Department of Computer Science & Engineering

Software Project Report

</> Hook:< A SOS app that alerts nearby users and contacts to respond instantly during emergencies. >

Course Title : Software Engineering
Course Code : CSE412
Section : 04
Group : 03
Semester : Summer 2025

Submitted To:

Yasin Sazid

Lecturer

Department of Computer Science & Engineering

East West University.

Submitted By:

Name	ID	Role	Contributions%
Abrar Khatib Lajim	2022-3-60-043	Documentation	33%
Md. Saiful Islam	2022-3-60-045	Implementation	33%
Umme Mukaddisa	2022-3-60-317	Planning	34%

Signature of Team Members

Abrar Khatib Lajim

Md. Saiful Islam

Umme Mukaddisa

Letter of Transmittal

To,

Yasin Sazid

Lecturer

Department of Computer Science & Engineering

East West University

Subject: Submission of Project Report on *"ResQMob: Smart Emergency Response & Community Safety System"*

Dear Sir,

With due respect, we would like to submit our project report on **"ResQMob: Smart Emergency Response & Community Safety System."** This report outlines our efforts in developing an intelligent, scalable emergency alert and response system that empowers users and communities to respond effectively in critical situations.

Throughout the project, we have diligently documented each phase from problem identification and requirement analysis to system design, implementation, and testing. We have integrated key features like real-time SOS, location sharing, predictive alerts, and community engagement functionalities to address real-world safety challenges, particularly within the context of Bangladesh.

We sincerely hope our work meets your expectations and contributes to a better understanding of emergency management through technology. Furthermore, we kindly request you to review our report and accept it as a part of our course submission. Please excuse any unintended errors, and your valuable feedback would be greatly appreciated.

Sincerely,

On behalf of the Group - 03

Abrar Khatib Lajim

2022-3-60-043

Department of Computer Science & Engineering

Acknowledgement

We would like to express our sincere gratitude to everyone who contributed to the successful completion of our project, **"ResQMob: Smart Emergency Response & Community Safety System."**

We are especially thankful to our course instructor, **Yasin Sazid**, for his continuous support, valuable feedback, and expert guidance throughout the project. His mentorship helped us stay focused and refine the system's direction.

We also appreciate the efforts of our dedicated team members for their collaboration, creative thinking, and technical contributions.

A special thanks goes to the communities and individuals who inspired this idea and provided insights into real-world emergency challenges in Bangladesh, helping us design a solution that is both practical and impactful.

Abstract

ResQMob is a smart, mobile-first emergency response and community safety application developed to address real-world safety challenges in urban environments like Dhaka.

It allows users to send instant SOS alerts, share real-time locations, notify emergency contacts, and engage nearby responders during critical situations. The app features an interactive map, risk prediction using machine learning, and community-based alert verification.

Built using **Flutter** and **Supabase/Firebase**, and developed following the **Agile methodology**, **ResQMob** empowers users and communities to act swiftly, reduce emergency response time, and enhance public safety.

Its scalable architecture and predictive capabilities also support smarter, data-driven decision-making for future safety planning.

Chapter 1. Introduction

1.1 Project Overview (as a User Story)

As a user, I want to send SOS alerts with one tap, share my real-time location, and notify emergency contacts and nearby community members so that I can receive help quickly and safely.

ResQMob is a real-time emergency response application designed to improve public safety and community-based support in Bangladesh. It provides location-aware SOS triggering, community alerts, emergency communication, and predictive analytics to reduce emergency response time and improve safety awareness.

1.2 Purpose and Scope

Purpose: ***ResQMob*** is a crowd-assisted emergency response mobile application that enables users to send SOS alerts, notify nearby helpers, and receive real-time assistance. It also features a community feed for safety updates and an ML-powered risk prediction system.

Scope Includes:

1. One-tap SOS alert system with emergency type selection (manual & auto-trigger)
2. Real-time location sharing
3. Real-time responder tracking
4. Interactive map with live emergency alerts and responder visibility
5. Safe zone navigation
6. Community safety tips and prediction modules (ML-based)
7. Admin tools for monitoring and evaluation
8. Admin monitoring & control
9. Advanced features like gesture activation, voice clips, image

1.3 Stakeholders

Primary Stakeholders:

1. End Users (alert senders, responders)
2. Emergency Responders (police, medical teams)
3. Admins (control & monitoring)

Secondary Stakeholders:

1. Developers & Designers
2. System Administrators
3. Data Analysts (ML model)
4. NGOs & Disaster Management Authorities

1.4 Technology Stack

Layer	Technology
Programming Language	**Dart** (Flutter framework) – for cross-platform mobile app (Android & iOS)
Frontend Framework	**Flutter** – UI toolkit for building interactive mobile app UI
Backend (Functions)	**Firebase Cloud Functions** (Node.js/TypeScript) for serverless backend logic (e.g., sending SOS SMS)

Database	**Firebase Firestore** (NoSQL, real-time database)
Authentication	Supabase Auth (JWT-based), Firebase Auth (optional hybrid)
Storage	Firebase Storage / Supabase Storage – for profile images & files
Cloud Messaging	Firebase Cloud Messaging (FCM) – for push notifications
APIs (External)	**HTTP** (via Dart http package) – to call SMS service (**Amar Sheba SMS** via Cloud Function)
Mapping / Location	Google Maps API / Geolocation Plugin – for user location & geofencing
Version Control	Git + GitHub/GitLab
CI/CD	GitHub Actions / Codemagic (Flutter CI/CD pipeline)

Chapter 2. Software Requirements Specification & Analysis

2.1 Stakeholder Needs & Analysis

➔ **Primary:** Citizens, Emergency Responders, Admins

➔ **Secondary:** Developers, NGOs, Law Enforcement, Hospitals

Primary Stakeholder Needs:

1. Quick SOS activation
2. Accurate location sharing
3. Real-time help tracking
4. Community-based safety updates

Secondary Stakeholder Needs:

1. Scalable backend
2. Secure data storage
3. ML-powered predictions
4. Compliance with safety regulations

Requirement Elicitation Methods

1. **Stakeholder Interviews:** Conducted interviews with friends and responders to gather real-world emergency needs.

Questions:

High-level (Contextual / General):

1. "When you or someone you know faced an emergency in Dhaka, what were the biggest challenges in getting quick help?"

2. “What role do you think mobile technology can play in making urban areas like Dhaka safer during emergencies?”

Medium-level (Behavioral / Need-based):

1. “If you could instantly send an SOS alert with your location to emergency services and nearby people, how valuable would that be for you?”
2. “How would you use a heatmap that shows risky areas in the city (e.g., avoiding unsafe routes, planning travel)?”
3. “During an emergency, would you prefer a very quick one-tap SOS alert, or the option to select a specific type of emergency like fire, medical, or police?”

Low-level (Specific / Functional):

1. “Which would you find more reliable in a crisis: triggering SOS by pressing the power button/shaking the phone, or opening the app?”
2. “If you needed to share extra details in an emergency, would you rather send a voice clip, upload a screenshot, or just rely on live location?”
3. “For responders handling many alerts at once, do you think automated severity levels and updates would be more helpful than waiting for manual updates from users?”

Answers:

Nusroth Nourin
ID: 2022-3-60-00

1. The biggest challenges people face when trying to get quick help during emergencies in cities like Dhaka is traffic jams because people can’t make quick help for medical service & also to reach the destination becomes hard.
2. With mobile technology instant SOS, awareness can be sent with live location to the responder.
3. It would be very helpful & very valuable if there’s any SOS system.

4. Quick tap SOS alert will be good.
5. Using the power button or shaking phone is reliable in emergency situations.
6. For extra details, I prefer voice clip
7. I think automated severity levels & updates would be more helpful.

Rawan Hassan
ID: 2022-3-60-174

1. When i'm in danger maybe for medical issue the biggest challenge for me in Dhaka city is traffic jam because of this i will fail in getting help from my people.
2. I think mobile technology will play an important role because nowadays everyone use mobile phones.
3. Sending an SOS alert with your location to emergency services & nearby people I think it will be a very valuable thing for Dhaka's recent situations.
4. I don't know but a heatmap concept will be really helpful for safe routing.
5. I prefer a quick one tap SOS alert.
6. I think pressing the power button will do good.
7. I think to share extra details, live location will be more efficient.
8. I prefer automated severity levels & updates.

Urban Emergency Research: Studied communication issues in Dhaka's emergency response systems.

Test User Feedback: Gathered insights from early prototype users to refine core features.

Competitive Analysis: Reviewed existing apps to identify gaps and opportunities for improvement.

Use Case Development: Created scenarios to define system behavior in various emergencies.

2.2 List of Requirements

Functional Requirements (FRs)

1. User authentication & role management
2. Send emergency alerts with location
3. Auto emergency post creation in community feed
4. Real-time responder tracking
5. Emergency type selection (Fire, Police, Medical, General)
6. Gesture-based SOS trigger
7. ML-based risk prediction and heatmap
8. Safe zone navigation
9. Admin dashboard with severity mapping
10. Voice clip & screenshot upload to admin

Non-Functional Requirements (NFRs)

1. **Platform:** The app must be responsive and support Android & iOS
2. **Security:** Encrypted communication & data storage
3. **Performance:** SOS broadcast in < 3 seconds
4. **Availability:** 99% uptime
5. **Scalability:** Handle 10k concurrent alerts
6. **Usability:** UI designed for quick actions (< 2 taps)

Extra-Ordinary Requirements (Wow Factors)

1. Predictive model to warn about high-risk areas in advance
2. Anonymous SOS triggering via button shortcut
3. ML-based Safe Road Routing
4. Auto-post situation updates in feed
5. Full phone monitoring for the admin during critical alerts

2.3 Quality Function Deployment

Table: Customer Requirements (CRs)

CR Code	Description
CR1	Users need a fast login process
CR2	The system should handle high traffic
CR3	Users want a mobile-friendly UI
CR4	Ensure data security
CR5	Provide real-time notifications

Table: Technical Requirements (TRs)

TR Code	Description
TR1	Implement OAuth-based login
TR2	Use a scalable cloud database
TR3	Responsive web/app design
TR4	Encrypt user data
TR5	Implement WebSocket for real-time alerts

Table: QFD Matrix (House of Quality)

Customer Requirements (CRs)	**Technical Requirements (TRs)**	**Relationship**
CR1: Fast login process	TR1: OAuth-based login	Strong
CR2: Handle high traffic	TR2: Scalable cloud database	Strong
CR3: Mobile-friendly UI	TR3: Responsive design	Medium
CR4: Ensure data security	TR4: Encrypt user data	Strong
CR5: Real-time notifications	TR5: WebSocket for alerts	Medium

Customer Requirements (CRs) List:

1. Quick emergency activation
2. Accurate location tracking
3. Real-time help updates
4. Community safety sharing
5. Predictive risk alerts
6. Advanced monitoring features
7. Ease of use in stress situations

Technical Requirements (TRs) List:

A. High-accuracy GPS integration

B. Low-latency notification system

C. Scalable backend architecture

D. Secure database & encryption

E. ML model for prediction

F. Map & navigation API integration

G. Button Trigger system

H. Media upload system

(●●● = Strong, ●● = Medium, ● = Weak relationship)

CR \ TR	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
1. Quick emergency activation	●●●	●●●	●●	●●	●	●	●●●	●
2. Accurate location tracking	●●●	●●	●	●	●	●●●	●	
3. Real-time help updates	●●	●●●	●●	●	●	●●	●	●
4. Community safety sharing	●	●●	●●●	●●	●	●		●●●
5. Predictive risk alerts	●	●	●●	●	●●●	●		
6. Advanced monitoring	●	●●	●●	●●●	●●	●		●●
7. Ease of use in stress	●	●	●	●		●	●●●	

2.4 Requirements Modeling

2.4.1 Table 0: Use Cases Overview

Level	Focus	Happens
0	High-Level Goal	User opens app, system initializes session, and prepares all safety features.
1	Main Interactions	Authentication, sending emergency alerts, receiving alerts, viewing community feed, risk prediction, admin monitoring.
2	Sub-Functions	Emergency type selection,location sharing, contact notifications, post management, ML-based risk prediction, safe zone display, responder count, severity escalation, and advanced monitoring.
3	Detailed Flows	Step-by-step actions, input/output handling, decision points, alternative flows, and error handling for all main and sub use cases.

2.4.1.1 Level-Wise Use Case Table for ResQMob

Table 1: Level 0 – High-Level Goal

Level	Use Case Name	Actor(s)	Description
0	Open the App	User	User opens and system initializes session & location access.

Table 2: Level 1 – Primary Interactions

Use Case	Actors Involved	Description
Authentication	User, Admin	Users register, log in, and securely access the system.
Send Emergency Alert	User, System	User sends SOS alert with location and emergency details.
Receive Alert Notification	System, Nearby Users, Emergency Contacts	System broadcasts alerts to responders and emergency contacts.
View Community Feed	User, System	Users post updates, react, comment, and see temporary alert posts.
View Risk Prediction	User, System (ML Module)	User views predicted high-risk zones and safe routes.
Admin Monitor Alerts	Admin, System	Admin monitors active alerts, severity levels, and responder data.

Table 3: Level 2 – Sub-Actions of Primary Use Cases

Main Use Case	Sub Use Cases	Actors	Description
Authentication	1. Registration 2. Login 3. Logout 4. Password Reset	User, Admin	Users create accounts, log in securely, log out, and reset passwords via email.
Send Emergency Alert	1. Select Emergency Type 2. Auto-Trigger SOS 3. Share Location	User, System	Users manually or automatically trigger SOS with type and location.
Receive Alert Notification	1. Notify Contacts 2. Show Map Marker 3. Provide Navigation	System, Nearby Users, Emergency Contacts	Alerts sent to contacts, map marker shown, navigation option provided.
View Community Feed	1. Create Post, 2. React to Post 3. Comment on Post 4. Auto Emergency Post	User, System	Users interact with feed; emergency alerts appear as temporary pinned posts.
View Risk Prediction	1. Predict Risk from Data 2. Show Safe Zones	User, ML Module	ML predicts risk zones and displays safe locations/routes.
Admin Monitor Alerts	1. View Active Alerts 2. View Responder Locations 3. Access Media	Admin, System	Admin sees live alerts, responders, and receives voice/screenshot/image updates.

Table 4: Level 3 – Detailed Flows

Parent Use Case	Child Use Cases	Actors	Description
Authentication	1. Two-Factor Authentication (2FA) 2. Role-Based Access Control 3. Session Management	User, Admin, System	Login with OTP or password; admin manages roles; auto logout after inactivity.
Send Emergency Alert	1. Tap Help Button 2. Gesture Activation (opt.) 3. Select Type 4. Increase Severity (opt.) 5. Share Location	User, System	User triggers alert via button/gesture, selects type, severity increases with repeated presses, location shared in real time.
Receive Alert Notification	1. Determine Proximity 2. Send Push/SMS 3. Show Map Marker 4. Navigate	System, Responders	System checks user location, sends alerts, displays marker, enables navigation.
View Community Feed	1. Create Post user 2. React user 3. Comment user 4. Auto Emergency Post	User, System	Users post and interact; on SOS trigger, system auto-creates top alert post with live updates.
View Risk Prediction	1. Fetch Model Data 2. Match with Location 3. Show Heatmap 4. Show Safe Zones	User, ML System	ML predicts high-risk areas and shows safe routes based on data.
Admin Monitor Alerts	1. Show Alerts with Severity Colors 2. Display Responder Count	Admin, System	Admin monitors severity, sees responder stats, and accesses multimedia updates.

