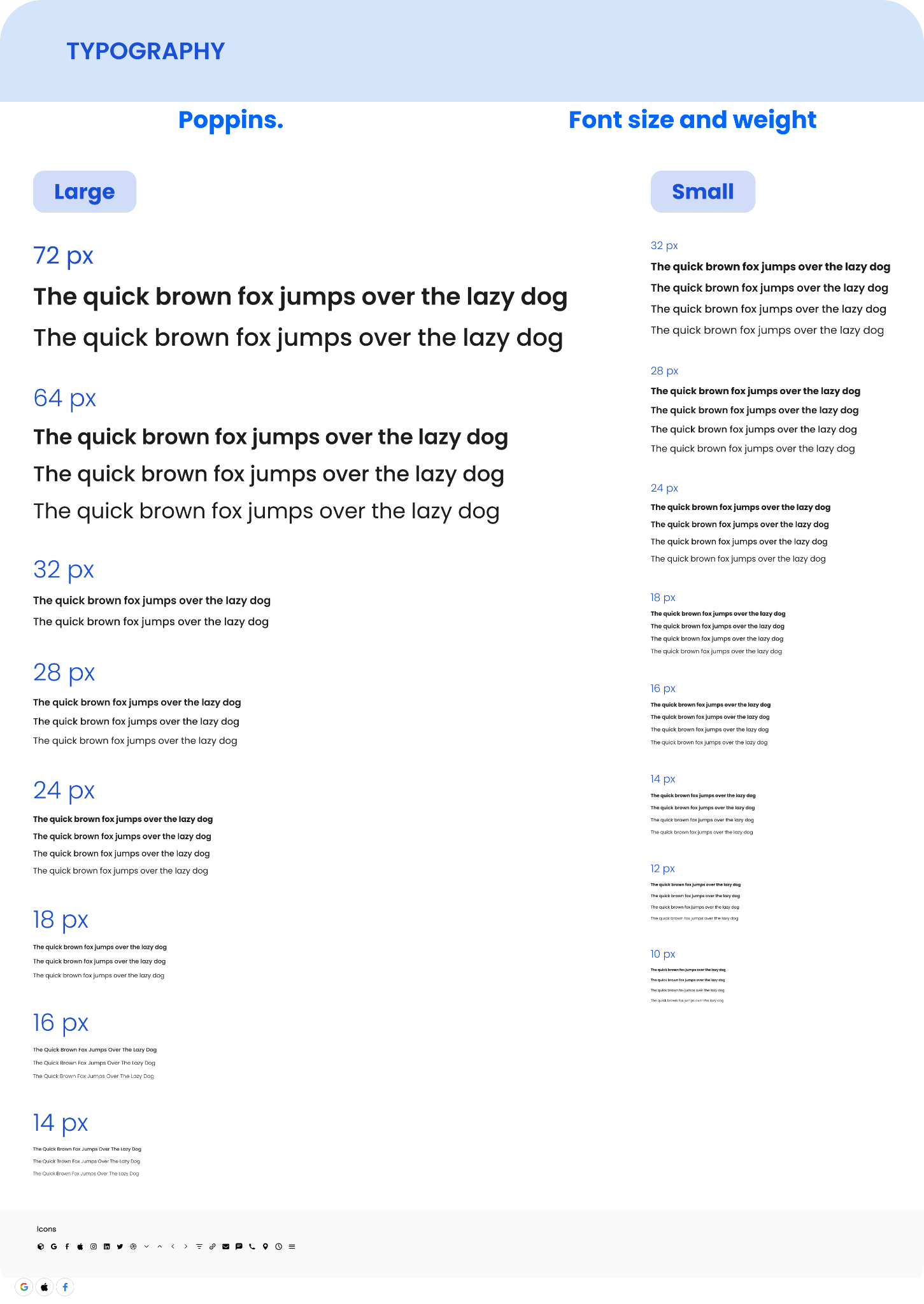
**

*Fig 1.2: Color Palette of HelpAmSafe App for calm and trustworthy feel.*

**Typography**

Typography plays a vital role in readability and visual hierarchy See *Fig 1.3*:

* **Font Choice:** Poppins was selected for its modern appearance and readability across different screen sizes.
* **Font Sizes:** Large fonts were used for headlines and critical information, medium for regular text, and smaller sizes for supplementary details, ensuring clear text hierarchy.

**

*Fig 1.3: Poppins Typography*

**Icons**

Icons were used to enhance usability and provide visual cues:

* **Style:** Simple, flat icons were chosen for their ease of recognition and consistency throughout the app.
* **Use:** Icons were utilized for navigation, action buttons, and status indicators, ensuring clear visual communication.

