**REPORT ON MOBILE APP DISASTER MANAGEMENT**

**PRESENTED BY**

**GROUP**

MEMBERS

EBAN BORIS MANYO (SENG22SE006)

NYENTY OBI RAMSON(SENG22SE008)

IKOME BRYAN JOHNSON MONDE(SENG22SE007)

AKAWUNG GABRIEL (SENG22TE001)

**1. Review and Analysis of Gathered Requirements**

**Completeness**

The following features are commonly expected in a disaster management app:

* User registration and authentication
* Emergency alert broadcasting (from authorities/admin)
* SOS distress signal from users (with GPS location)

Real-time map with disaster zones

Emergency contacts and shelters

Disaster preparedness tips

Push notifications and SMS alerts

Community helep

Multi Lingual

**Clarity**

Examples of Clear Requirements:

“The system shall allow users to register using email or phone number.”

“Admin can send broadcast alerts to all users within a selected region.”

Examples of Unclear Requirements:

“Send emergency information” → Who sends it? What kind of information? To whom?

“Show disaster map” → What data source is used? What type of map (static, live)?

Analysis: Clarify vague verbs like "show", "send", "manage", and ensure every actor (user/admin/system) is well-defined.

**Technical Feasibility**

Feasible Features:

Push notifications (via Firebase)

GPS-based location services

REST API for alert management

Potentially Challenging Features:

Offline alerting system (requires local caching and conditional sync)

Real-time maps (requires integration with GIS or mapping APIs)

Multilingual disaster messages (requires content management logic)

Analysis: Most features are feasible for mobile development with Android/iOS SDKs. Ensure platform capabilities are matched to requirements.

**Dependency Relationships**

User registration is required for sending personal SOS alerts.

Location access is required for both alert targeting and map features.

Admin panel must exist before alerts can be created or broadcast.

Analysis: Recognize these relationships to avoid implementation conflicts later.

**Identified Inconsistencies, Ambiguities, or Missing Information**

**Issue Type**

Example / Description

**Ambiguity**

“Users receive alerts” — What kind of alerts? How are they prioritized?

“App shows shelters nearby” — What’s the data source? What distance range?

**Inconsistency**

Requirement says users can edit profile, but no mention of where that data is stored or accessed

Requirement says app will send SMS alerts, but no SMS gateway or service is mentioned

**Missing Info**

No mention of how the app handles no internet connectivity during disaster

No indication of how user feedback or report feature is handled post-disaster

**3. Requirement Prioritization**

Use MoSCoW or a simple scale:

Must have: Emergency alerts, location tracking, contact emergency services

Should have: Disaster preparation tips, real-time map

Could have: Chat with volunteers, donation features

Won’t have (for now): AI prediction features

**4. Classification of Requirements**

**Functional Requirements**

**FR1: User Registration/Login**

Users must be able to register and log in using email or phone number.

**FR2: Real-Time Alerts**

Users receive alerts based on location and disaster type in particular area.

**FR3: Report Disaster**

Users can submit reports including photos, video, location, and type of disaster.

**FR4: Admin Dashboard**

Admins can view reports, approve or verify incidents, and push alerts.

#### FR5: First Responder Interface

Displays verified incidents with directions.

#### FR6: Emergency Contacts and Safety Tips

Static pages with emergency numbers and disaster safety info.

#### FR7: Map Integration

Displays disaster zones and reported incidents.

#### FR8: Disaster History Logs

Store and retrieve previous incidents with date, time, and location.

**FR9:** **Community Helep**

Request community help

**Non-Functional Requirements**

#### NFR1: Performance

Must support at least 10,000 concurrent users.

#### NFR2: Security

Data must be encrypted and user authentication is required.

#### NFR3: Availability

99.5% uptime with support for cloud hosting and backups.

#### NFR4: Usability

Intuitive UI/UX with accessibility support.

#### NFR5: Scalability

Ability to scale up to multiple regions and disaster types.