Use Case Diagrams

Fig : Level 0-High-Level Goal

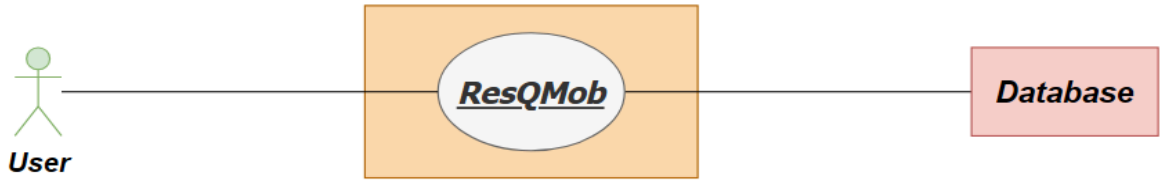


Fig : Level 1-**Primary Interactions**

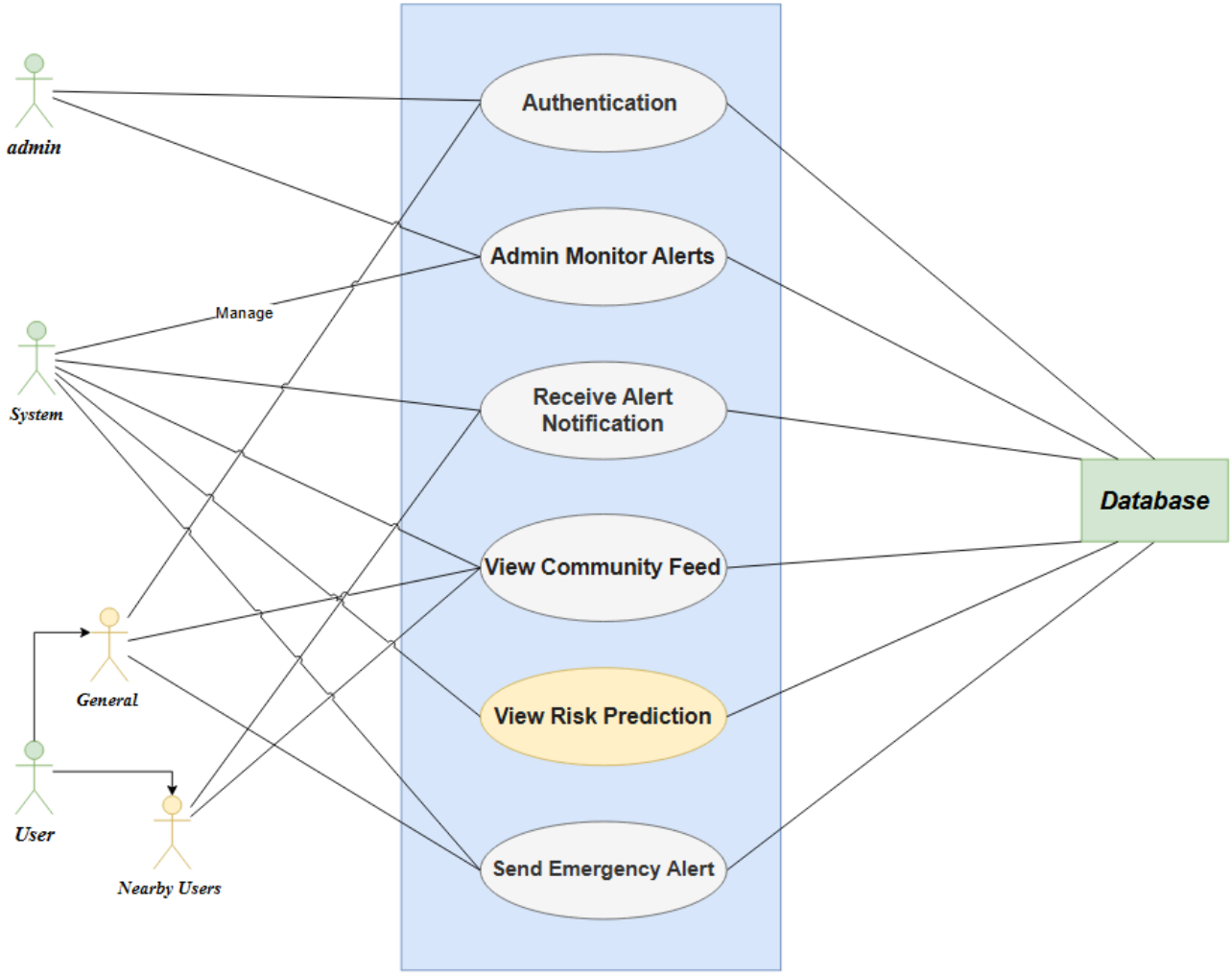
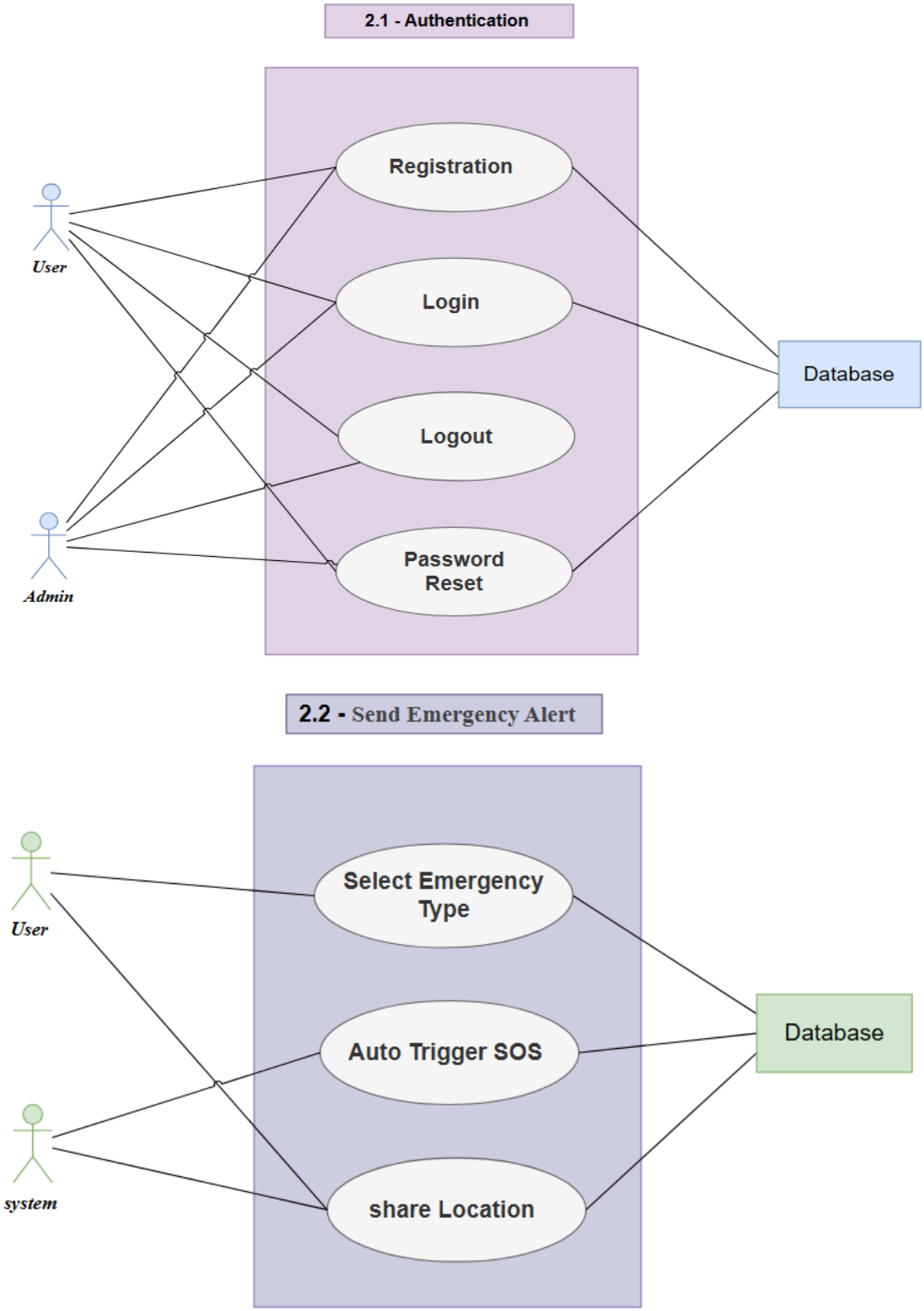
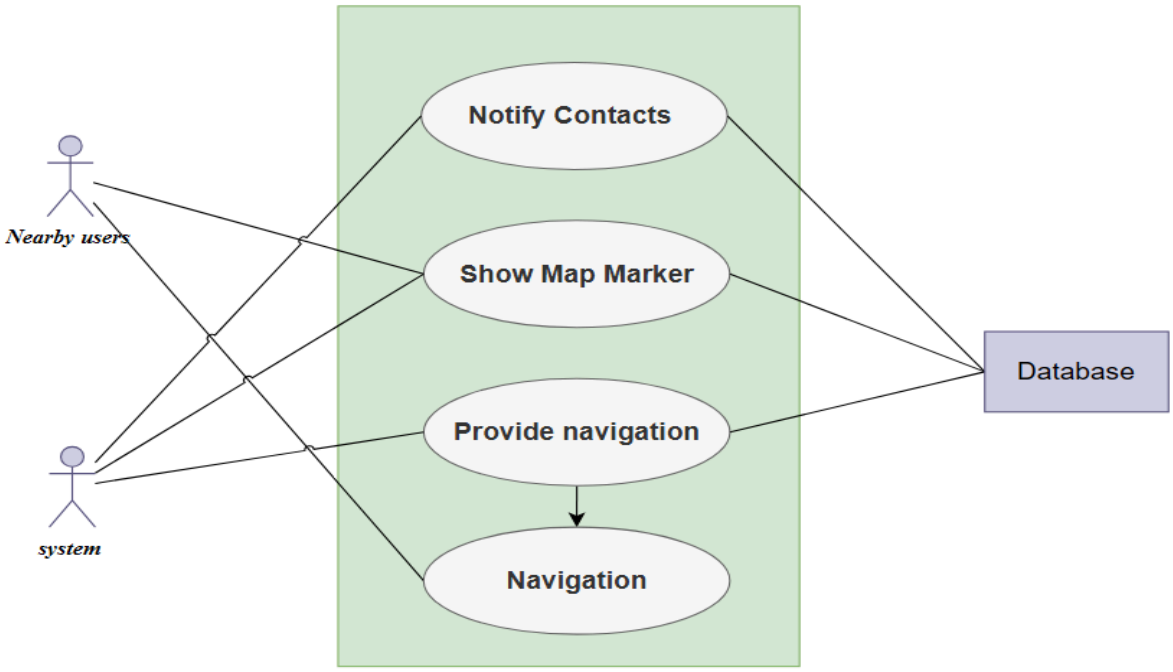


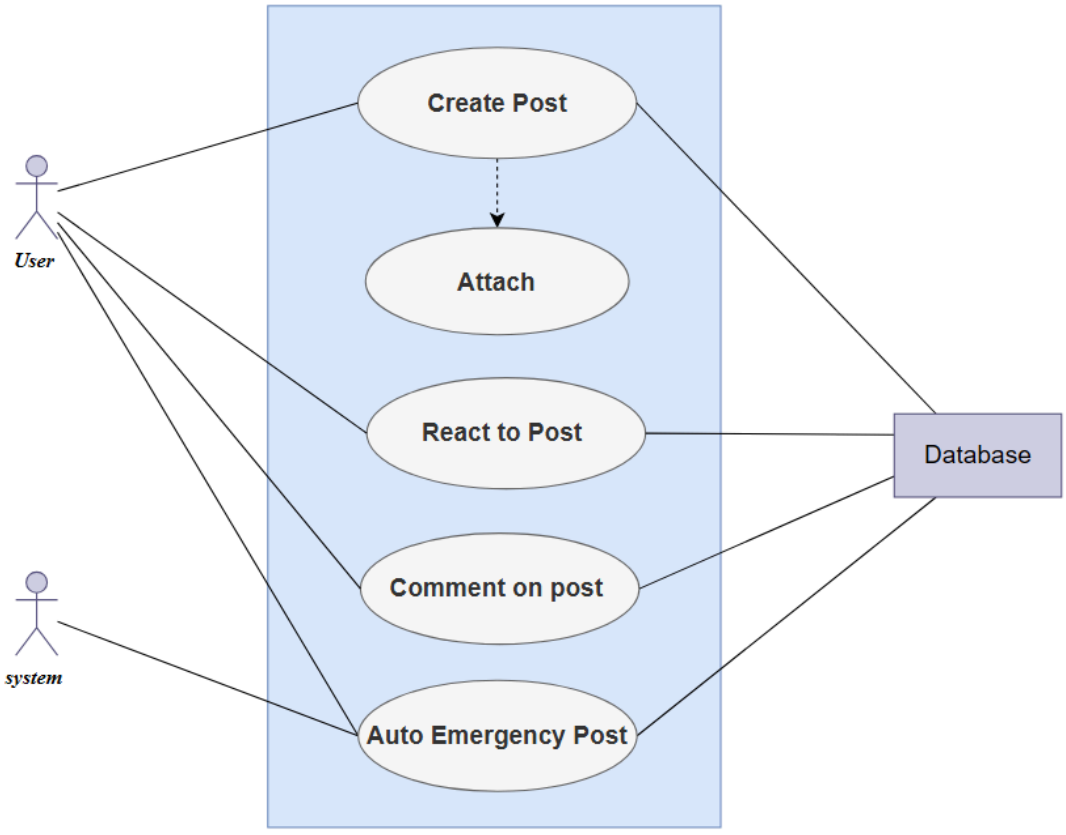
Fig : Level 2 –Sub-Actions of Primary Use Cases

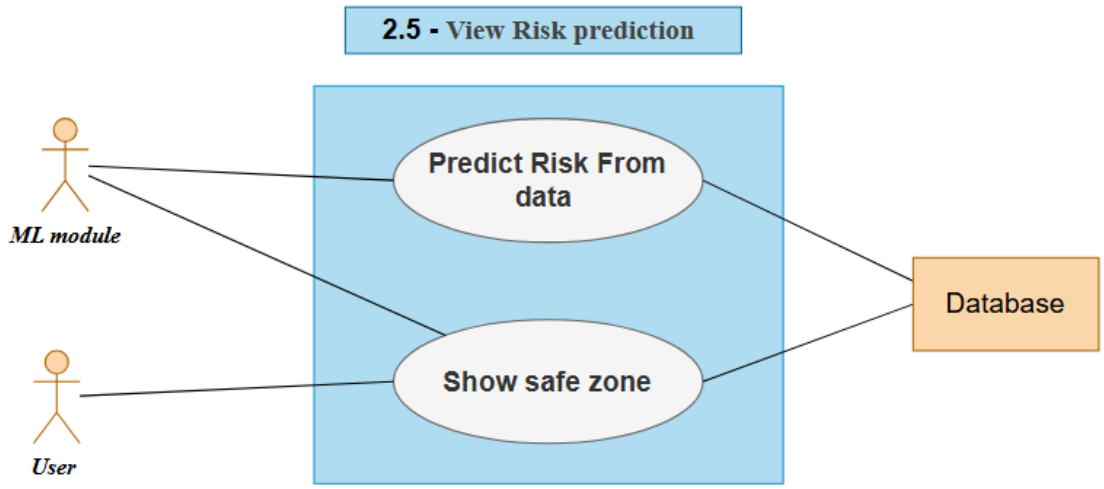


2.3 - Received Alert Notification



2.4 - View Community Feed




2.5 - View Risk prediction
ML module
User
Predict Risk From data
Show safe zone
Database

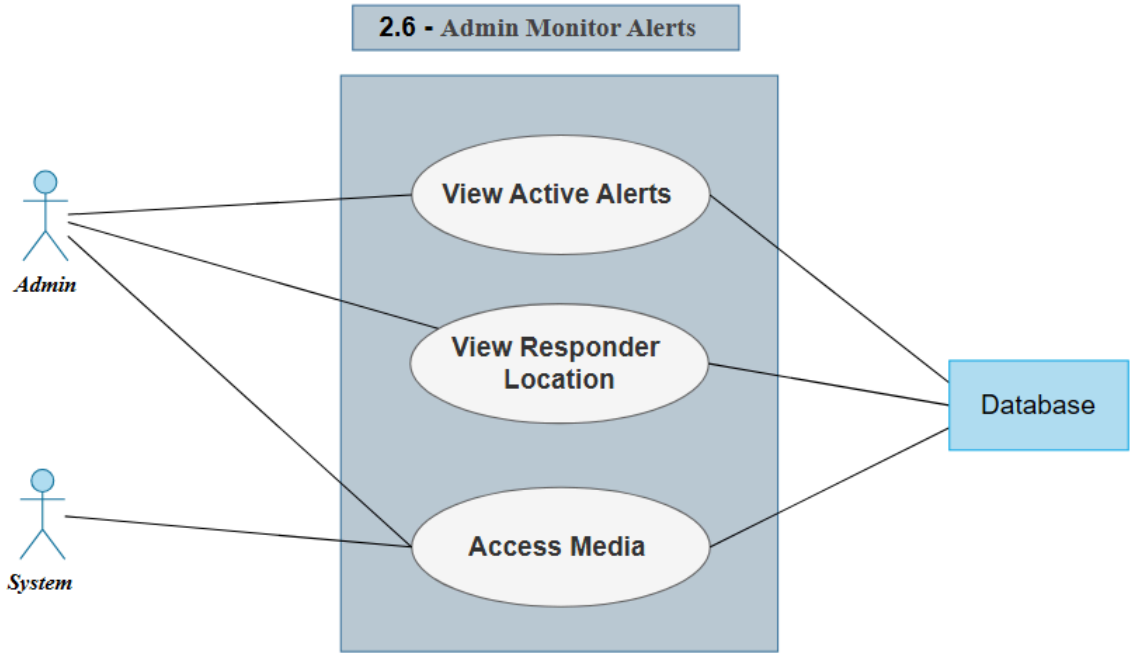
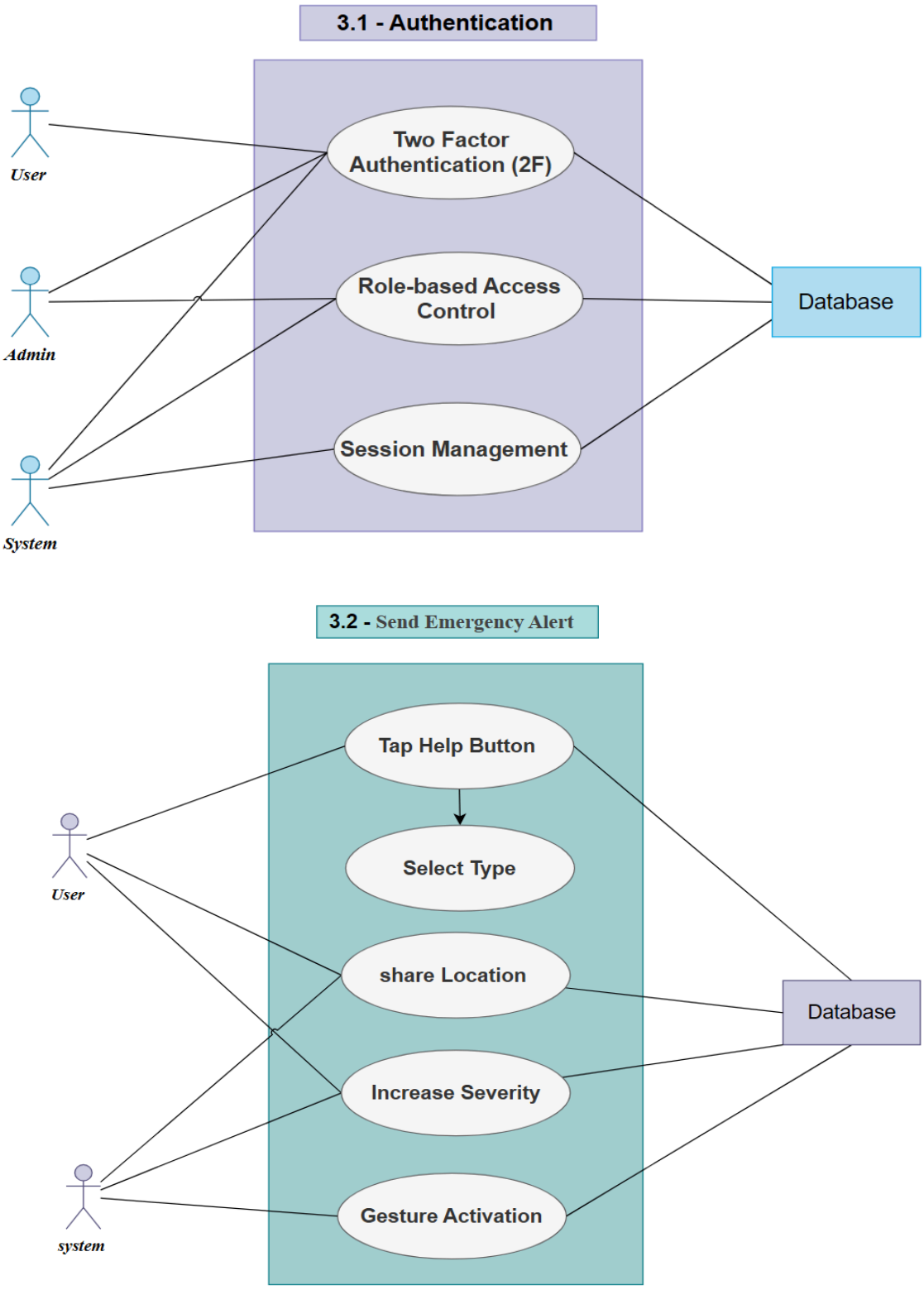
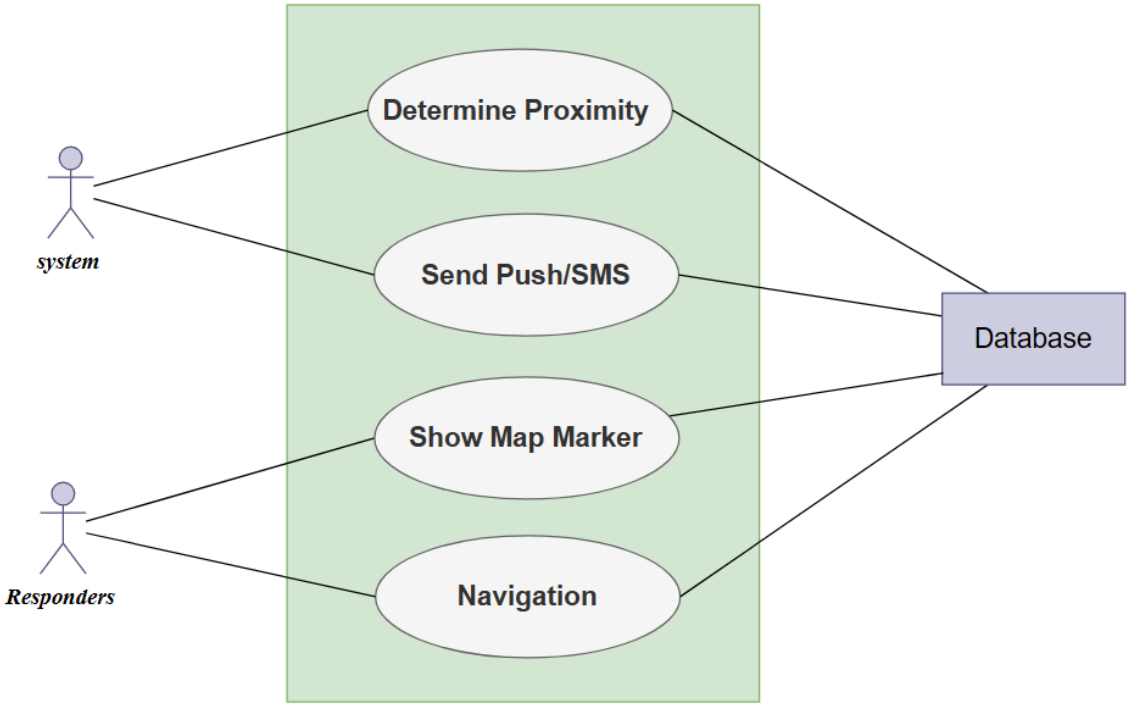