**Other Visual Elements**

**Buttons:** Designed with rounded edges, clear labels, and contrasting colors to ensure they stand out and are easy to interact with.

**Images:** Contextual images were used sparingly, with descriptive alternative text for accessibility, providing visual context without overwhelming the interface.

## Interaction Design

Interaction design focuses on how users interact with the app, ensuring a smooth and intuitive user experience. It includes navigation, gestures, and form inputs. We will explore the interaction design principles applied to ensure smooth user interactions and intuitive navigation within the app.They include using;

**Buttons**

They were designed to facilitate easy tapping and intuitive actions. These are Call-To-Action buttons (CTA). Some of their qualities include;

* **Clear Labels:** Buttons were labeled clearly to convey their purpose, facilitating easy navigation and action.
* **Size and Placement:** Buttons were sized appropriately and placed strategically to ensure they are easily reachable and stand out from other elements.

**Gestures**

Gestures were employed to enhance navigation and interaction:

* **Swipe Gestures:** Used for navigation between tabs and within content sections, providing a seamless browsing experience.
* **Tap:** Single taps were used to open items, links, and interact with the map, ensuring straightforward interaction.

**Form Inputs**

Form inputs were designed for simplicity and ease of use. They include features such as:

* **Minimalism:** Input fields were kept minimal and essential to reduce user effort and ensure forms are easy to complete.
* **Types:** Dropdowns, radio buttons, and checkboxes were used for selections, ensuring straightforward input without confusion.

## Accessibility

Accessibility is a fundamental aspect of user interface design, ensuring that all users, including those with disabilities, can access and interact with the app effectively. By incorporating accessibility features, we aim to create a more inclusive and user-friendly experience for everyone.

**Screen Reader Support**

* To accommodate users with visual impairments, all interactive elements within the app are labeled with descriptive text for screen readers.
* Proper use of semantic HTML elements and ARIA (Accessible Rich Internet Applications) roles ensures that screen readers can interpret and navigate the app's content accurately.
* Headings, lists, and other structural elements are used to provide a clear and organized hierarchy of information, facilitating easy navigation for screen reader users.

**Visual Contrast and Clarity**

* High contrast between text and background colors improves readability, particularly for users with low vision or color vision deficiencies.
* Clear and legible typography with appropriate font sizes enhances readability for all users, including those with visual impairments.
* Consistent use of color and visual elements ensures that important information is easily distinguishable, reducing cognitive load for users with visual impairments.

**Flexible Text Sizing and Layout**

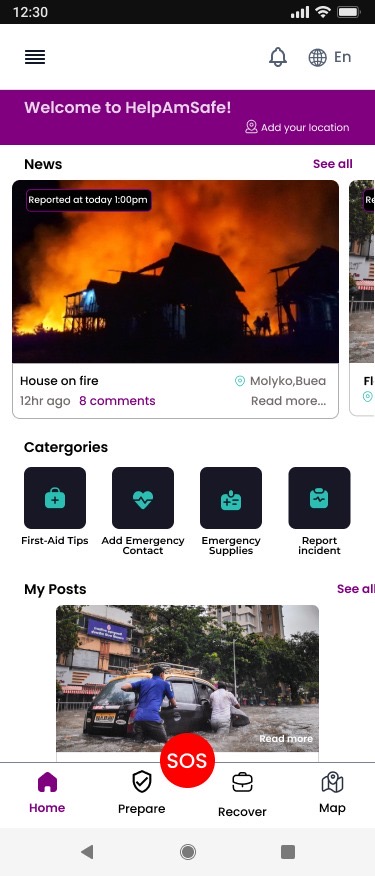
* Support for dynamic text resizing allows users to adjust the text size according to their preferences or accessibility needs, catering to users with visual impairments who may require larger text for readability.
* Responsive design principles ensure that the app's layout adapts to different screen sizes and orientations, providing a consistent and accessible experience across a variety of devices and screen resolutions.