2.6 - Admin Monitor Alerts
Admin
System
View Active Alerts
View Responder Location
Access Media
Database

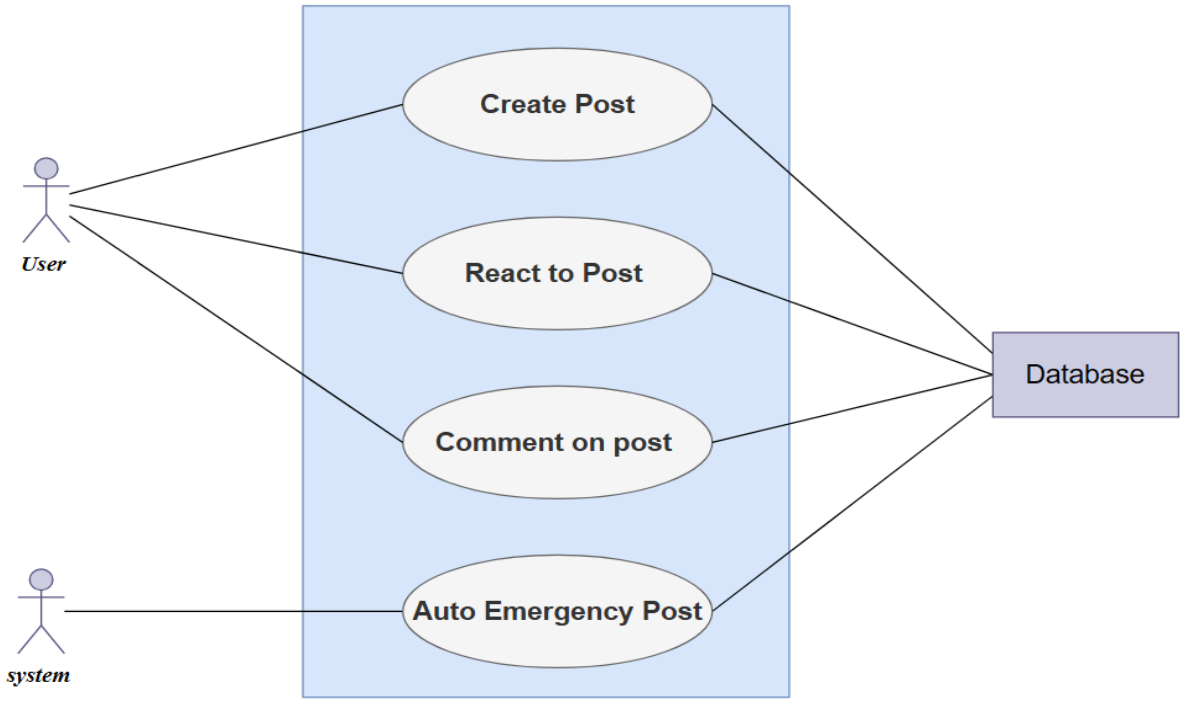
Fig : Level 3 – Detailed Flows



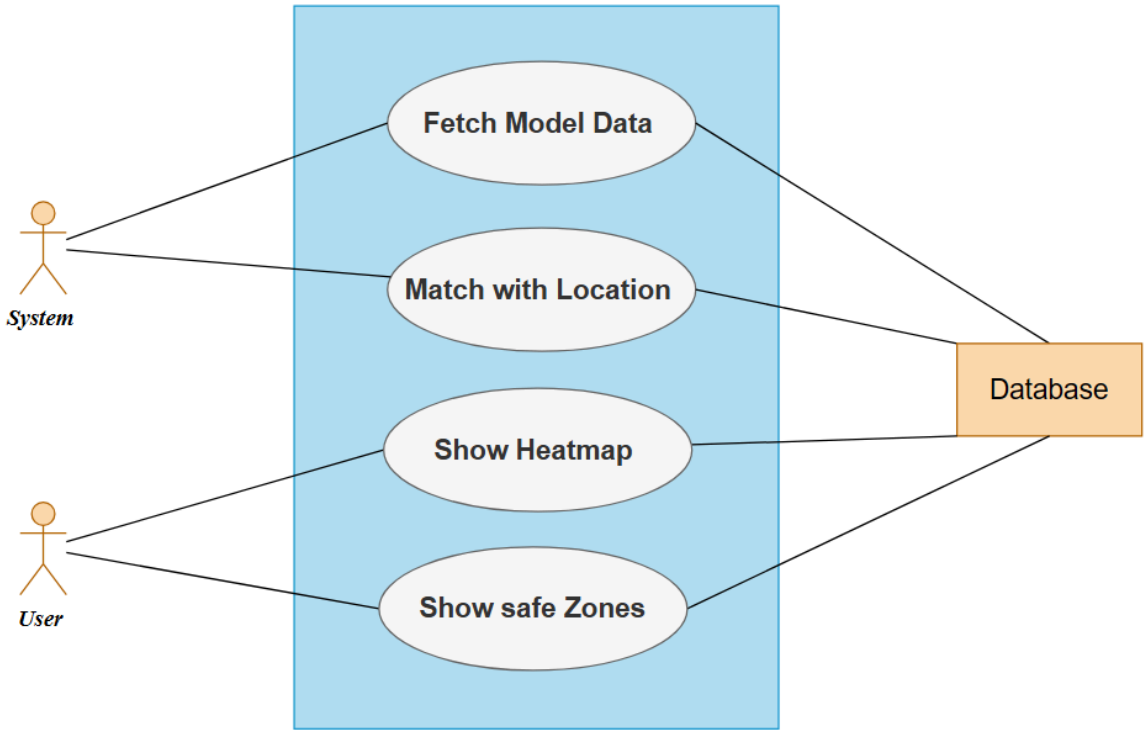
3.3 - Received Alert Notification



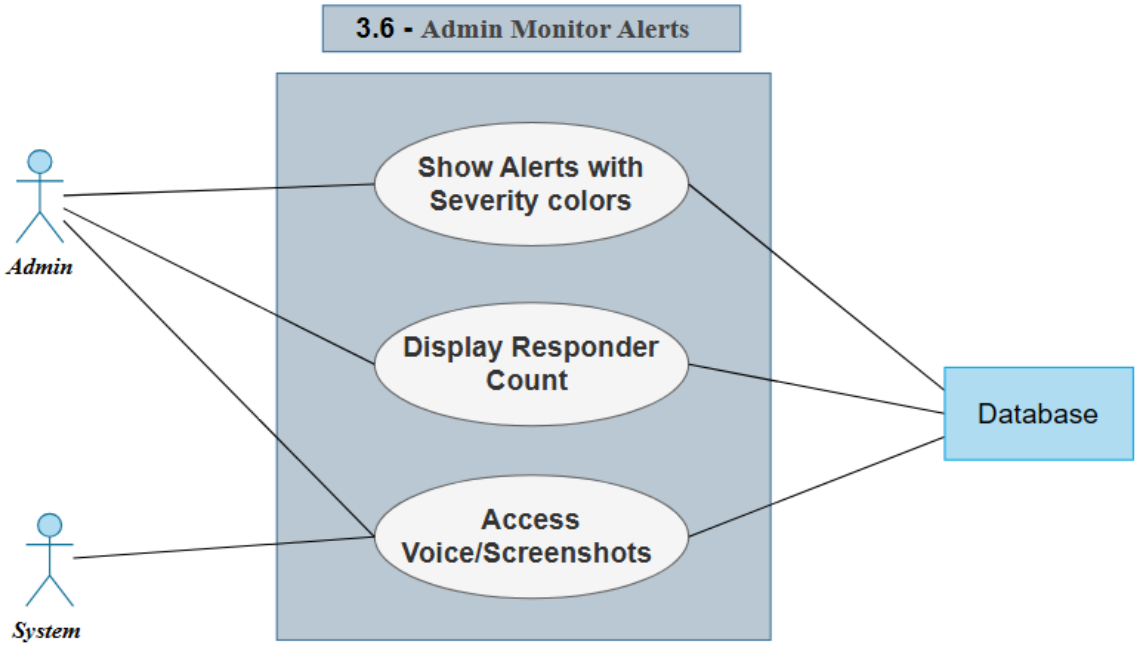
3.4 - View Community Feed



3.5 - View Risk prediction

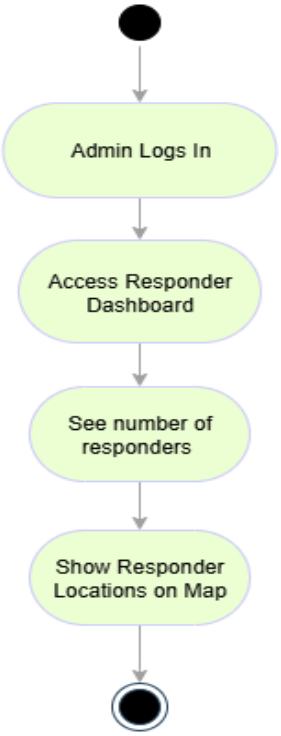


3.6 - Admin Monitor Alerts

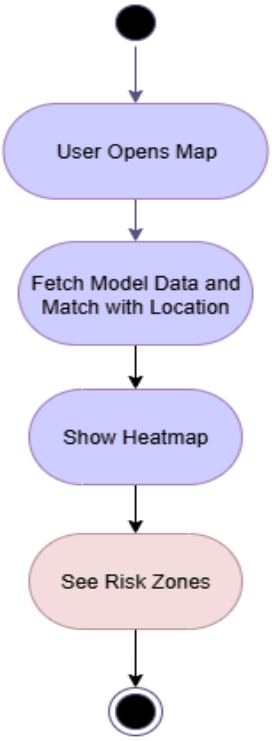


2.4.2 Activity Diagrams

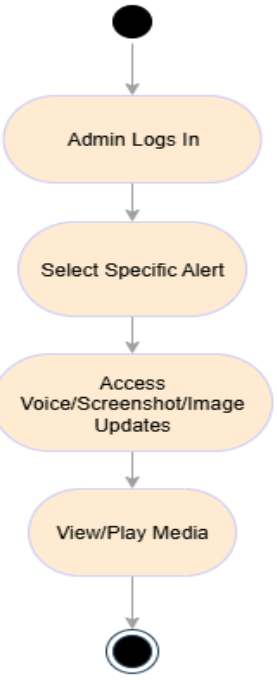
Activity Diagram for Admin monitor alert
(View Active Alerts)



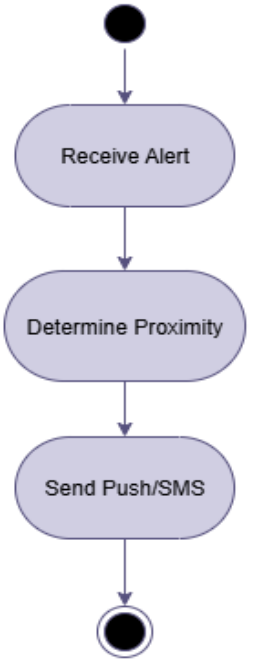
Activity Diagram for View Risk Predictions



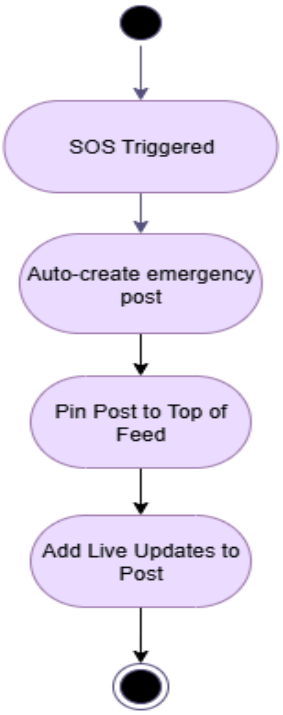
Activity Diagram for Admin monitor alert
(Access Media)



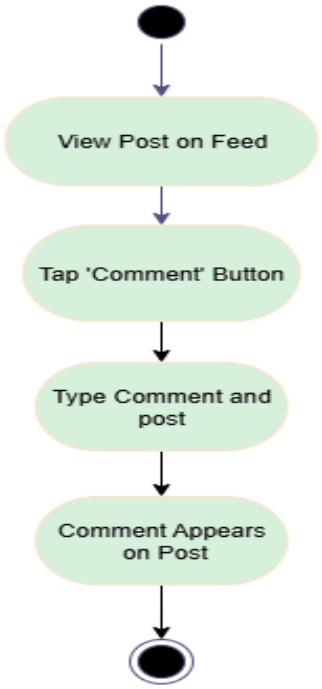
Activity Diagram for Receive alert notification
(Alerts sent to contacts)



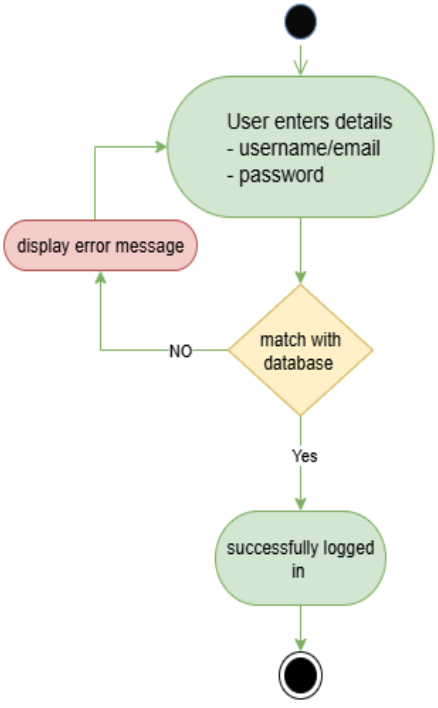
Activity Diagram for Community Feed
(Auto Emergency Post)



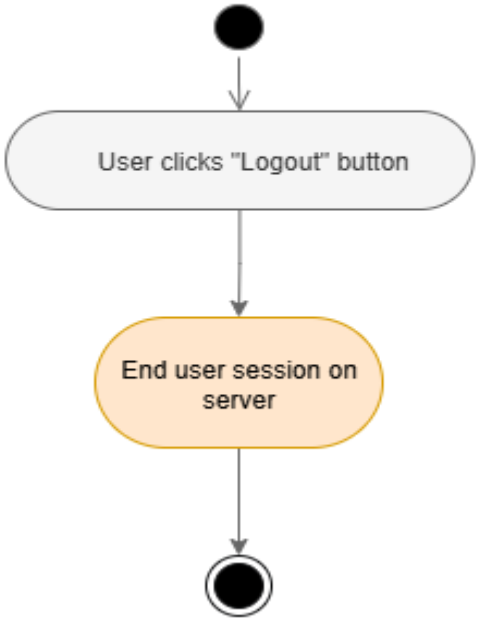
Activity Diagram for Community Feed
(comment to Post)



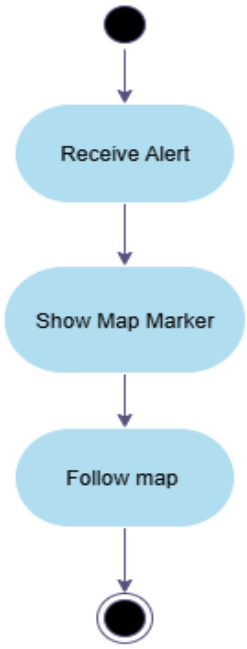
Activity Diagram for logging



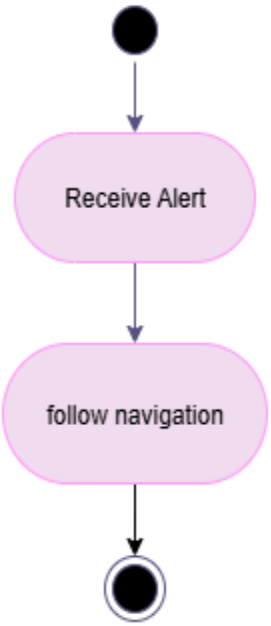
Activity Diagram for logout



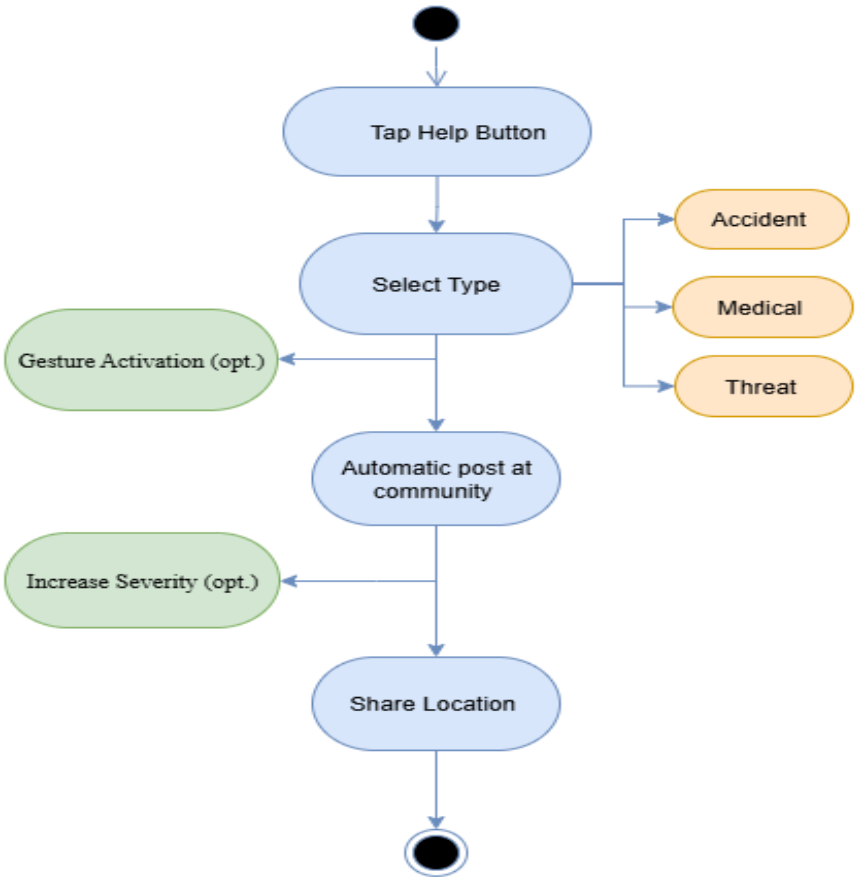
Activity Diagram for Receive alert notification
(Map marker shown)