# **Functionalities**

## Core Features

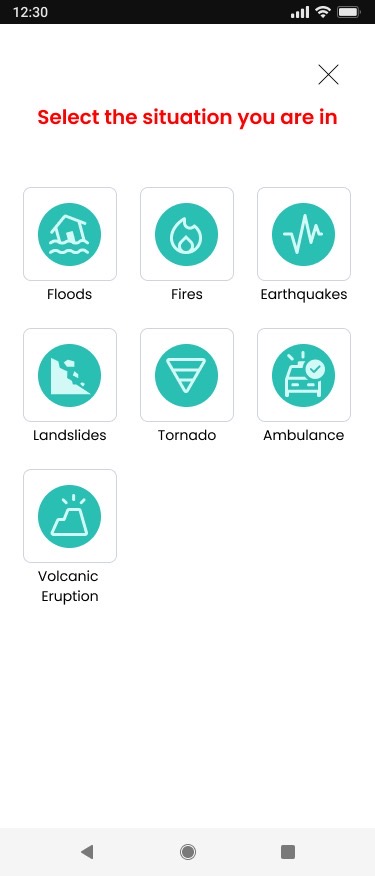
**1. News Feed**

The news feed is a critical component of our disaster management system, designed to provide users with real-time updates and information during emergencies. This feature aggregates news from various reliable sources, ensuring that users have access to the latest developments regarding ongoing disasters. The news feed includes alerts from government agencies, weather updates, and notifications about relief efforts. Users can customize their feed to receive information relevant to their specific location or interests.



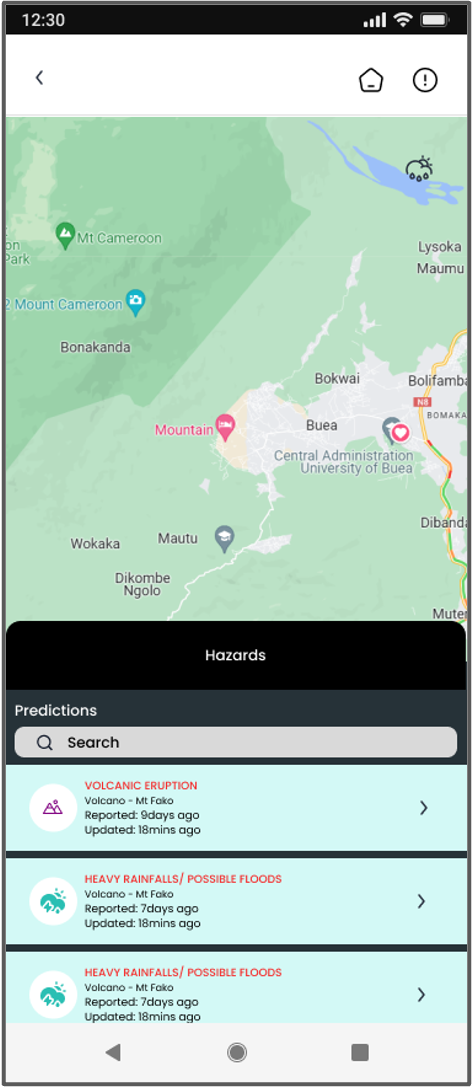
**2. Emergency Contact**

The emergency contact feature is designed to facilitate quick and efficient communication during a disaster. Users can store and access contact information for local emergency services, such as police, fire departments, medical services, and other relevant agencies. Additionally, this feature allows users to send predefined emergency messages to their saved contacts with a single tap, providing critical information such as their current location and the nature of the emergency. This feature aims to reduce response times and ensure help is dispatched swiftly.



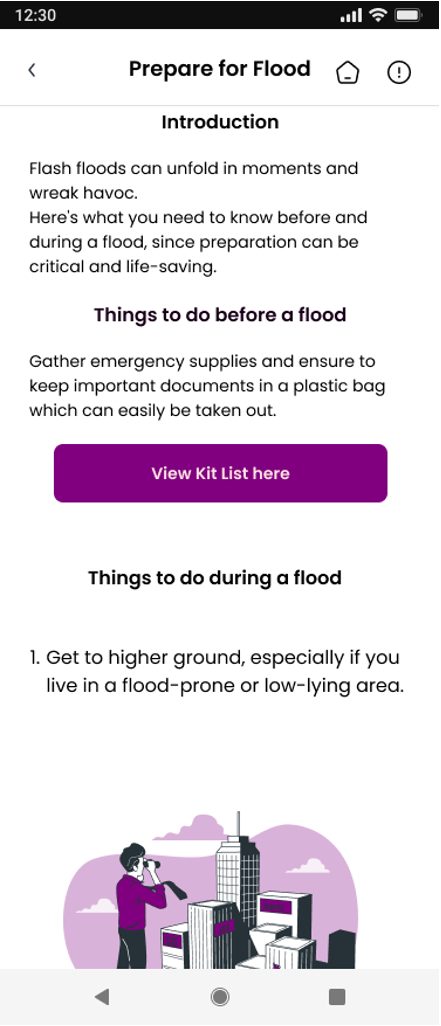
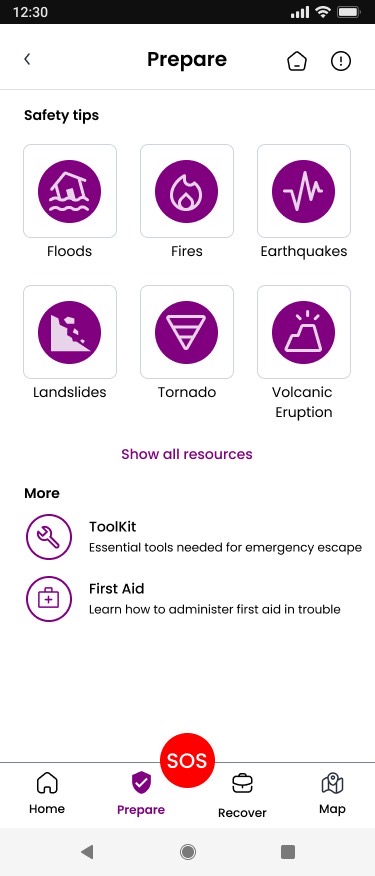
**3. Map**