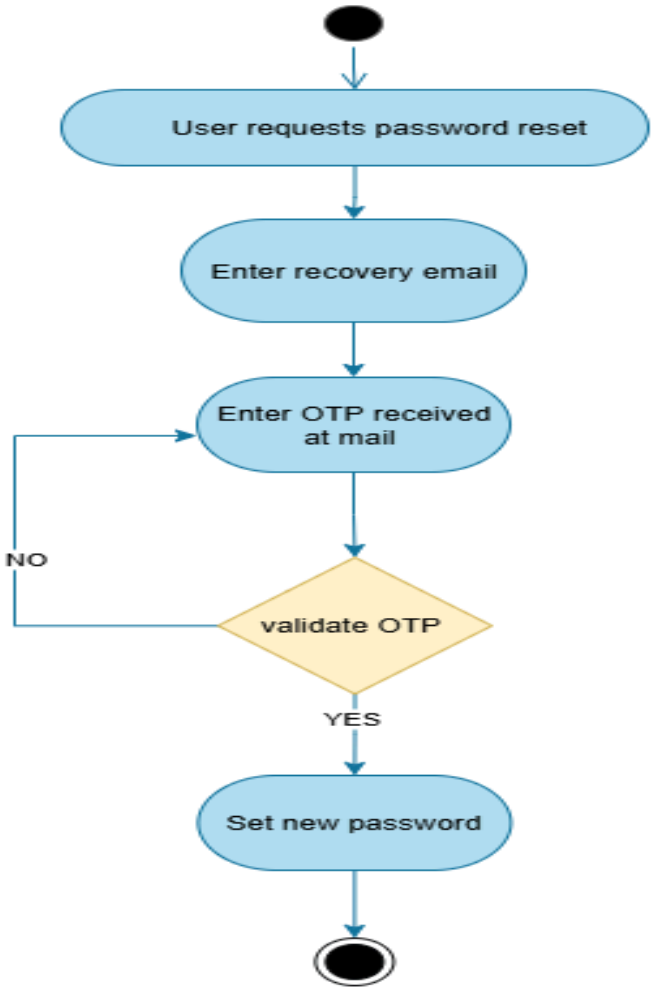
Activity Diagram for Receive alert notification
(Navigation option provided)



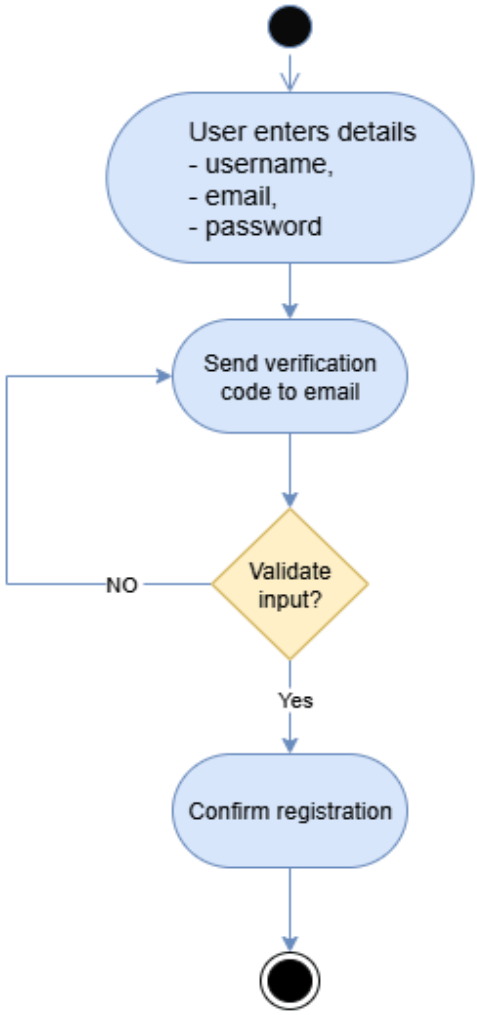
Activity Diagram for Send Emergency Alert



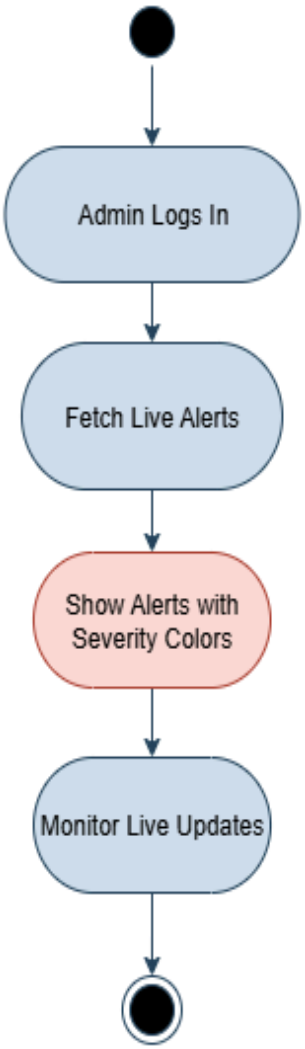
Activity Diagram for Password Reset



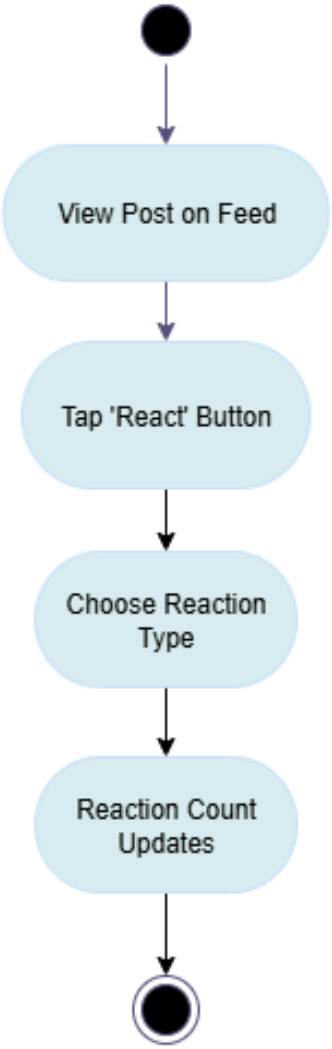
Activity Diagram for Registration



Activity Diagram for Admin monitor alert
(View Active Alerts)

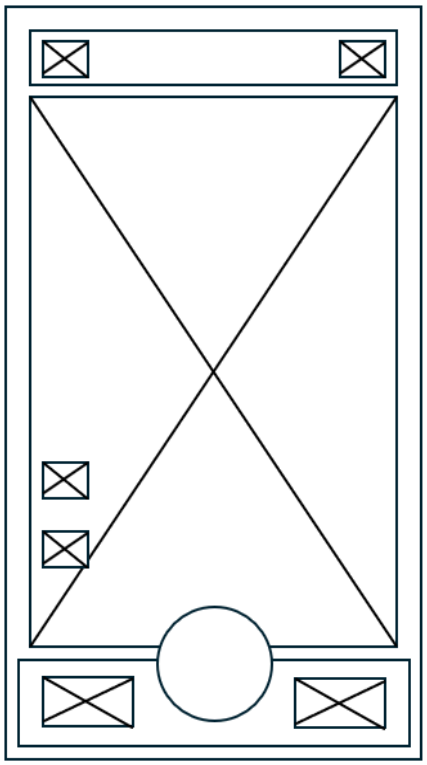


Activity Diagram for Community Feed
(Create Post)

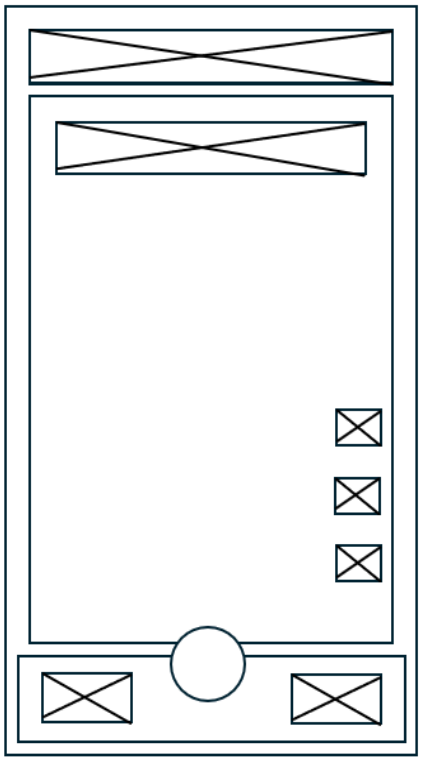


2.4.3 Prototyping (Wireframes / UI Sketches)

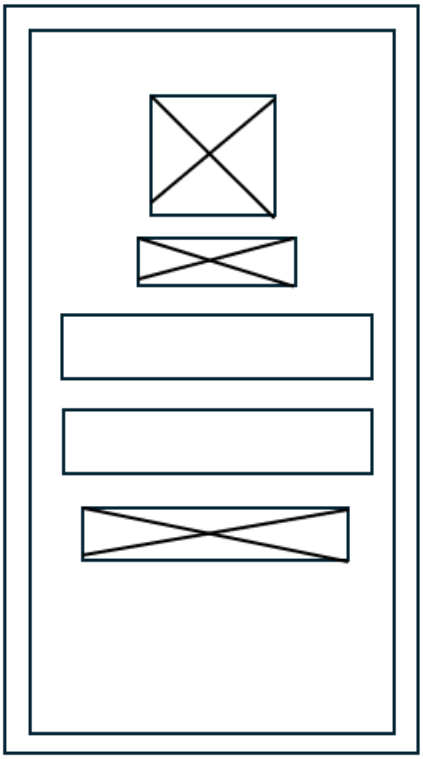
homescreen



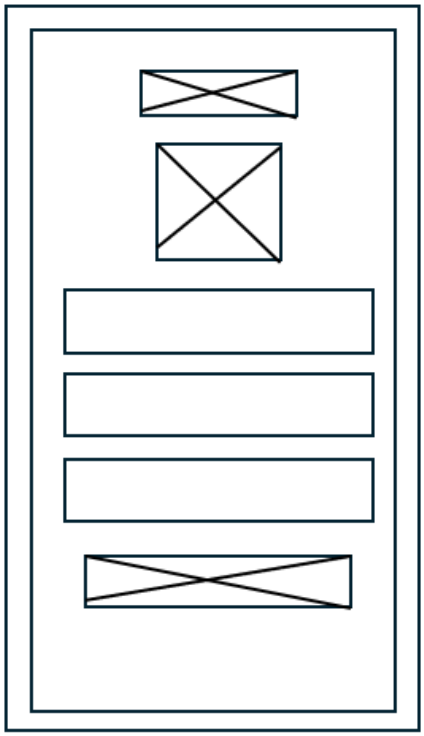
Safe map



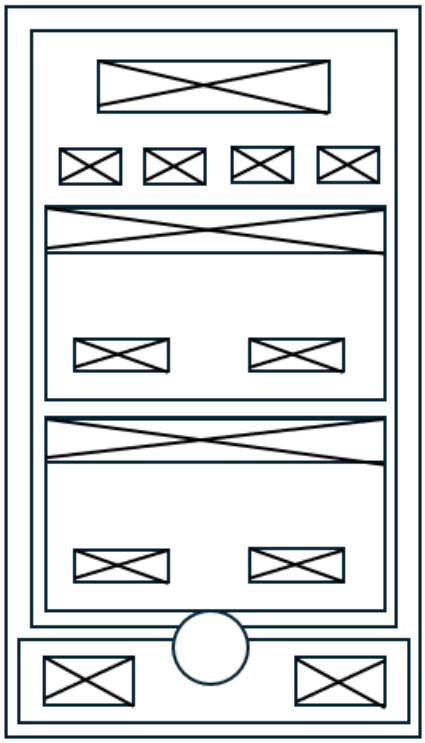
Login



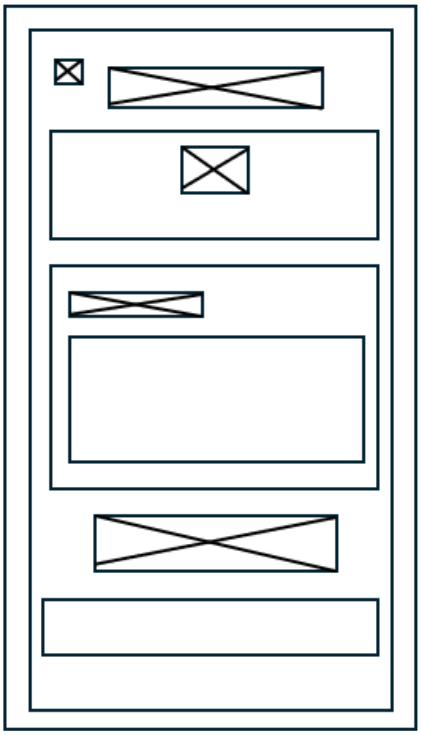
Sign Up



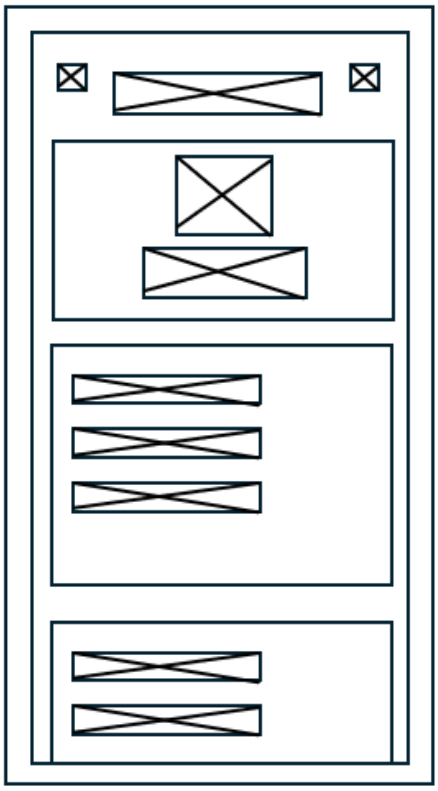
Active Alerts



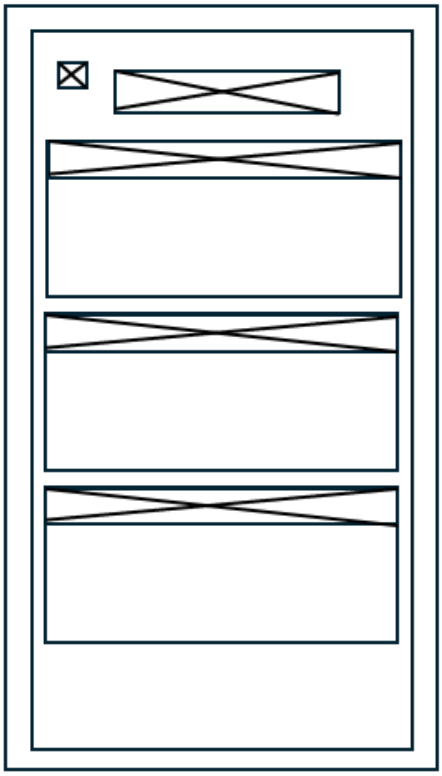
Send Feedback



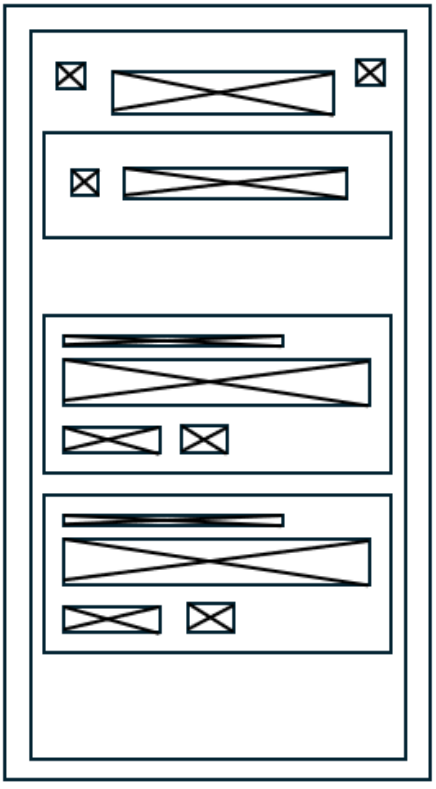
Profile



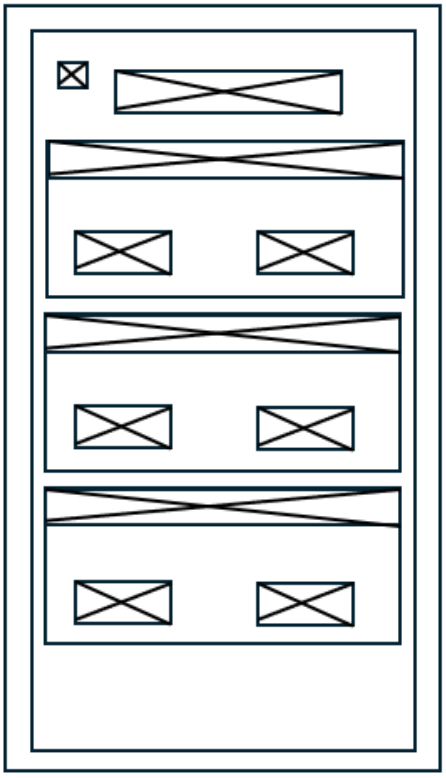
Alert History / Response



Community Feed



Nearby Station / Hospital



Chapter 3. Software Design

3.1 Architectural Design

3.1.1 Architectural Context Diagram

External Entities & Interactions:

- ➔ **User (Alert Sender / Responder)** → Uses mobile app to send/receive alerts
- ➔ **Admin** → Monitors active alerts, manages responders, accesses data
- ➔ **Emergency Contacts** → Receive SMS or push notifications during alerts
- ➔ **Nearby Users** → Receive alerts within range
- ➔ **ML Prediction Service** → Provides danger zone predictions
- ➔ **Maps API (Google Maps)** → Provides navigation & safe zone locations
- ➔ **Notification Service (Firebase Cloud Messaging)** → Sends push alerts
- ➔ **Database (PostgreSQL)** → Stores alerts, posts, user profiles
- ➔ **Community Feed Users** → Post, react, and comment in feed

Diagram Structure (in Pressman style)

Level	Description
Level 0	**System Context View** – Shows how ResQMob interacts with external entities like users, ML service, Maps API, Firebase, database, etc.
Level 1	**Top-Level Components** inside ResQMob – Main internal subsystems (auth, alert, feed, ML, etc.) and their interaction with external services.

Level 2	**Subsystem Decomposition** – Break down major components like "Emergency Alert Module" or "Community Feed" into more detailed components (e.g., Alert Triggering, Geo-Fencing, Feed Ranking, Comment System, etc.)
Level 3 (Optional)	**Module or Class Level Design** – Only if required. Elaborates internal structures of subcomponents with class diagrams or specific algorithms.