The map feature integrates advanced geospatial technology to offer users a visual representation of their surroundings during a disaster. The map provides real-time updates on affected areas, evacuation routes, shelter locations, and the status of essential services. Users can navigate the map to find safe zones, road closures, and points of interest such as hospitals and relief centers. The interactive nature of the map allows users to zoom in for detailed local information or zoom out for a broader view of the disaster’s impact.



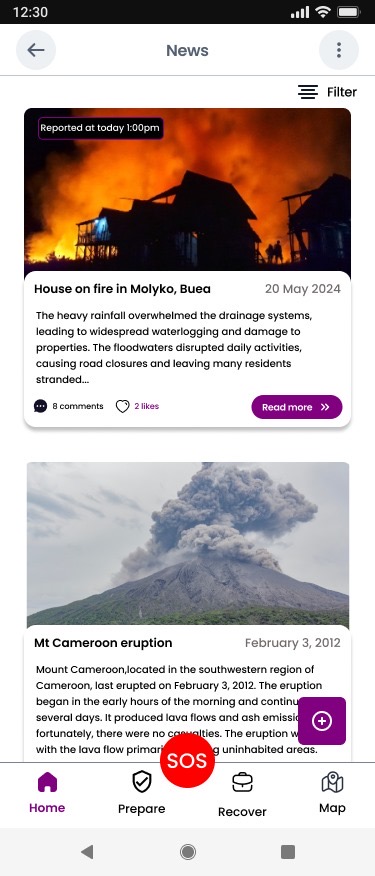
**4. Preparedness Information**

Preparedness information is an educational resource designed to help users prepare for various types of disasters. This feature includes guidelines and checklists for creating emergency kits, developing evacuation plans, and securing homes and workplaces. It also offers tips on how to respond during different types of emergencies, such as earthquakes, floods, hurricanes, and fires. By providing this information, we aim to enhance the overall resilience of users and communities, ensuring they are better equipped to handle emergencies.



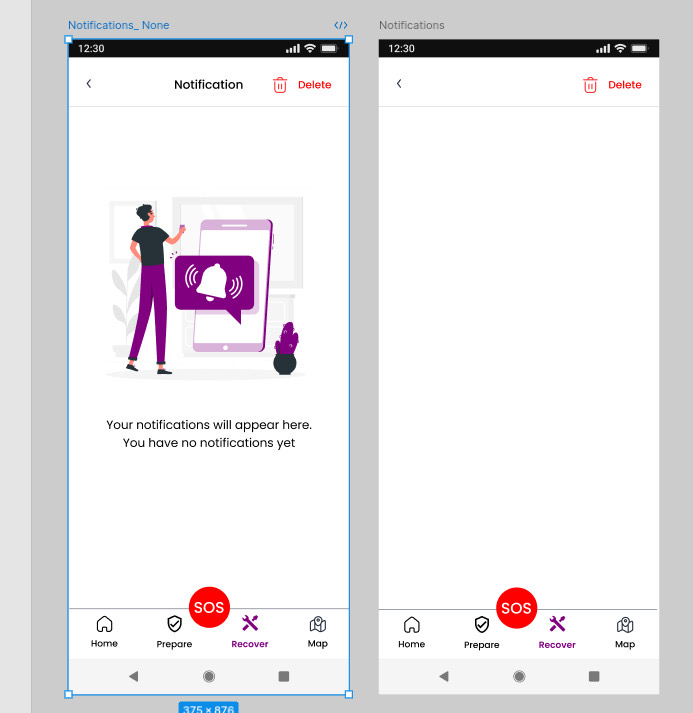
**5. Incident Reporting**

The incident reporting feature empowers users to contribute to the disaster management efforts by reporting incidents in real-time. Users can submit reports detailing the nature of the incident, its location, and any immediate needs or hazards. These reports are then relayed to emergency services and relevant authorities to facilitate a rapid response. The system also allows users to attach photos or videos to their reports, providing valuable visual information that can aid in assessment and response efforts. This feature enhances situational awareness and enables a more coordinated and effective disaster response.



**6. Notifications About Incidents**

Notifications about incidents are a vital feature that ensures users receive immediate alerts about emergencies and critical developments. The system sends push notifications to users’ devices, providing real-time information on nearby incidents, evacuation orders, safety instructions, and other urgent updates. These notifications can be customized based on user preferences and location settings, ensuring that users receive the most relevant and timely information. This feature is designed to keep users informed and safe, facilitating quick decision-making and response during disasters.



## Service Integration

Integrating Google’s map and weather services into our disaster management system enhances the functionality and accuracy of our core features. By leveraging Google Maps and Google Weather APIs, we provide users with real-time geospatial and meteorological data, crucial for making informed decisions during disasters. This section details the integration process and the benefits it brings to our system.

**Features and Benefits:**

1. Real-Time Navigation and Traffic Data:

Google Maps offers real-time navigation and traffic updates, allowing users to find the fastest and safest routes during evacuations. This feature is particularly useful for identifying road closures, traffic jams, and alternative routes.

2. Accurate Geolocation Services:

By utilizing Google’s advanced geolocation services, our map feature can precisely pinpoint users’ locations, ensuring that emergency responders receive accurate coordinates for rescue operations.

3. Detailed Mapping and Satellite Imagery:

Google Maps provides detailed mapping and high-resolution satellite imagery, which can help users visualize affected areas, understand the terrain, and locate essential services such as hospitals, shelters, and relief centers.

4. Customizable Map Layers:

Users can overlay various map layers, such as weather patterns, flood zones, and fire perimeters, to get a comprehensive view of the disaster’s impact. This customization helps in better planning and response.

**Integration Process:**

1. API Key Acquisition:

Obtain API keys from Google Cloud Platform for both Google Maps and Google Places APIs.

2. API Implementation:

Integrate the Google Maps API into our mobile app by embedding the provided code snippets and configuring the API settings to enable required features such as geocoding, routing, and traffic updates.

3. User Interface Design:

Design an intuitive interface that allows users to interact seamlessly with the map, including zooming, panning, and switching between different views (map, satellite, terrain).

4. Testing and Optimization:

Test the map functionality under various conditions to ensure accuracy and reliability. Optimize the app to handle high traffic loads during peak times, such as during a disaster.

## Offline Functionality

During disasters, internet connectivity can often be unreliable or completely unavailable. To ensure our disaster management system remains effective in such situations, it is crucial to incorporate offline functionalities. These offline features will allow users to access essential information and perform critical tasks without relying on an active internet connection.

**1. Offline Maps**

- Pre-Downloaded Maps: Allow users to download maps of their local area in advance. These maps will include critical information such as road networks, evacuation routes, and locations of shelters and essential services.

- Offline Navigation: Enable offline turn-by-turn navigation using GPS data, helping users find safe routes even without internet access.

**2. Stored Emergency Contacts**

- Local Storage of Contacts: Ensure that emergency contact information is stored locally on the device, allowing users to access and call emergency services or predefined contacts without needing internet connectivity.

- Predefined Emergency Messages: Allow users to send predefined emergency SMS messages to their contacts, which can include their current GPS location.

**3. Offline Incident Reporting**

- Local Report Storage: Allow users to create and store incident reports locally on their device when offline. Once connectivity is restored, these reports can be automatically uploaded to the central system.

- Multimedia Attachments: Enable users to attach photos and videos to incident reports even when offline, which will be uploaded along with the report when connectivity is available.

**4. Offline Preparedness Information**

- Pre-Loaded Guides and Checklists: Provide users with access to a comprehensive set of preparedness guides and checklists stored locally on their device. This includes information on how to prepare for different types of disasters and step-by-step emergency procedures.