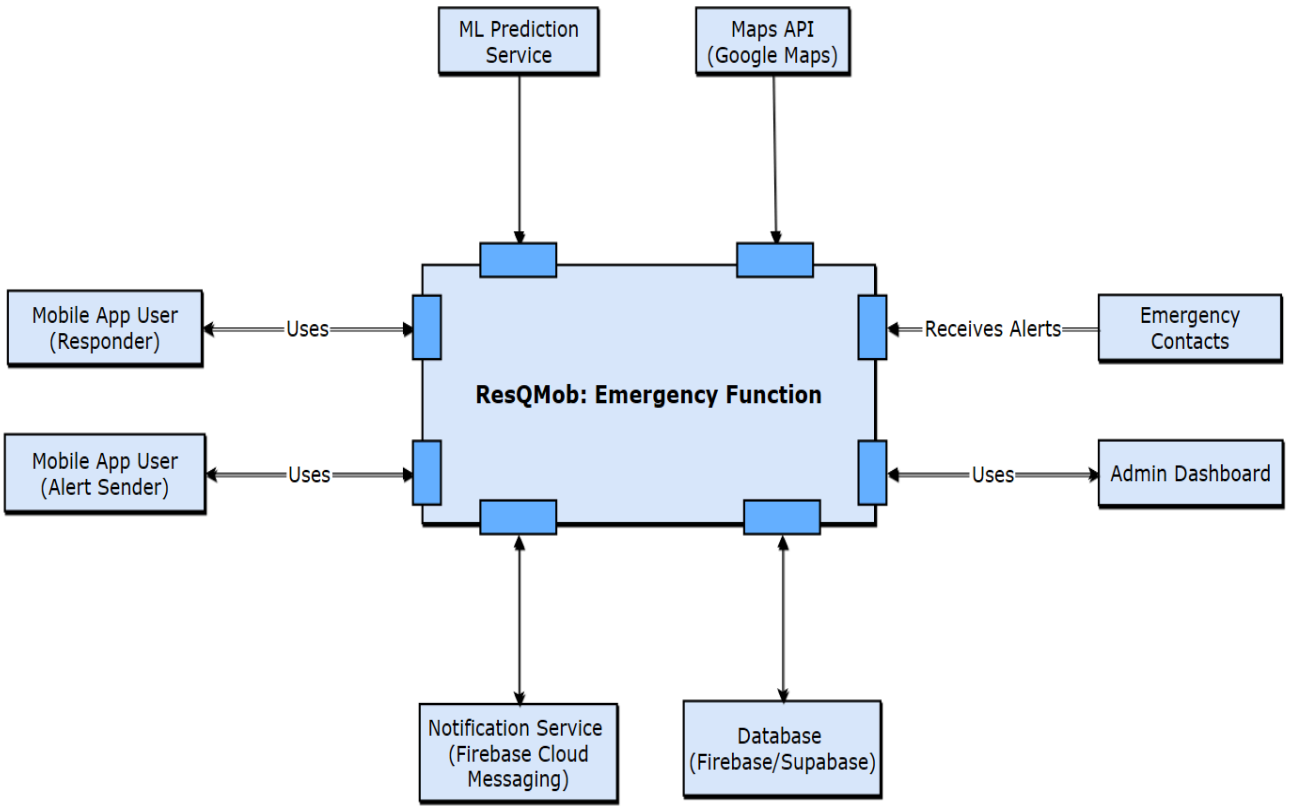
Level 0 (Context Diagram):

➔ **Actors:** User, Nearby Users, Admin, Emergency Contacts, External Services (Maps API, ML Model)

➔ **System:** ResQMob Mobile App & Backend

➔ **Data Flows:** SOS requests, notifications, location updates, feed updates, ML prediction

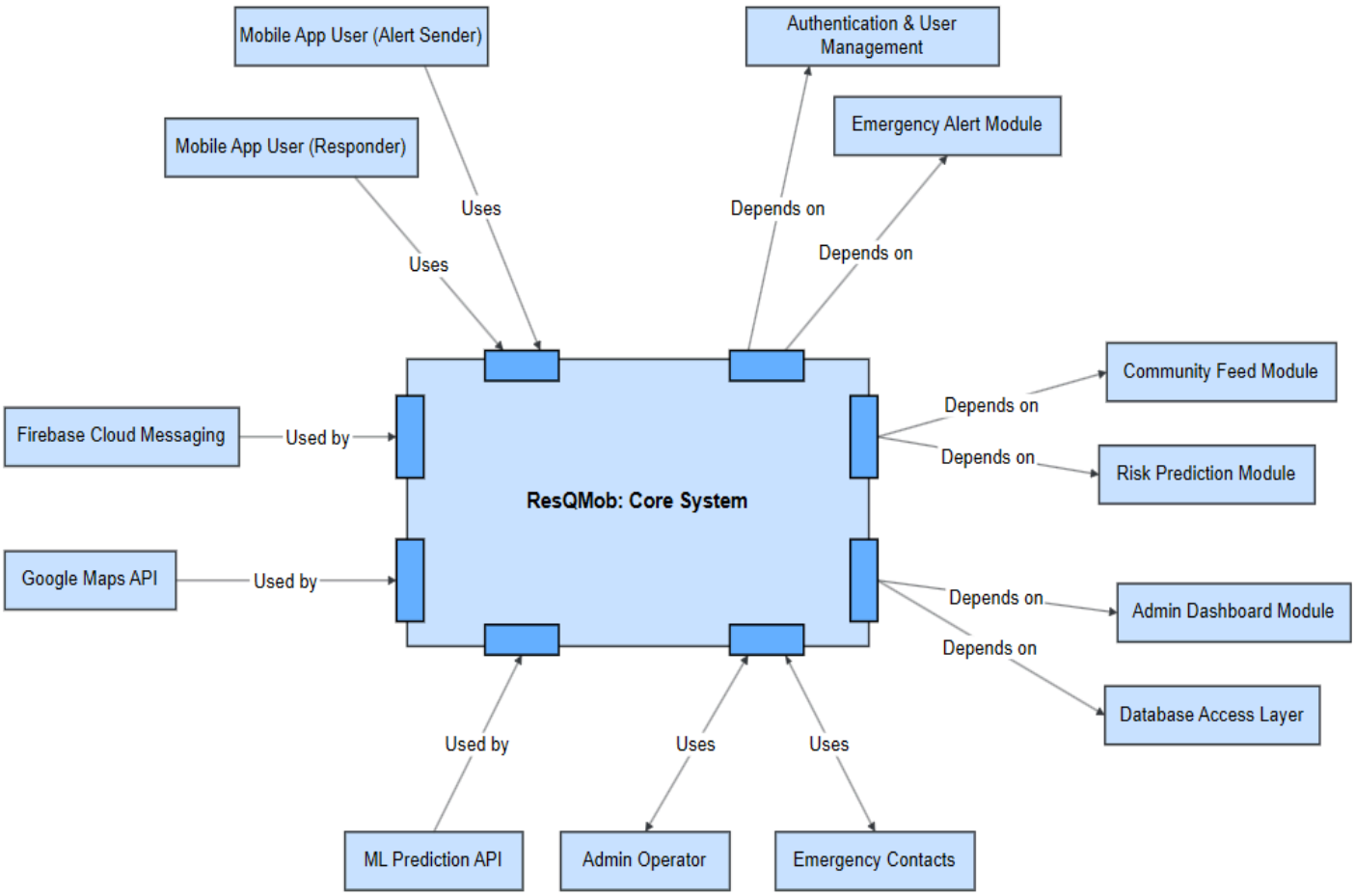
Fig 1: Level 0 Context Diagram



Level 1 - Top-Level Component Diagram (Pressman style layout)

- ➔ Emergency Alert Module (Manual, Auto)
- ➔ Location & Geo-fencing (Live Location, Danger Zone)
- ➔ Responder Management (Matching, Notifications)
- ➔ Community Feed (Posts, Reactions)
- ➔ ML Safe Road Prediction (Risk Scoring)
- ➔ Auth & User Management (Signup, Roles)
- ➔ External Services (SMS, Maps, Social APIs)

Fig 2: Level 1 Top-Level Component Diagram



Level 2 - Subsystem Decomposition (Pressman style)

Fig 3: Level 2A. Emergency Alert Module

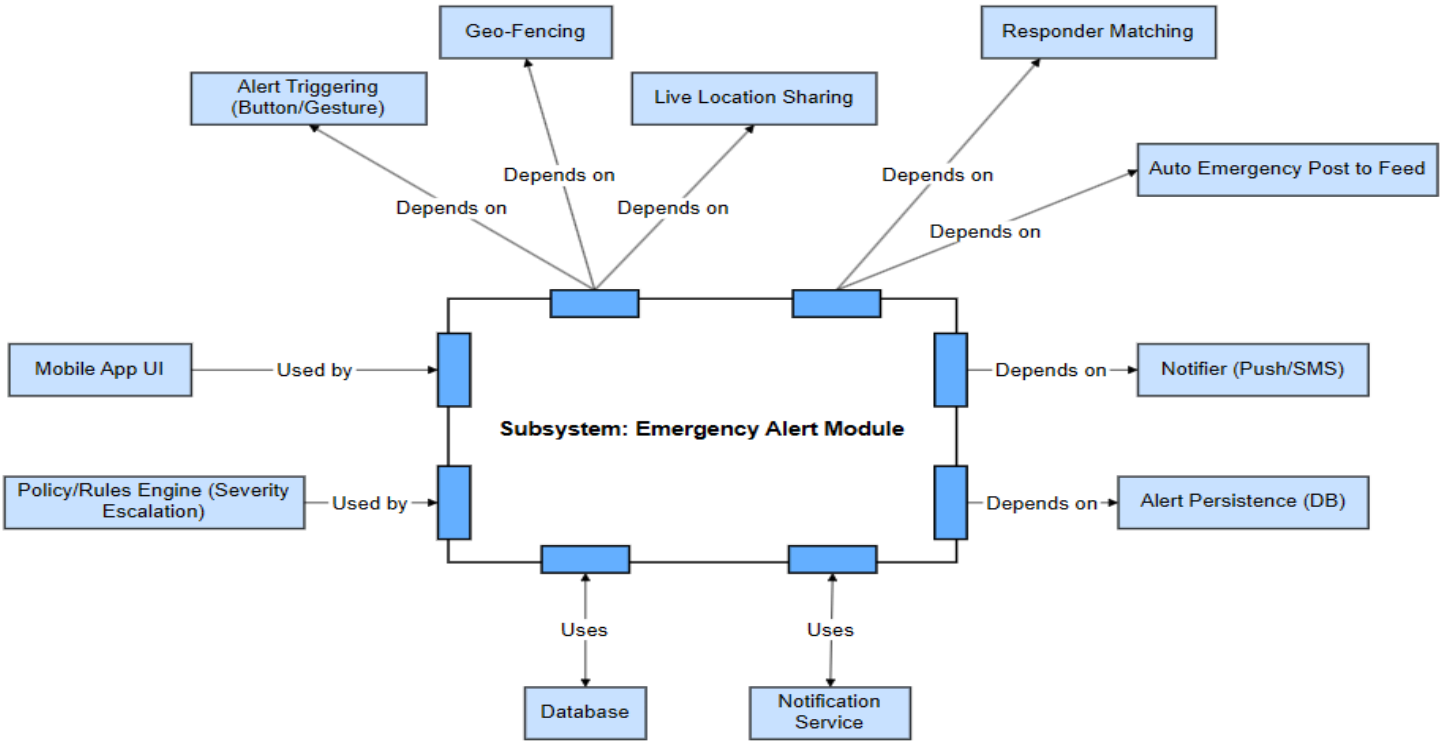
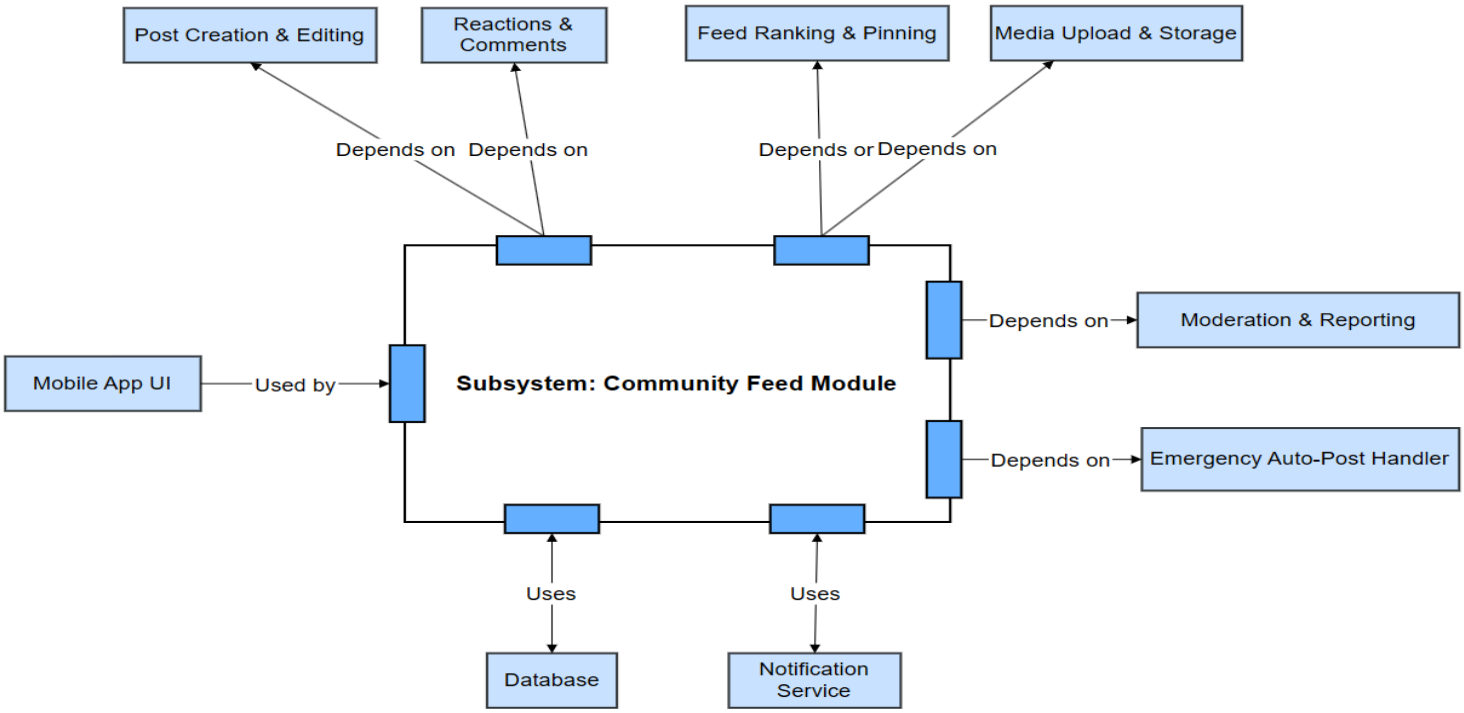
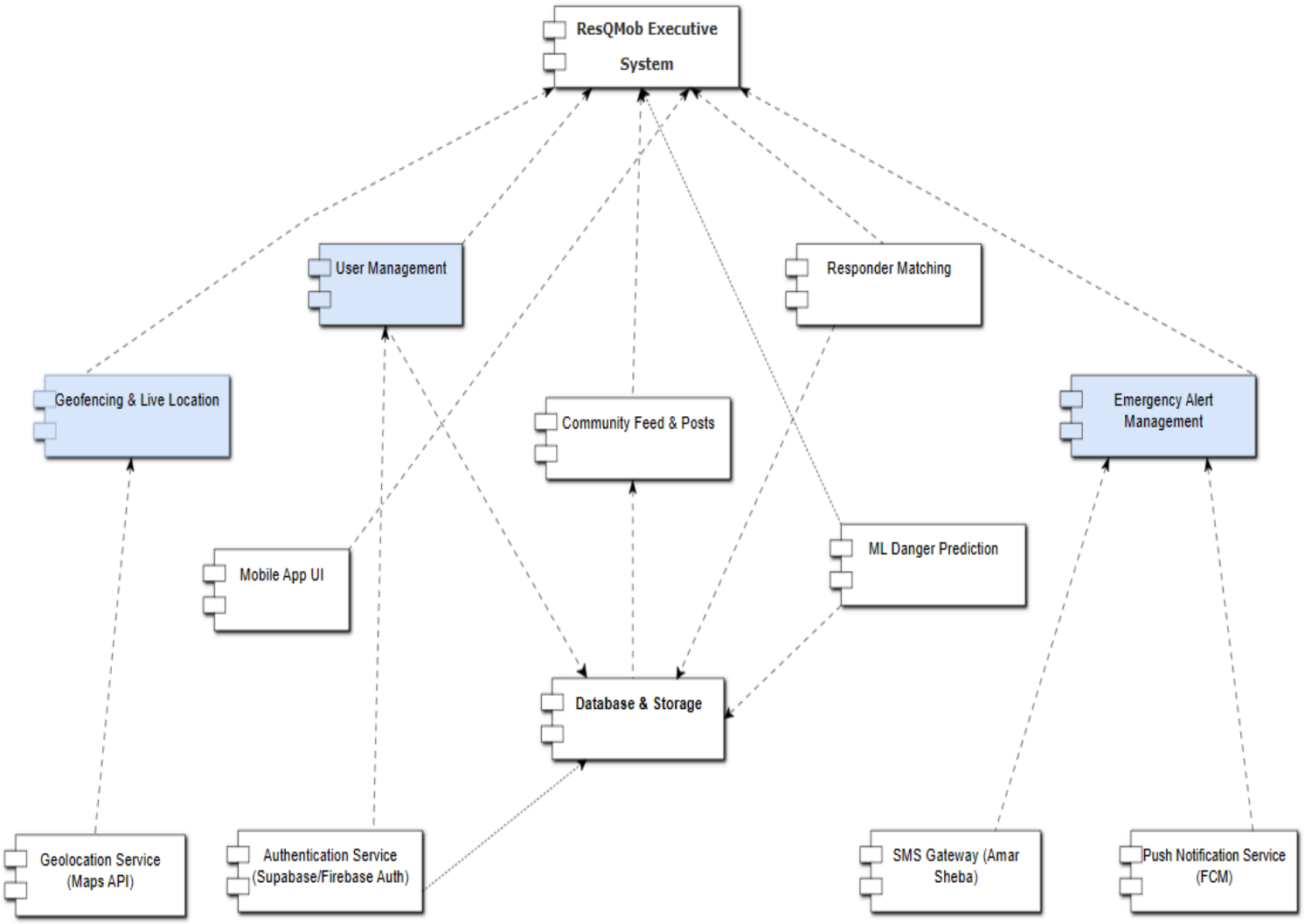


Fig 4: Level 2B. Community Feed Module



3.1.2 Top-Level Component Diagram



3.1.3 Instantiation of Each Component with Elaboration.

Component	Elaborated Sub-Components	Description of Role
ResQMob Executive	- System Initializer - Session Manager - Config Handler	Central control for system startup, session lifecycle, and configuration.
External Communication	- Push Notification Manager - SMS Gateway (Amar Sheba) - Email Alerts	Handles all outgoing emergency alerts and notifications to responders, users, and authorities.
GUI (Mobile App)	- Emergency Alert Button UI - Map & Location Sharing View - Community Feed UI - Profile & Settings	Provides user interaction points for triggering SOS, viewing live updates, managing profile.
Internet Interface	- Firebase API Connector - Firebase Realtime Updates - Edge Functions Handler	Connects frontend with backend services, handles API calls and real-time updates.
Security	- Auth Manager (Firebase Auth) - Role/Permission Handler - Data Encryption Module	Provides secure login, authorization (admin/responder/user), and data protection.
Control Panel Processing	- SOS Trigger Processor - Responder Matching Engine - Emergency Flow Manager	Core logic for processing SOS alerts, finding responders, and handling escalation.
Detector/Alert Management	- Geofencing Module - ML-based Danger Predictor - Behavior & Sensor Data Analyzer	Manages automatic triggers from sensors, predicts possible threats, detects false positives.
Alarm/Response Processing	- Alert Validation - Prioritization Engine - Escalation Manager	Processes emergency alerts, prioritizes based on severity, and routes to nearest responders.

Scheduler	- Task Queue Manager - Notification Scheduler - Retry Mechanism	Ensures timed notifications, manages task execution, retries failed alerts.
Phone Communication	- Call API Integration - VOIP/SIP Handler - Emergency Hotline Dialer	Direct calling features for connecting with responders or authorities.
Community Feed	- Post/React System - Lost & Found - Safety Announcements	Manages user-generated posts, reactions, safety info sharing.
Sensor/Data Input	- GPS Location Sensor - Accelerometer/Shake Trigger - Background Service	Collects real-time location and sensor data for emergency detection.
Alarm Output	- Siren Alert UI - Device Vibration & Sound - External Alert Integration	Provides physical/emotional alarm feedback to user.