- Interactive Checklists: Allow users to interact with and check off items on preparedness checklists without needing an internet connection.

**5. Offline Notifications and Alerts**

- Cached Notifications: Store recent notifications and alerts locally so that users can access critical information received before losing connectivity.

- Local Alert System: Implement a local alert system that can use the device’s built-in sensors to detect potential hazards (e.g., severe weather warnings) and notify users even when offline.

**6. Data Synchronization**

- Automatic Syncing: Ensure that all data (maps, reports, contacts) created or accessed offline is automatically synchronized with the central server once the device regains internet connectivity. This ensures that users' actions and updates are not lost and that they have the most up-to-date information when online.

# **Implementation Details**

## 1. Tech Stack

**Client-Side Development**

* Framework: React Native
  + Purpose: Used for building the mobile application to ensure a smooth and responsive user interface across both iOS and Android platforms.
  + Features: Component-based architecture, rich ecosystem, hot reloading, and cross-platform compatibility.

**Testing Framework**

* Framework: Jest
  + Purpose: Employed for unit and integration testing of the user interface to ensure functionality, reliability, and performance.
  + Features: Snapshot testing, mocking capabilities, and easy integration with React Testing Library.

## 2. Architecture

**Overview**

The architecture of the disaster management mobile application is designed to ensure scalability, maintainability, and efficiency. It is structured into several layers to separate concerns and improve modularity.

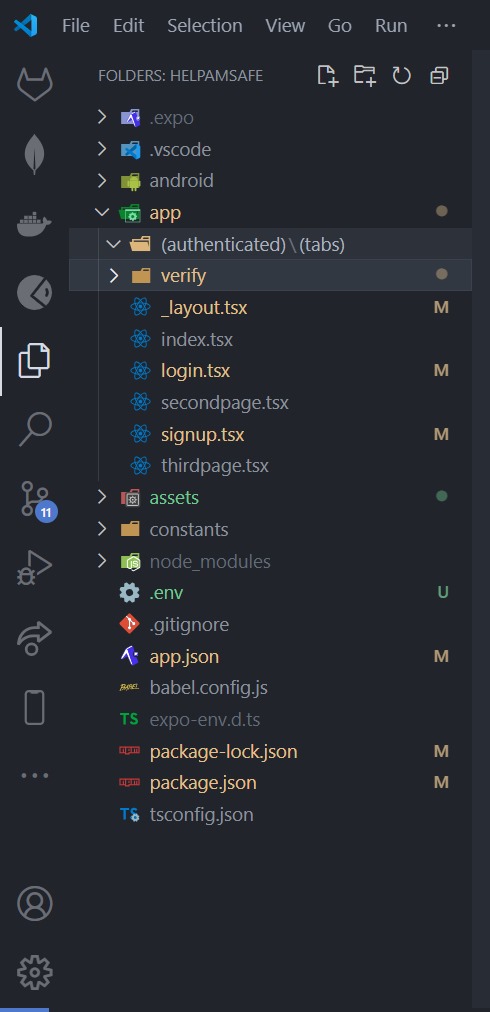
**Layers**

1. Presentation Layer
   * Components: React Native components responsible for rendering the user interface and handling user interactions.
   * State Management: Utilized React hooks and Context API for managing component state and global state.
2. Business Logic Layer
   * Services: JavaScript modules and functions that handle core business logic and data processing.
   * API Integration: Modules that communicate with backend services through HTTP requests using libraries like Axios.
3. Data Layer
   * Local Storage: Utilized AsyncStorage for storing user preferences and session data.
   * Remote Data: Managed remote data fetching and caching mechanisms to ensure data consistency and offline capabilities.

## 3. Code Structure

**Directory Layout**

The project directory is organized to promote clarity and maintainability.



## 

# **Testing and Quality Assurance**

## Usability Testing

**Introduction**

This report outlines the top five usability testing questions that will be used to evaluate the user interface (UI) of our disaster management mobile application. These questions are designed to gather comprehensive insights into the intuitiveness, task completion, clarity, user expectations, and visual/functional issues of the application.

1. **Usability Testing Questions**

* **Pre-test Questions**
  + **Objective:** To gather background information on the user’s experience and expectations before they interact with the application.
  + **Question:** "Have you used similar applications before?"
  + **Rationale:** Understanding the user’s prior experience with similar applications helps gauge their familiarity and set a baseline for their expectations.
* **Task-based Questions**
  + **Objective:** To assess the user’s ability to navigate and complete specific tasks within the application.
  + **Question:** "Was the process to complete this task clear to you?"
  + **Rationale:** This question evaluates the clarity and intuitiveness of the task flow, identifying potential areas of confusion or complexity.
* **Post-test Questions**
  + **Objective:** To gather the user’s overall impressions and detailed feedback after completing all tasks.
  + **Question:** "What did you think about the overall usability of the interface?"
  + **Rationale:** Post-test questions provide a holistic view of the user’s experience, highlighting strengths and areas for improvement in the overall interface design.
* **Performance-related Questions**
  + **Objective:** To examine the user’s ability to complete tasks efficiently and without errors.
  + **Question:** "Did you encounter any difficulties while performing the task?"
  + **Rationale:** Identifying performance-related issues helps in pinpointing specific obstacles that users face, which can then be addressed to improve efficiency and ease of use.
* **Perception-related Questions**
  + **Objective:** To explore the user’s subjective feelings about the interface.
  + **Question:** "How would you rate the design of the application?"
  + **Rationale:** User perception of the design influences their overall satisfaction and trust in the application. This question helps gauge the aesthetic appeal and perceived professionalism of the interface.

## Functional Testing of the User Interface using Jest

1. **Introduction**

This report outlines the functional testing strategy for the user interface (UI) of our disaster management mobile application. Functional testing aims to verify that the application functions as intended and meets the specified requirements. Jest, a popular JavaScript testing framework, will be employed to conduct these tests due to its robustness, ease of use, and integration capabilities.

1. **Objectives**
2. **Validate Core Functionality:** Ensure that all UI components perform their intended functions correctly.
3. **Detect Defects:** Identify any defects or issues within the UI components.
4. **Ensure Stability:** Verify the stability and reliability of the application under various conditions.
5. **Scope of Testing**

The scope of this functional testing includes all major UI components of the disaster management application, such as:

* **Login and Authentication Screens:** Verifying user login, logout, and error messages.
* **Dashboard:** Ensuring that all data is correctly displayed and updated in real-time.
* **Alert System:** Testing the creation, display, and acknowledgment of alerts.
* **User Profile Management:** Validating user data updates and settings adjustments.
* **Navigation:** Checking the functionality of navigation menus and links.

1. **Testing Environment**

* **Testing Tool:** Jest
* **Programming Language:** JavaScript
* **Frameworks:** React Native
* **Browsers:** Chrome, Firefox, Safari (for web interfaces)
* **Devices:** Various Android and iOS devices (for mobile interfaces)

1. **Testing Methodology**
2. **Test Case Design:**
   * Each UI component will have a set of test cases designed to validate its functionality.
   * Test cases will cover positive scenarios (where the component behaves as expected) and negative scenarios (where the component handles errors gracefully).
3. **Test Implementation:**
   * Test cases will be implemented using Jest.
   * Each test will include setup, execution, and teardown phases to ensure isolation and repeatability.
4. **Test Execution:**
   * Tests will be executed automatically using continuous integration (CI) pipelines to ensure regular validation.
5. **Test Reporting:**
   * Jest provides built-in reporting tools that will be used to generate test execution reports.
   * Reports will include test results, coverage analysis, and identified defects.

**F. Sample Test Cases**

**Login and Authentication**

* **Test Case 1:** Verify that a user can log in with valid credentials.
  + **Steps:**
    1. Navigate to the login screen.
    2. Enter valid email and password.
    3. Click the login button.
  + **Expected Result:** The user is redirected to the dashboard.
* **Test Case 2:** Verify that an error message is displayed for invalid credentials.
  + **Steps:**
    1. Navigate to the login screen.
    2. Enter invalid email or password.
    3. Click the login button.
  + **Expected Result:** An error message is displayed indicating incorrect email/password.

**Dashboard**

* **Test Case 3:** Verify that the dashboard displays real-time data correctly.
  + **Steps:**
    1. Log in as a user.
    2. Navigate to the dashboard.
    3. Check the data displayed.
  + **Expected Result:** Data on the dashboard matches the backend data source.

**Alert System**

* **Test Case 4:** Verify that an alert can be created and displayed.
  + **Steps:**
    1. Log in as an admin.
    2. Navigate to the alert creation screen.
    3. Enter alert details and submit.
  + **Expected Result:** The alert is displayed on all relevant user dashboards.

# **Deployment and Maintenance**

## Deployment Process

The deployment process for our disaster management mobile application is a critical phase that ensures the app is properly delivered to users and operates seamlessly in real-world environments. The deployment process is structured to minimize downtime, ensure smooth rollouts, and maintain a high level of reliability and performance. Here's a detailed outline of our deployment process:

**1. Build Preparation:**

- Code Freeze: Implement a code freeze to stabilize the application and prevent new changes from being introduced.