3.2 Component-Level Design

3.2.1 Elaboration of Design Components

Main Components

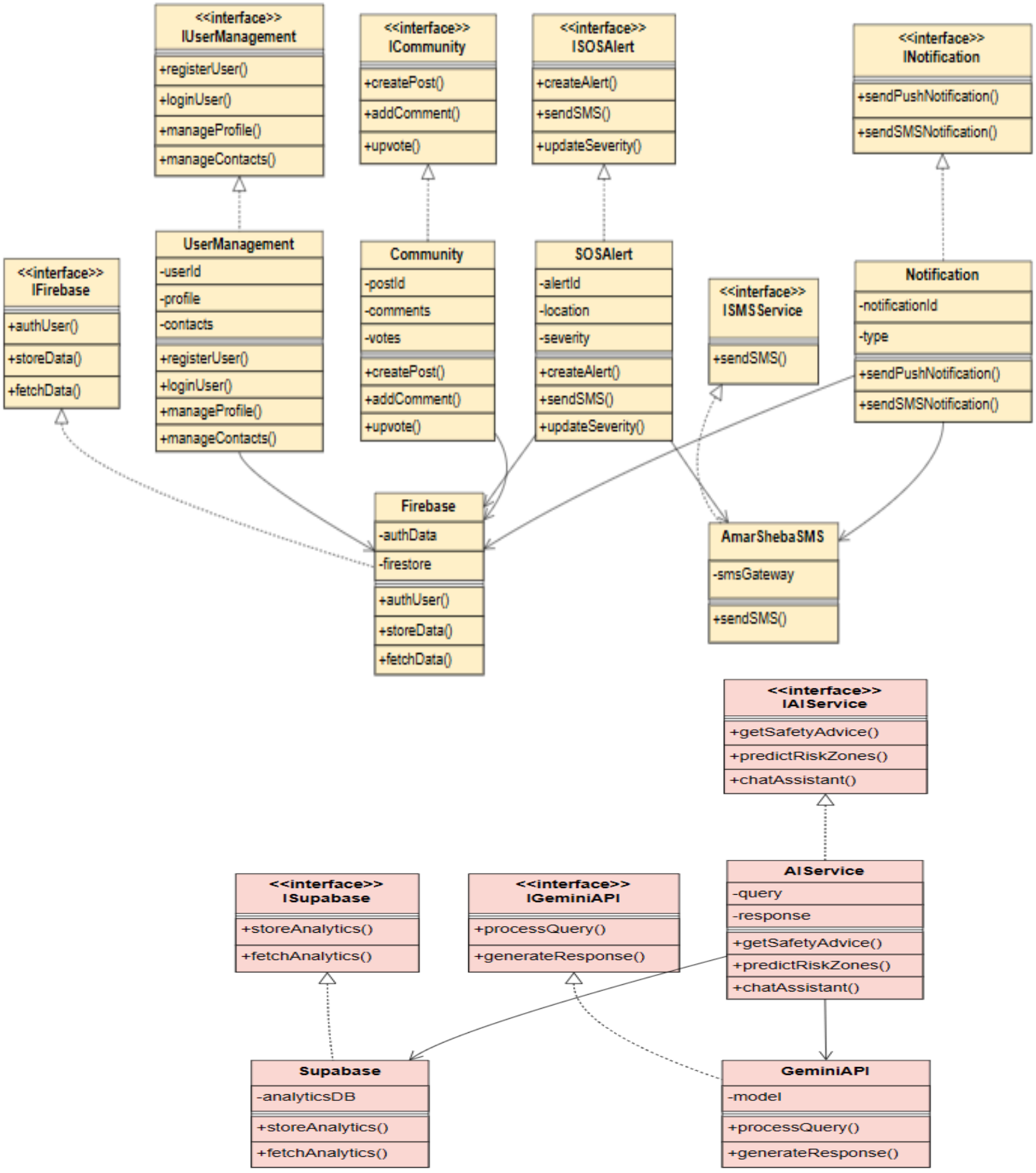
1) **User Module**
    a) Handles user registration, login, authentication (Firebase Auth).
    b) Manages profiles, emergency contacts, and preferences.
2) **SOS & Alert Module**
    a) Core component for generating emergency alerts.
    b) Uses GPS location + network data.
    c) Integrates with Amar Sheba SMS API for sending alerts.
    d) Stores alert data in Firebase/Firestore.

3) **Social & Community Module**
    a) Allows users to post, comment, and react to community updates.
    b) Useful for peer-to-peer help and local awareness.
    c) Stores posts/comments in Firestore.
4) **AI Assistant (Gemini API)**
    a) Provides real-time safety guidance (e.g., “nearest police station”).
    b) Predicts high-risk zones using feedback + map data.
    c) Acts as chatbot for quick help.
5) **Database & Storage Module**
    a) Firebase Firestore: real-time NoSQL DB for alerts, users, posts.
    b) Supabase (Postgres): structured storage for analytical data & ML.
    c) Firebase Storage: media uploads (photos, voice clips).
6) **Responder & Admin Module**
    a) Used by police, hospitals, or community responders.
    b) Provides dashboard to view alerts, severity levels, live tracking.
    c) Sends response updates back to users.
7) **Notification & Communication Module**
    a) Push notifications (Firebase Cloud Messaging).
    b) SMS alerts (Amar Sheba API).
    c) In-app real-time updates.

Table: Component(Pressman Style)

Component	Responsibilities	Inputs	Outputs	Dependencies
User Management	Auth, profiles, emergency contacts	Credentials, profile info, contacts	User session, stored profile	Firebase Auth, Firestore
SOS & Alert	SOS generation & dissemination	User ID, location, emergency type	Alert records, SMS messages	GPS, Firestore, Cloud Functions, SMS API
Community Interaction	Posts, comments, upvotes	User posts, comments, votes	Stored posts/comments, community feed	Firestore, Firebase Storage
AI Assistance	Safety guidance, chatbot, ML risk prediction	Queries, map data, feedback	AI responses, heatmap, safety suggestions	Gemini API, Supabase
Responder & Admin	Dashboard for responders	Alerts, severity updates, responder input	Updated status, responder feedback	Firestore, Cloud Functions
Notification & Communication	Push & SMS notifications	Triggers (alerts, updates, posts)	Push notifications, SMS alerts	FCM, Amar Sheba SMS API

Fig: 5 & 6 Competent Diagram


<<interface>>
IUserManagement
+registerUser()
+loginUser()
+manageProfile()
+manageContacts()
<<interface>>
ICommunity
+createPost()
+addComment()
+upvote()
<<interface>>
ISOSAlert
+createAlert()
+sendSMS()
+updateSeverity()
<<interface>>
INotification
+sendPushNotification()
+sendSMSNotification()
<<interface>>
IFirebase
+authUser()
+storeData()
+fetchData()
UserManagement
-userId
-profile
-contacts
+registerUser()
+loginUser()
+manageProfile()
+manageContacts()
Community
-postId
-comments
-votes
+createPost()
+addComment()
+upvote()
SOSAlert
-alertId
-location
-severity
+createAlert()
+sendSMS()
+updateSeverity()
<<interface>>
ISMSService
+sendSMS()
Notification
-notificationId
-type
+sendPushNotification()
+sendSMSNotification()
Firebase
-authData
-firestore
+authUser()
+storeData()
+fetchData()
AmarShebaSMS
-smsGateway
+sendSMS()
<<interface>>
IAIService
+getSafetyAdvice()
+predictRiskZones()
+chatAssistant()
<<interface>>
ISupabase
+storeAnalytics()
+fetchAnalytics()
<<interface>>
IGeminiAPI
+processQuery()
+generateResponse()
AIService
-query
-response
+getSafetyAdvice()
+predictRiskZones()
+chatAssistant()
Supabase
-analyticsDB
+storeAnalytics()
+fetchAnalytics()
GeminiAPI
-model
+processQuery()
+generateResponse()

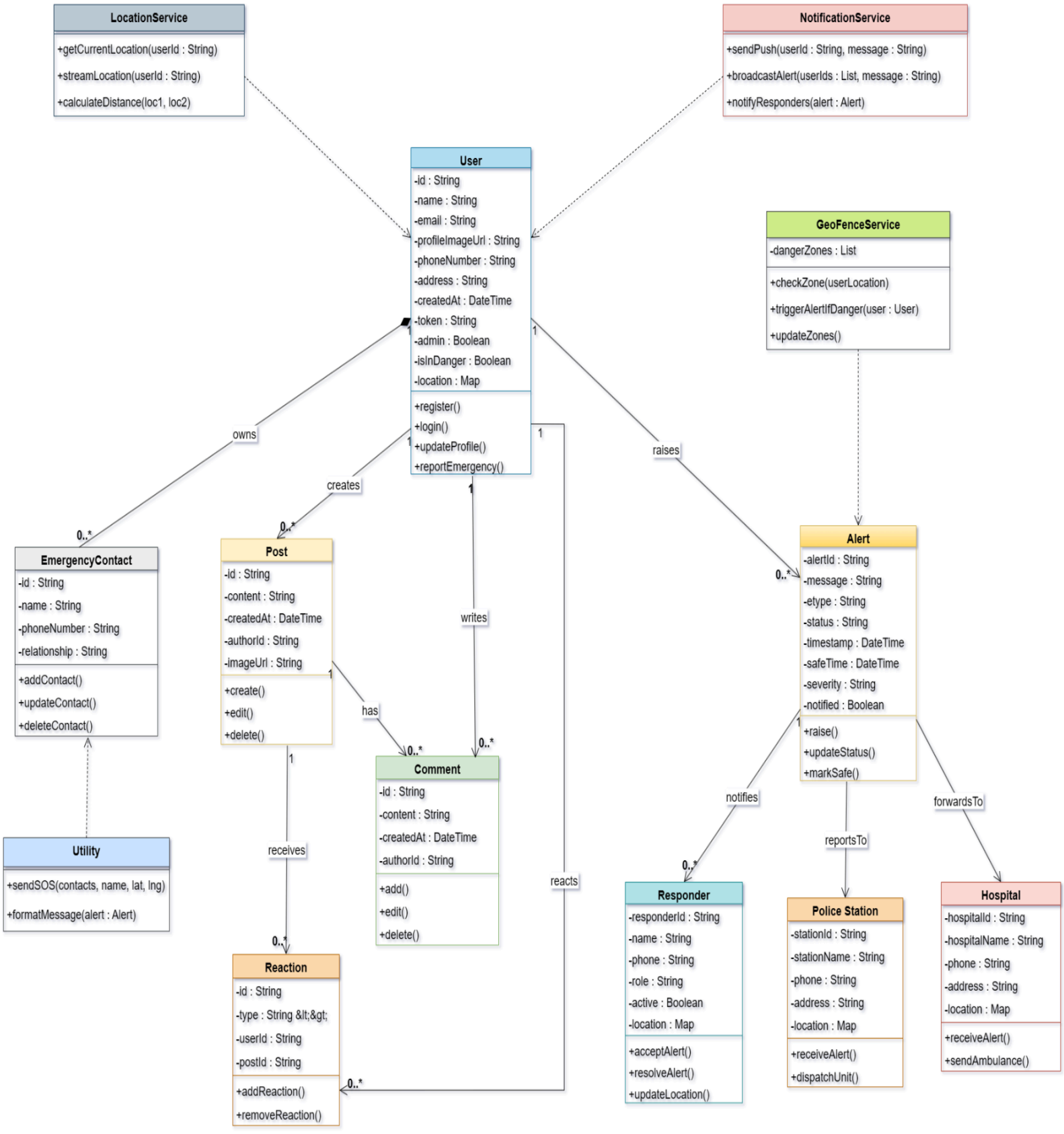
.

3.2.2 Class Diagram (Derived from Design Components)

Class Table:

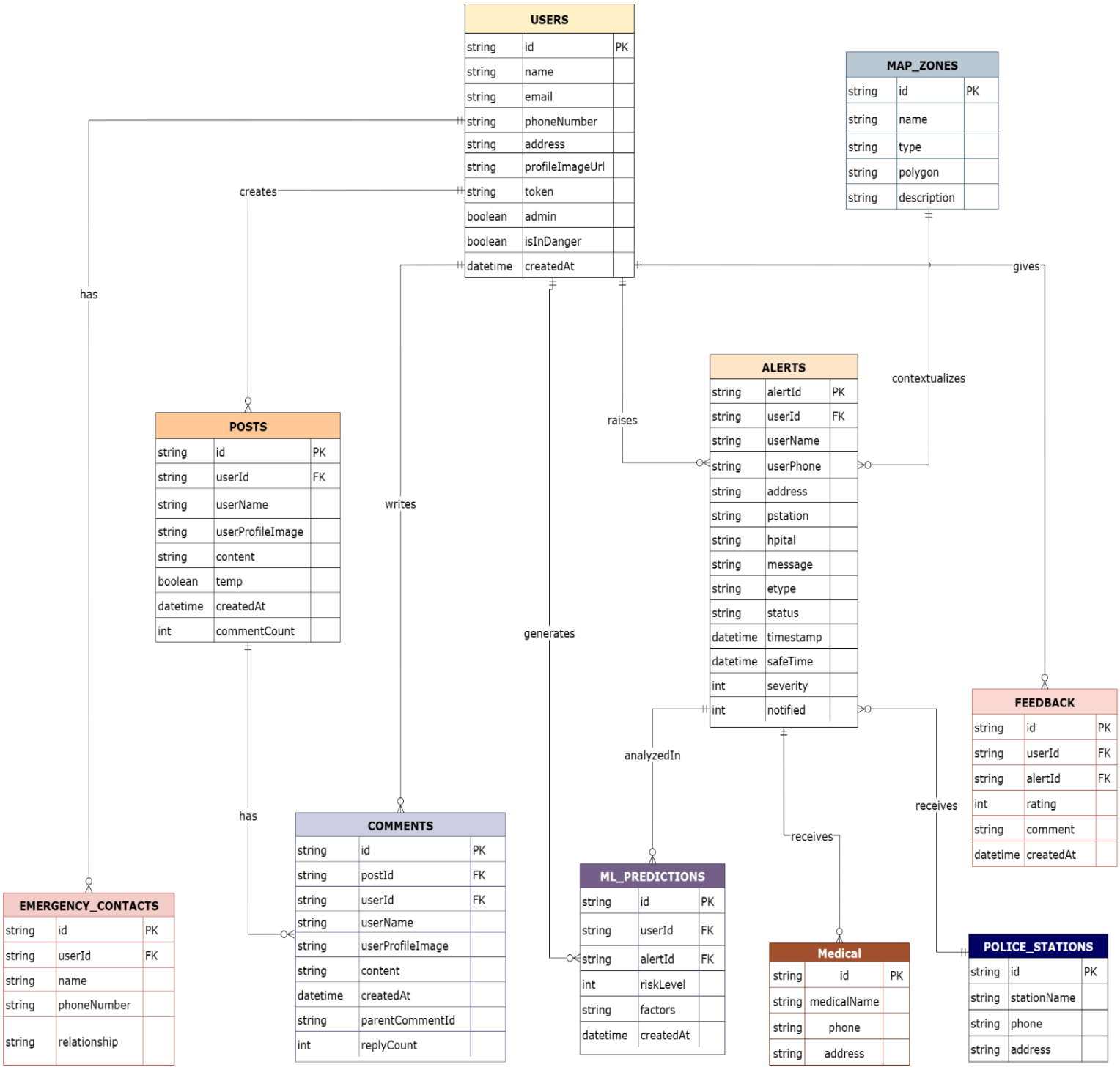
Class	Attributes	Methods	Responsibilities	Relationships
User	id : Stringname : Stringemail : StringprofileImageUrl : StringphoneNumber : Stringaddress : StringcreatedAt : DateTimetoken : Stringadmin : BooleanisInDanger : Booleanlocation : Map	register()login()updateProfile()reportEmergency()	Represents system users (regular, admin). Maintains personal info, manages profile, and initiates emergency reporting.	1..* EmergencyContact1..* Post1..* Alert
Emergency Contact	id : Stringname : StringphoneNumber : Stringrelationship : String	addContact()updateContact()deleteContact()	Stores user's emergency contacts to notify during emergencies.	Belongs to User (aggregation).
Post	id : Stringcontent : StringcreatedAt : DateTimeauthorId : StringimageUrl : String	create()edit()delete()	Represents community feed posts (text, image).	User (author)1..* Comment1..* Reaction
Comment	id : Stringcontent : StringcreatedAt : DateTimeauthorId : String	add()edit()delete()	Allows users to comment on community posts.	Belongs to Post
Reaction	id : Stringtype : String <> (like, alert, support)userId : StringpostId : String	addReaction()removeReaction()	Represents quick interactions with posts.	Belongs to Post, User

Fig 7: UML CLASS DIAGRAM



3.3 Database Design

Fig 8: ER Diagram



3.3.1 Table 1: Functions (Backend / Cloud Functions)

User Functions createUser(data) → Add new user getUserById(userId) → Fetch user profile updateUser(userId, data) → Edit profile/location/status addEmergencyContact(userId, contact) removeEmergencyContact(userId, contactId)	Backup_User Functions createUser(data) → Add new user getUserById(userId) → Fetch user profile updateUser(userId, data) → Edit profile/location/status addEmergencyContact(userId, contact) removeEmergencyContact(userId, contactId)
Post Functions createPost(data) getPosts(limit, filter) → fetch feed addUpvote(postId, userId) removeUpvote(postId, userId) deletePost(postId) Comment Functions addComment(postId, comment) getComments(postId) replyToComment(postId, commentId, reply) upvoteComment(postId, commentId, userId)	**Alert Functions** createAlert(data) getActiveAlerts() markAlertSafe(alertId) assignResponder(alertId, userId) notifyResponders(alertId) → via **sendSos SMS** Police Station Functions addStation(data) getNearestStation(lat, lng) updateStation(id, data)

Utility sendSos(contacts, name, lat, lng) → already implemented in sms.dart Can be triggered automatically when isInDanger = true or createAlert() is called.	**In short:** Collections = users, posts, posts/{postId}/comments, alerts, police_stations Functions = CRUD + special ones (sendSos, markAlertSafe, notifyResponders, getNearestStation).