- Build Configuration: Configure the build settings for different environments (development, staging, production).

- Environment Variables: Ensure all necessary environment variables are set correctly for each deployment environment.

**2. Continuous Integration and Continuous Deployment (CI/CD):**

- CI/CD Pipeline: Utilize a CI/CD pipeline with tools like Jenkins, GitHub Actions, or CircleCI to automate the build, test, and deployment processes.

- Automated Testing: Run automated tests, including unit tests, integration tests, and end-to-end tests, to ensure the application is stable and bug-free.

**3. Staging Deployment:**

- Staging Environment: Deploy the application to a staging environment that mirrors the production environment.

- User Acceptance Testing (UAT): Conduct UAT with a select group of users to identify any last-minute issues and gather feedback.

- Bug Fixes: Address any issues identified during UAT and update the application as needed.

**4. Production Deployment:**

- Final Build: Create the final build for production using the latest stable codebase.

- Deployment Tools: Use deployment tools such as Fastlane for iOS and Android deployments, ensuring a smooth and automated release process.

- App Store Submission: Submit the application to the Apple App Store and Google Play Store, adhering to their guidelines and requirements.

- Monitoring: Monitor the deployment process in real-time to quickly identify and address any issues that arise.

**5. Post-Deployment Verification:**

- Smoke Testing: Conduct smoke tests in the production environment to verify the critical functionalities of the application.

- Performance Monitoring: Use tools like New Relic, Firebase Performance Monitoring, or Datadog to monitor the app's performance and ensure it meets performance benchmarks.

- Error Tracking: Implement error tracking with tools like Sentry or Crashlytics to capture and address any runtime errors experienced by users.

## Maintenance Plan

Maintaining the disaster management mobile application is essential to ensure its ongoing reliability, security, and performance. The maintenance plan includes regular updates, monitoring, and support activities:

**1. Regular Updates:**

- Security Patches: Regularly apply security patches to address vulnerabilities and protect user data.

- Feature Enhancements: Continuously gather user feedback to introduce new features and improve existing functionalities.

- Bug Fixes: Promptly address and fix any bugs reported by users or identified through monitoring tools.

**2. Monitoring and Analytics:**

- Performance Monitoring: Continuously monitor the app's performance metrics such as load times, response times, and resource usage.

- User Analytics: Track user behavior and usage patterns using tools like Google Analytics for Firebase to understand user needs and improve the app experience.

- Crash Reporting: Utilize crash reporting tools to detect and resolve application crashes quickly.

**3. Backup and Recovery:**

- Data Backup: Regularly backup user data and application data to prevent data loss in case of system failures.

- Disaster Recovery: Implement a disaster recovery plan to restore the application and data swiftly in the event of a catastrophic failure.

**4. Documentation and Training:**

- Documentation: Maintain comprehensive documentation for the application's architecture, deployment processes, and troubleshooting procedures.

- Training: Provide ongoing training for the development and support teams to ensure they are equipped to handle new updates and issues.

## User Support

Providing robust user support is crucial to ensure user satisfaction and address any issues users may encounter with the disaster management mobile application. The user support strategy includes multiple channels and proactive support measures:

**1. Support Channels:**

- In-App Support: Integrate an in-app support feature allowing users to easily report issues or seek help directly from the app.

- Email Support: Provide a dedicated support email address for users to contact for assistance.

- Live Chat: Implement live chat support for real-time assistance during critical times.

- Help Center: Develop a comprehensive online help center with FAQs, troubleshooting guides, and instructional articles.

**2. Proactive Support:**

- User Onboarding: Provide a user-friendly onboarding process with tutorials and guides to help users understand the app's features.

- Push Notifications: Use push notifications to inform users about critical updates, new features, and important alerts.

- Feedback Collection: Regularly collect user feedback through surveys and feedback forms to identify areas for improvement.

**3. Issue Resolution:**

- Ticketing System: Implement a ticketing system to manage and track user issues and ensure timely resolution.

- SLA Management: Establish Service Level Agreements (SLAs) to set expectations for response and resolution times for user issues.

- Knowledge Base: Maintain an internal knowledge base for the support team to quickly reference solutions for common issues.

By following a structured deployment and maintenance plan and providing comprehensive user support, we ensure the disaster management mobile application remains reliable, secure, and user-friendly, ultimately contributing to its effectiveness in emergency situations.