3.3.2 Table 2: Fetching

1. **Table users** { id, name,email ,profile_image_url, phone_number ,address ,created_at , fcm_token , msg, is_admin, is_in_danger , location_latitude, location_longitude }

2. **Table emergency_contacts** { id, user_id, name, phone_number, relationship }

3. **Table alerts** { id, user_id, user_name, user_phone, address, police_station, hospital, message, emergency_type, status, timestamp, safe_time, severity, notified_count, location_latitude, location_longitude }

4. **Table posts** { id, user_id, user_name, user_profile_image, content, is_temporary, created_at, comment_count }

5. **Table comments** { id, post_id, user_id, parent_comment_id, user_name, user_profile_image, content, created_at, reply_count }

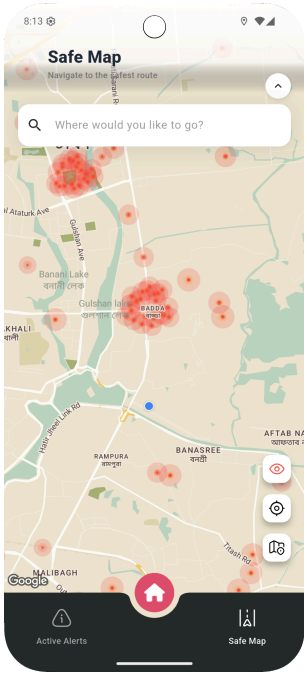
6. **Table feedback_messages** { id, user_id, message, created_at }

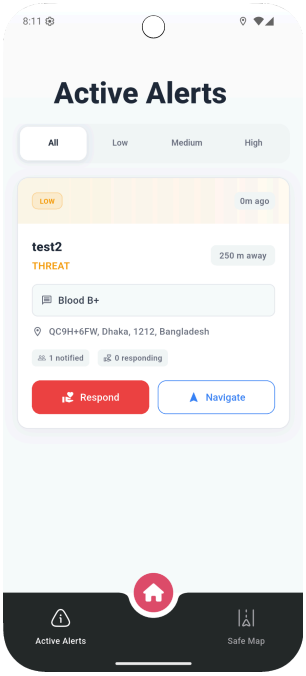
7. **Table hospitals** { id, hospital_name, phone, address, latitude, longitude }

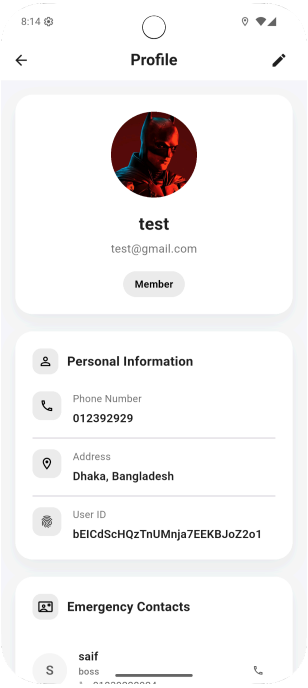
8. **Table police_stations** { id, station_name, phone, address, latitude, longitude }

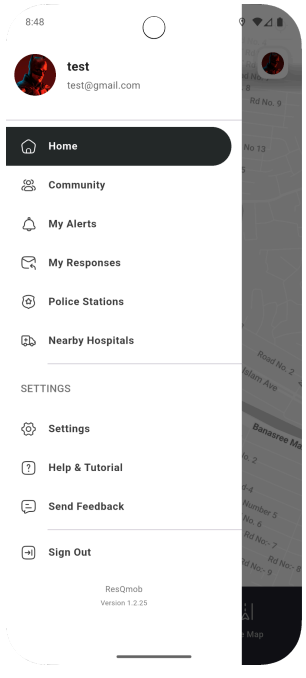
3.3 User Interface Design

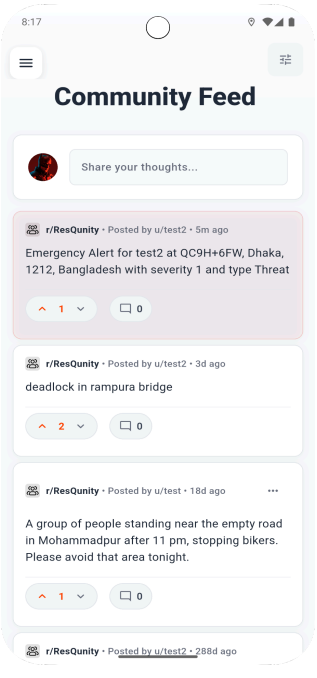
User UI Screens:

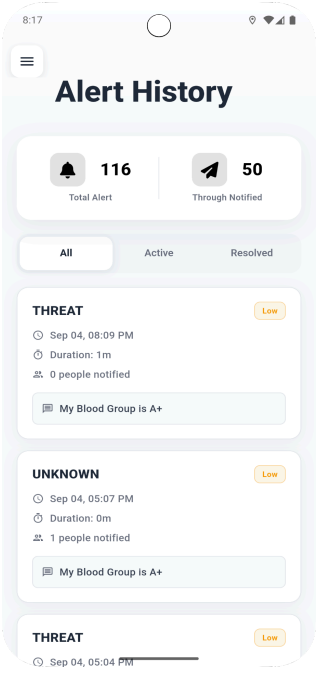


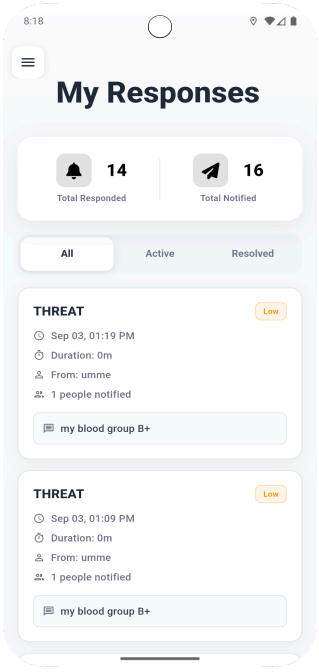


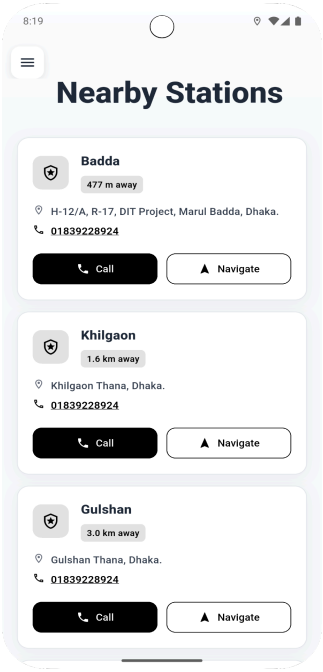


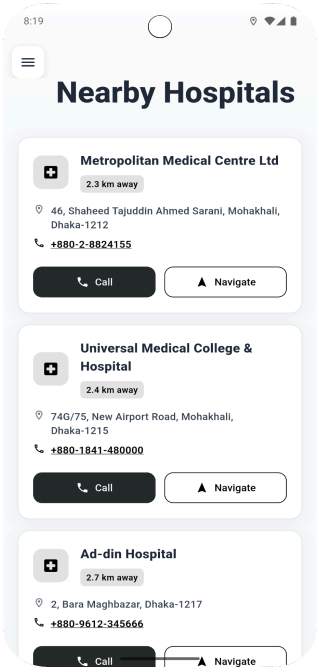


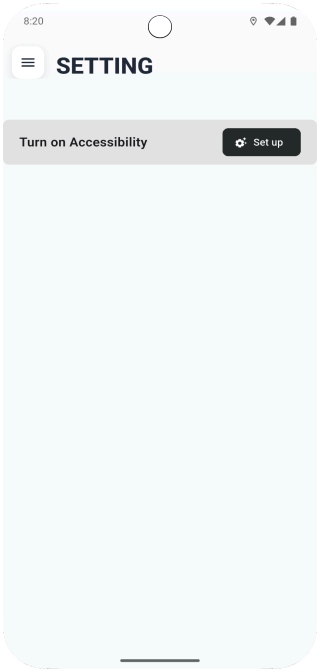


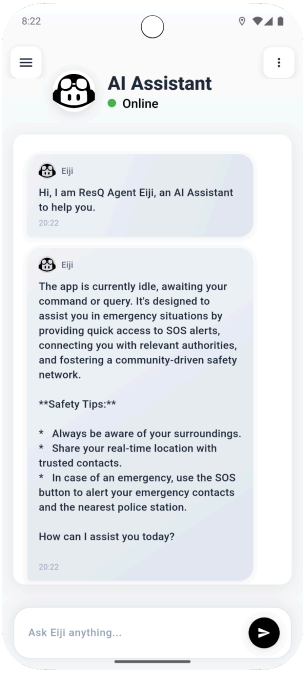


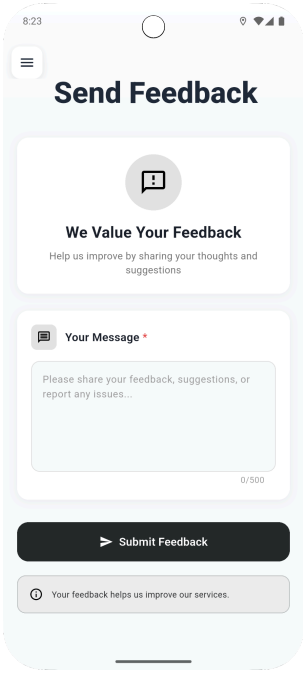

8:18
My Responses
14
Total Responded
16
Total Notified
All
Active
Resolved
THREAT
Low
Sep 03, 01:19 PM
Duration: 0m
From: umme
1 people notified
my blood group B+
THREAT
Low
Sep 03, 01:09 PM
Duration: 0m
From: umme
1 people notified
my blood group B+

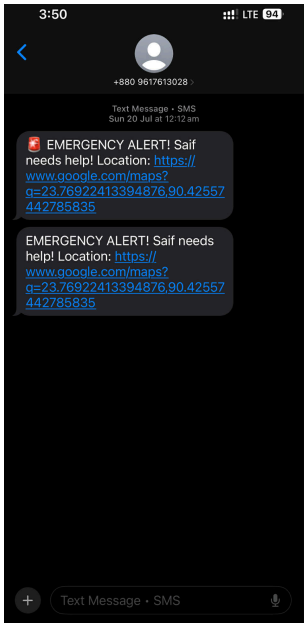

8:19
Nearby Stations
Badda
477 m away
H-12/A, R-17, DIT Project, Marul Badda, Dhaka.
01839228924
Call
Navigate
Khilgaon
1.6 km away
Khilgaon Thana, Dhaka.
01839228924
Call
Navigate
Gulshan
3.0 km away
Gulshan Thana, Dhaka.
01839228924
Call
Navigate


8:19
Nearby Hospitals
Metropolitan Medical Centre Ltd
2.3 km away
46, Shaheed Tajuddin Ahmed Sarani, Mohakhali, Dhaka-1212
+880-2-8824155
Call
Navigate
Universal Medical College & Hospital
2.4 km away
74G/75, New Airport Road, Mohakhali, Dhaka-1215
+880-1841-480000
Call
Navigate
Ad-din Hospital
2.7 km away
2, Bara Maghbazar, Dhaka-1217
+880-9612-345666
Call
Navigate

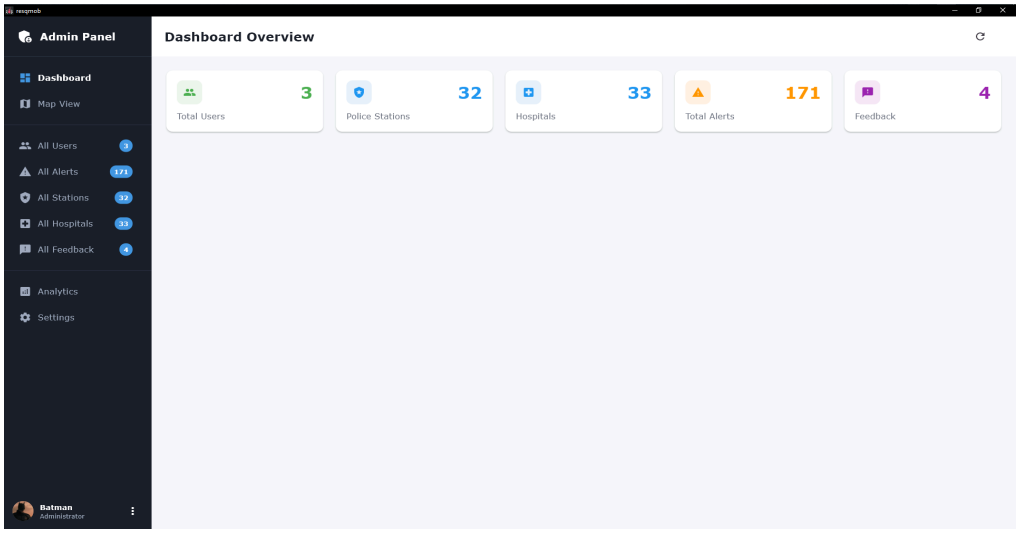

8:20
SETTING
Turn on Accessibility
Set up

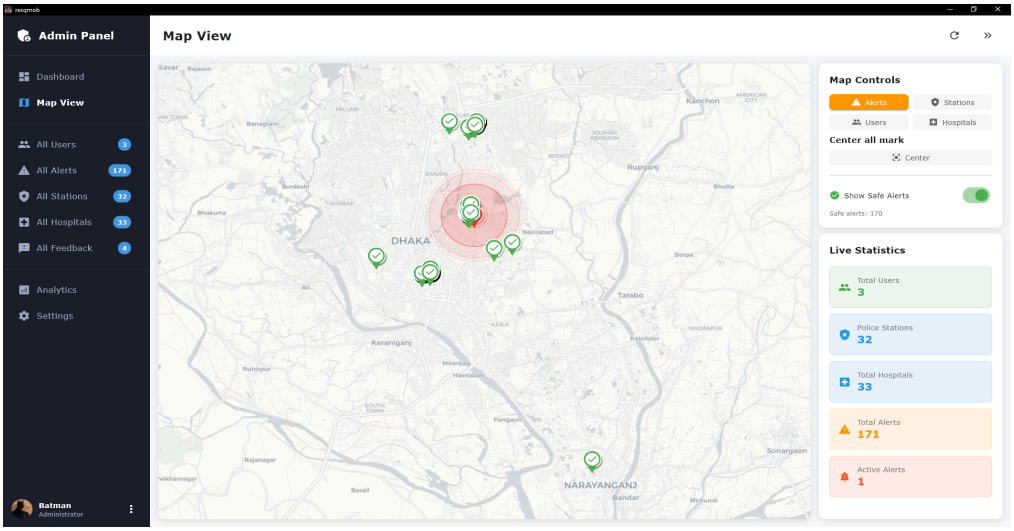

8:22
AI Assistant
Online
Eiji
Hi, I am ResQ Agent Eiji, an AI Assistant to help you.
20:22
Eiji
The app is currently idle, awaiting your command or query. It's designed to assist you in emergency situations by providing quick access to SOS alerts, connecting you with relevant authorities, and fostering a community-driven safety network.
Safety Tips:
* Always be aware of your surroundings.
* Share your real-time location with trusted contacts.
* In case of an emergency, use the SOS button to alert your emergency contacts and the nearest police station.
How can I assist you today?
20:22
Ask Eiji anything...

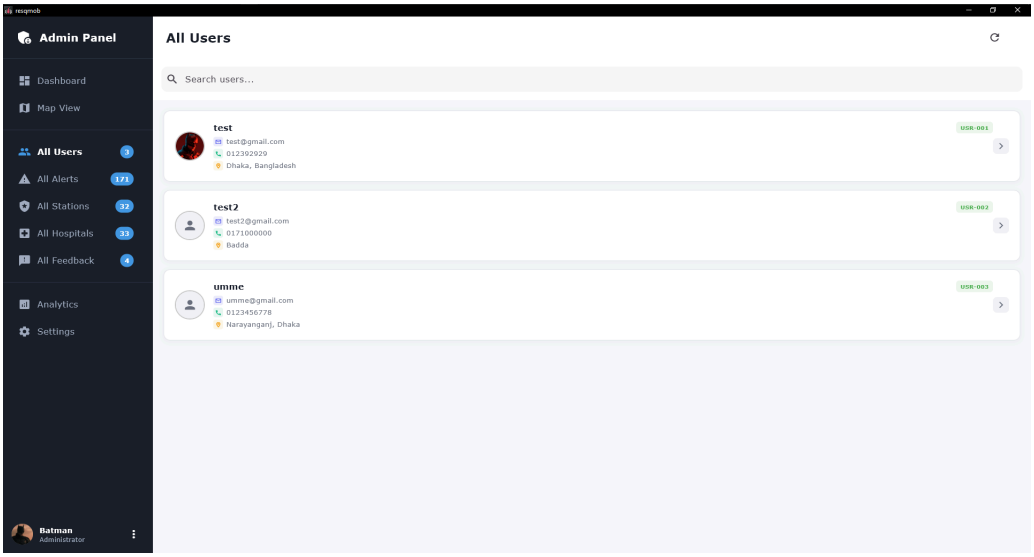

8:23
Send Feedback
We Value Your Feedback
Help us improve by sharing your thoughts and suggestions
Your Message *
Please share your feedback, suggestions, or report any issues...
0/500
Submit Feedback
Your feedback helps us improve our services.

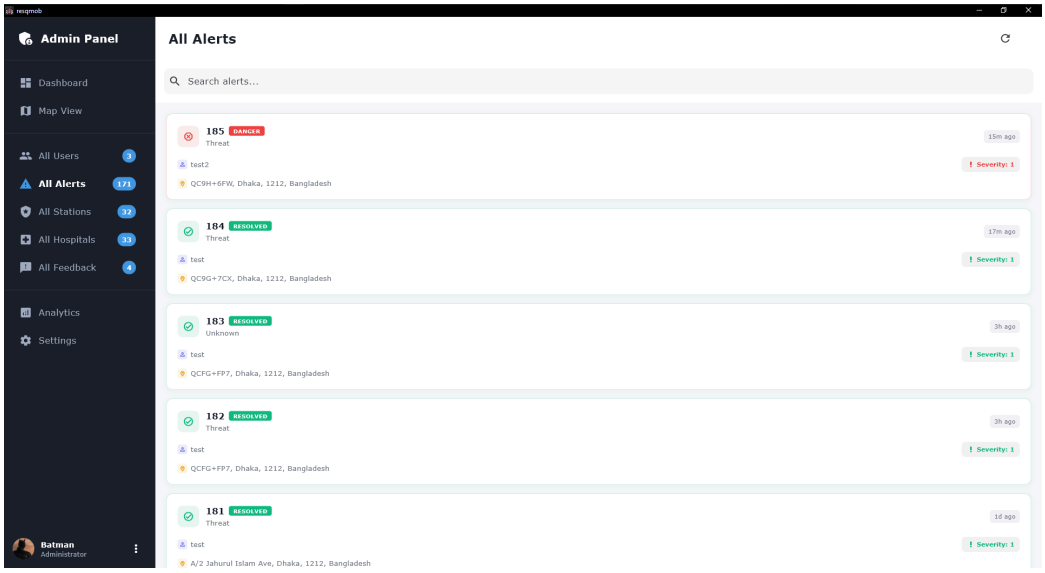

3:50
LTE 94
+880 9617613028
Text Message • SMS
Sun 20 Jul at 12:12 am
EMERGENCY ALERT! Saif needs help! Location: https://www.google.com/maps?q=23.76922413394876,90.42557442785835
EMERGENCY ALERT! Saif needs help! Location: https://www.google.com/maps?q=23.76922413394876,90.42557442785835
Text Message • SMS

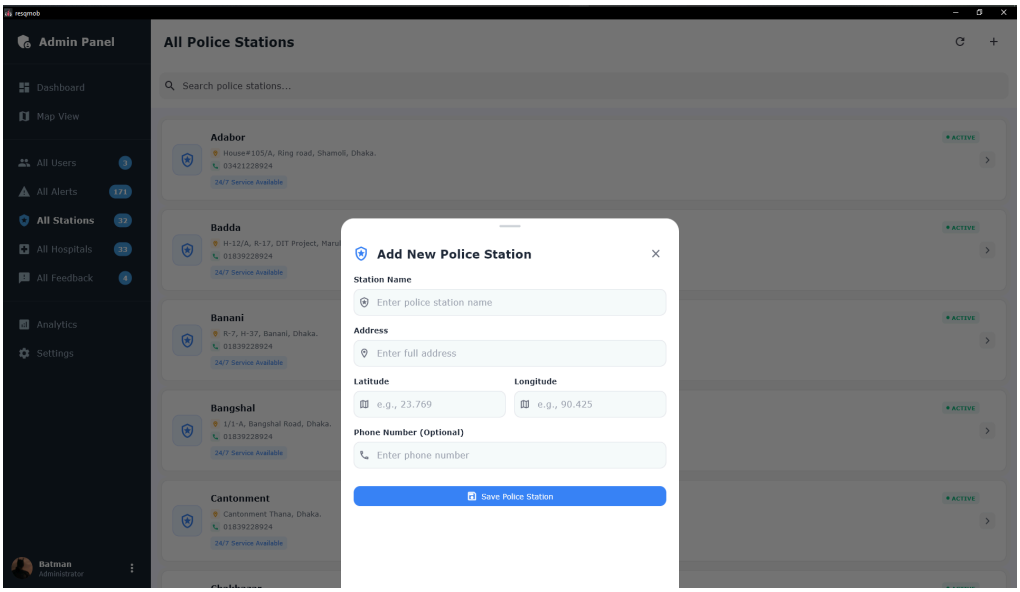
Admin Panel Screens:

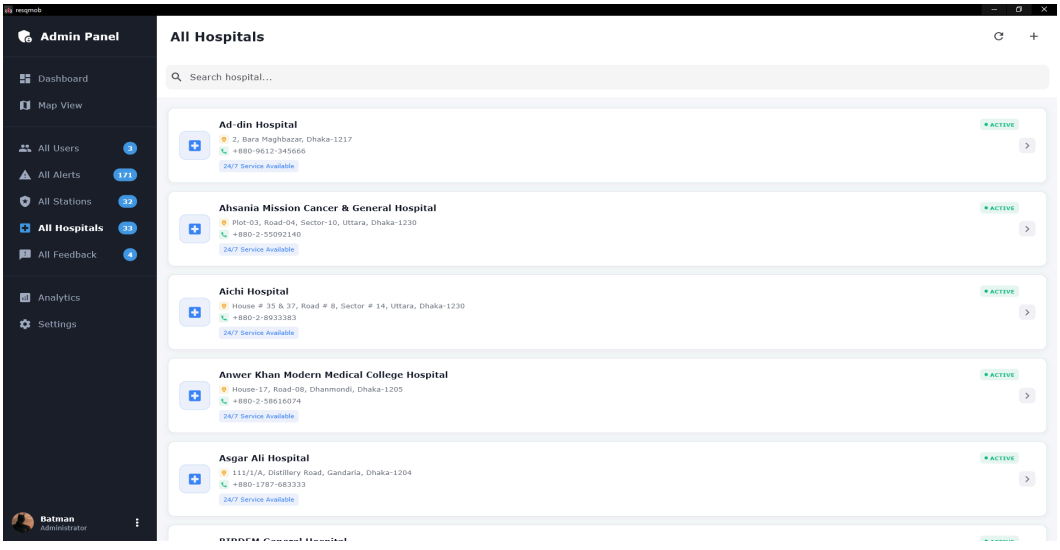


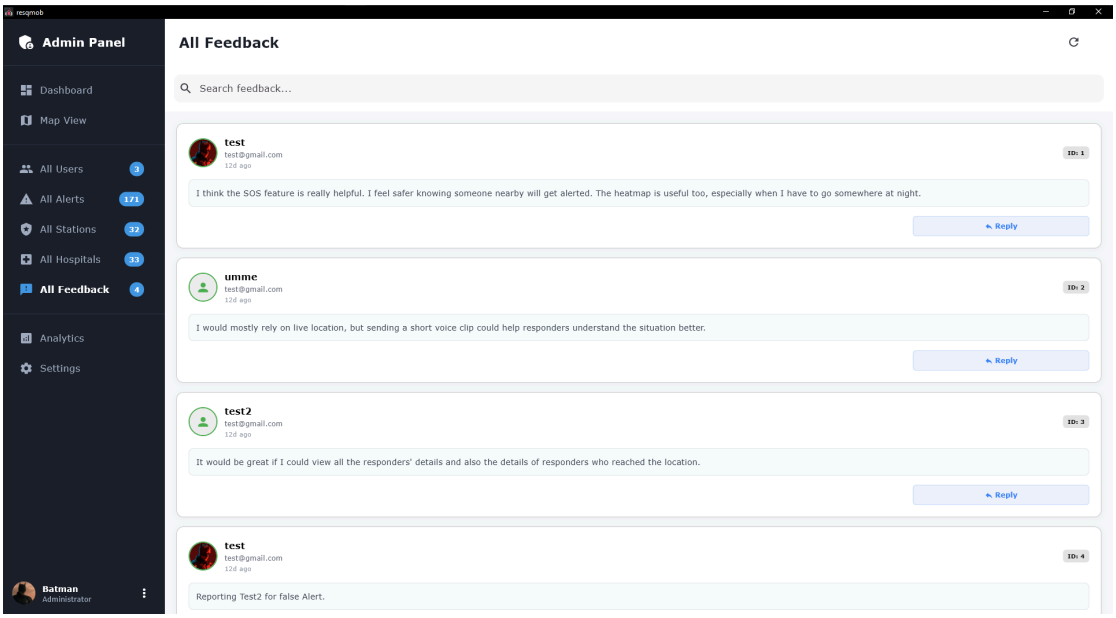





resqmob
Admin Panel
Dashboard
Map View
All Users 3
All Alerts 171
All Stations 32
All Hospitals 33
All Feedback 4
Analytics
Settings
Batman
Administrator
All Alerts
Search alerts...
185 DANGER
Threat
15m ago
test2
Severity: 1
QC9H+6FW, Dhaka, 1212, Bangladesh
184 RESOLVED
Threat
17m ago
test
Severity: 1
QC9G+7CX, Dhaka, 1212, Bangladesh
183 RESOLVED
Unknown
3h ago
test
Severity: 1
QCFG+FP7, Dhaka, 1212, Bangladesh
182 RESOLVED
Threat
3h ago
test
Severity: 1
QCFG+FP7, Dhaka, 1212, Bangladesh
181 RESOLVED
Threat
1d ago
test
Severity: 1
A/2 Jahurul Islam Ave, Dhaka, 1212, Bangladesh


resqmob
Admin Panel
All Police Stations
Search police stations...
Adabor
House#105/A, Ring road, Shamoli, Dhaka.
03421228924
24/7 Service Available
ACTIVE
Badda
H-12/A, R-17, DIT Project, Maruf
01839228924
Banani
R-7, H-37, Banani, Dhaka.
Bangshal
1/1-A, Bangshal Road, Dhaka.
Cantonment
Cantonment Thana, Dhaka.
Add New Police Station
Station Name
Enter police station name
Address
Enter full address
Latitude
e.g., 23.769
Longitude
e.g., 90.425
Phone Number (Optional)
Enter phone number
Save Police Station


resqmob
Admin Panel
All Hospitals
Search hospital...
Ad-din Hospital
2, Bara Maghbazar, Dhaka-1217
+880-9612-345666
24/7 Service Available
ACTIVE
Ahsania Mission Cancer & General Hospital
Plot-03, Road-04, Sector-10, Uttara, Dhaka-1230
+880-2-55092140
Aichi Hospital
House # 35 & 37, Road # 8, Sector # 14, Uttara, Dhaka-1230
+880-2-8933383
Anwer Khan Modern Medical College Hospital
House-17, Road-08, Dhanmondi, Dhaka-1205
+880-2-58616074
Asgar Ali Hospital
111/1/A, Distillery Road, Gandaria, Dhaka-1204
+880-1787-683333



3.3.2 Navigation Flow

1. **Open App** → Authentication → Home Screen
2. **Homepage** → Trigger SOS → Type of emergency (optional)→Alert activated to different levels →Auto post in community feed for updates →Mark a **Safe** →Post deleted
3. **Homepage** → Active Alerts → View active alerts
4. **Homepage** → Enter destination→ Find selected routes & other routes according to safety percentages.
5. **Admin Login** → Dashboard → Monitor Alerts
6. **Menu:**

   6.1 Community → Create post → Upvote → Do comment

6.2 My Alerts → Alert's history → Total alerts count → Total notified users → Active alerts

→ All alerts → Active alerts → Resolved alerts

6.3 My Responses → Total responded alerts → Total notified alerts

6.4 Police Stations → Nearby Stations → Make call → Navigate

6.5 Nearby Hospitals → Nearest Hospitals → Make call → Navigate

6.6 Settings → Set up accessibility

6.7 Help Tutorial → AI assistant → Provide information → Provide guidance

6.8 Send Feedback → App feedback → User feedback → Post feedback

6.9 Sign out

7. **Profile** →Set personal information → Phone number → Address → User ID by ResQmob →Emergency Contact → Emergency message

Chapter 4. Implementation

4.1 Code Structure

```
├── android/              # Android-specific configuration & build files
├── ios/ (if present)     # iOS-specific files
├── web/                  # Web deployment configuration
├── windows/              # Windows desktop support files
├── lib/                  # Main Flutter application code
│   ├── main.dart         # Entry point
│   ├── wrapper.dart      # flow control after main
│   ├── pages/            # App screens (Home, SOS, Settings, etc.)
│   ├── .backend/         # Firebase, location, SOS service handlers
│   ├── class models/     # Data models
│   └── modules/          # Reusable UI components
├── assets/               # Images, fonts, and other static assets
├── .firebase/            # Firebase-related local config
├── firebase.json         # Firebase hosting & project settings
├── apphosting.yaml       # Hosting config for Firebase
├── pubspec.yaml          # Flutter dependencies and metadata
├── pubspec.lock          # Locked dependency versions
├── analysis_options.yaml # Code analysis rules
├── test/                 # Unit and widget tests
└── README.md             # Project documentation
```

4.2 GitHub Repository URL

XhAfAn1/Emergency-SOS-System

https://github.com/XhAfAn1/Emergency-SOS-System

Chapter 5. Software Testing

5.1 White Box Testing (Unit Testing)

Class 1: UserModel (fromJson)

```
factory UserModel.fromJson(Map<String, dynamic> json) {
1.  List<EmergencyContact> contacts = [];
2.  if (json['emergencyContacts'] != null) {
3.      contacts = (json['emergencyContacts'] as List)
4.          .map((c) => EmergencyContact.fromJson(c))
5.          .toList();
    }
6.  bool adminValue = (json['admin'] is bool)
7.      ? json['admin']
8.      : json['admin'].toString().toLowerCase() == 'true';
9.  return UserModel(
10.     id: json['id'] ?? '',
11.     name: json['name'] ?? '',
12.     admin: adminValue,
13.     email: json['email'] ?? '',
14.     emergencyContacts: contacts,);
15.}
```

Control Flow Graph (CFG):

1. Start → Parse JSON → If emergencyContacts != null → Loop through contacts → Add to list → End

2. Start → Parse JSON → If emergencyContacts == null → Skip loop → End

3. Parse admin → If value is bool → assign → Else → convert string "true"/"false"

Graph Nodes:

A: Start → (1) init contacts

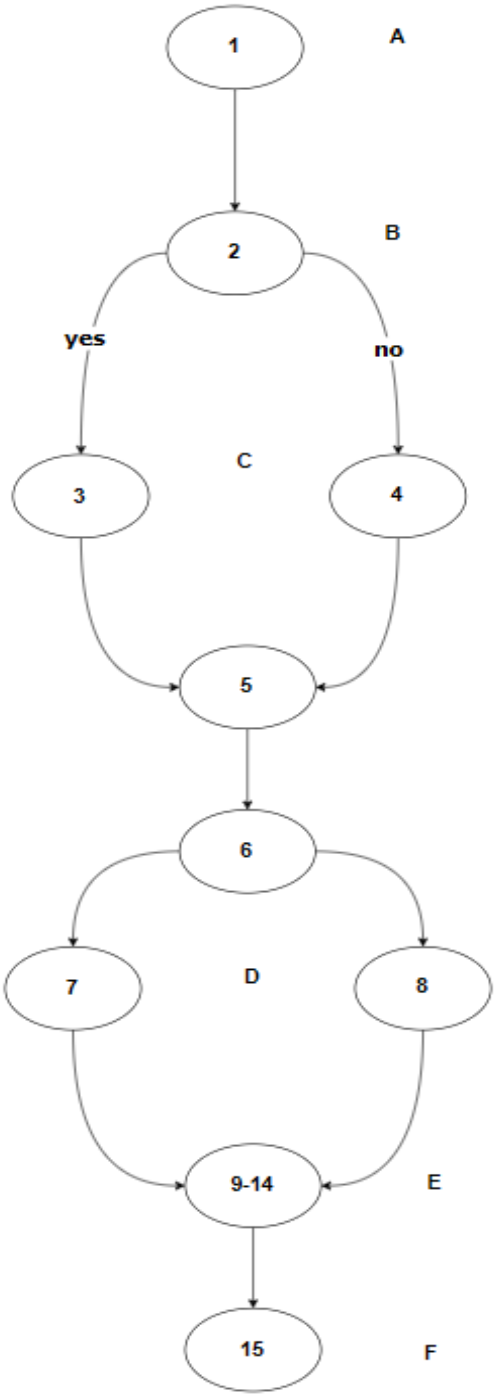
B: Condition: emergencyContacts present?

C: (3–5) Parse list → assign contacts

D: (6–8) Admin type check + assign

E: (9–14) Return UserModel

F: End



Cyclomatic Complexity (CC):

Method : Regions of the Graph + 1

➔ Count regions enclosed in the DD graph

➔ Regions = 2

Cyclomatic Complexity = R+1 =3

Independent Paths:

➔ **P1:** JSON contains contacts, valid `admin` boolean

A → B(Yes) → C → D → E → F

➔ **P2:** JSON contains no contacts

A → B(No) → D → E → F

➔ **P3:** JSON with `"admin": "true"` as string

A → B(No) → D(No) → E → F

Test Cases from Independent Paths

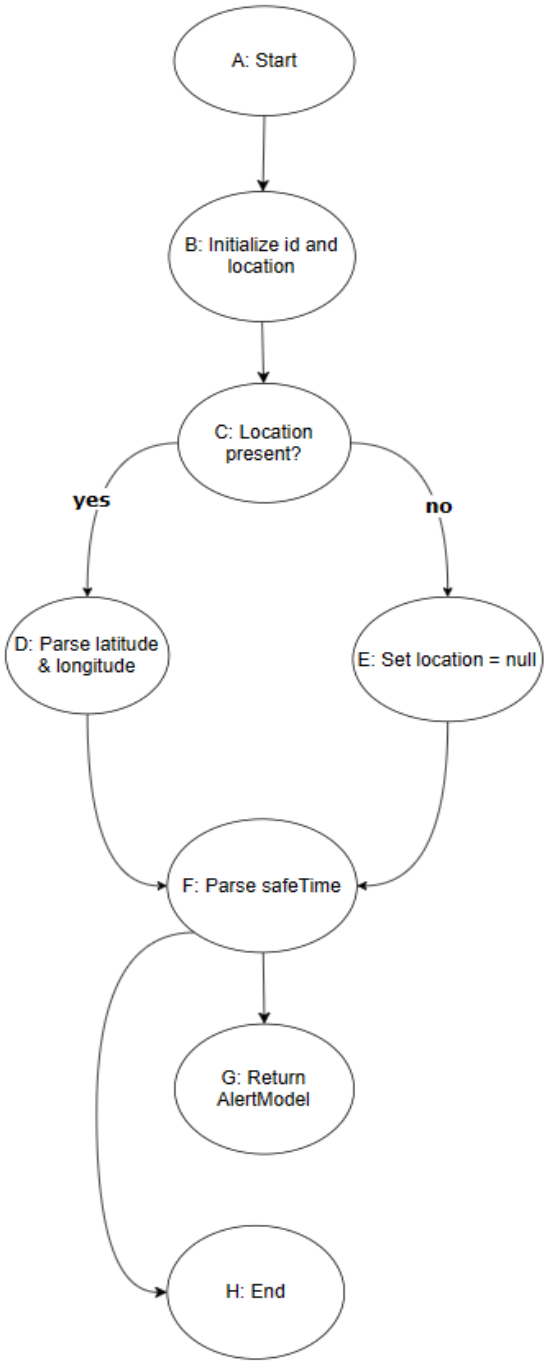
Test Case ID	Input JSON	Expected Behavior
TC-UM-01	{ "id": "1", "EFTE": "UMME", "admin": true, "email": "a@test.com", "emergencyContacts":[{"name":"SAIF","01458265365":"123"}] }	Creates UserModel with 1 contact, admin = true
TC-UM-02	{ "id": "2", "LAJIM": "", "admin": "true", "email": "c@test.com" }	Creates UserModel with empty contact list, admin = true (converted from string)
TC-UM-03	{ "id": "3", "LAJIM": "admin": "true", "email": "c@test.com" }	Creates UserModel with admin = true

Case ID	Component	Class	Path	Input	Expected Output	Actual Output	Pass/Fail
WB-UM-01	User Parsing	User Model	**P1**	JSON with 2 contacts + admin: true	User with 2 contacts, admin = true	User created successfully with contacts	Pass
WB-UM-02	User Parsing	User Model	**P2**	JSON with no contacts	User with empty list	User created successfully with empty list	Pass
WB-UM-03	User Parsing	User Model	**P3**	JSON with "admin": "true" (string)	Admin = true	As expected	Pass

Class 2: AlertModel (fromJson)

```
factory AlertModel.fromJson(Map<String, dynamic> json, String docId) {
1.  String id = docId;
2.  Map<String, dynamic>? location;
3.  if (json['location'] != null) {
4.     location = {
5.       'latitude': json['location']['latitude']?.toDouble(),
6.       'longitude': json['location']['longitude']?.toDouble(),
       };
    } else {
7.     location = null;
    }
8.  Timestamp? safeTime = json['safeTime'];

9.  return AlertModel(
10.    alertId: id,
11.    location: location,
12.    safeTime: safeTime,
13.  );}
```



DD Graph:

A: Start

B: Initialize `id` and `location`

C: Condition → *Location present?*

D: Parse latitude & longitude (if location present)

E: Set `location = null` (if location missing)

F: Parse `safeTime`

G: Return `AlertModel`

H: End

CFG:

1. **P1:** Location present → Parse lat/lng → safeTime present → Return
2. **P2:** Location missing → location = null → safeTime missing → Return
3. **P3:** Location present → Parse lat/lng → safeTime missing → Return

Cyclomatic Complexity:

Decision 1: location != null? → Yes / No

Decision 2: safeTime != null? → Yes / No

CC= Number of Decisions+1 = 2+1 = **3**

Independent Paths:

➔ **P1:** Location present, safeTime present
A → B → C(Yes) → D → F → G → H

➔ **P2:** Location missing, safeTime missing
A → B → C(No) → E → F → G → H

➔ **P3:** Location present, safeTime missing

A → B → C(Yes) → D → F → G → H

Test Case ID	Component	Class	Input	Expected Output	Actual Output	Pass/Fail
WB-AM-01	Alert Parsing	Alert Model	JSON with location data	Alert created with latitude & longitude	Location parsed successfully	Pass
WB-AM-02	Alert Parsing	Alert Model	JSON without location	Alert created with location = null	Alert stored with null location	Pass
WB-AM-03	Alert Parsing	Alert Model	JSON without safeTime	Alert created with safeTime = null	Alert stored with null safeTime	Pass

5.2 Black Box Testing (Functional Testing)

Feature 1: SOS Alert Trigger

Key Boundaries Identified:

- ➔ **Distance ranges**: 0-1000m (severity 1), 1000-10000m (severity 2), 10000-15000m (severity 3)
- ➔ **Severity levels**: 1-3 (maximum severity is 3)
- ➔ **Alert count**: Used for generating alertId (count + 15)
- ➔ **Location coordinates**: Valid latitude (-90 to 90), longitude (-180 to 180)

Test Cases Table:

Case ID	Component	Class	Input	Expected Output	Actual Output	Pass/Fail
TC_001	AlertSystem	Equivalence	currentPosition = null, user exists	Location request triggered, then normal alert flow	Failed to get location message	Failed
TC_002	AlertSystem	Equivalence	Valid location, user.isInDanger = false	New alert created, notifications sent to users within 1000m	Alert Created Successfully	Passed
TC_003	AlertSystem	Equivalence	Valid location, user.isInDanger = true, severity < 3	Severity incremented, extended range notifications sent	Severity incremented	Passed
TC_004	AlertSystem	Equivalence	Valid location, user.isInDanger = true, severity = 3	"Severity already at maximum" message shown	Severity is already at maximum	Passed
TC_005	AlertSystem	User Distance Boundary	Distance = 0m from user	No notification sent (0 < distance condition fails)	Failed to get user Distance	Failed
TC_006	AlertSystem	User Distance Boundary	Distance = 1m from user	Notification sent (within severity 1 range)	Alert Created Successfully	Passed
TC_007	AlertSystem	User Distance Boundary	Distance = 1001m from user	No notification sent for severity 1 (exceeds range)	No notification sent for severity 1	Passed
TC_08	AlertSystem	User Distance Boundary	Distance = 15000m, severity = 3	No notification sent (exceeds severity 3 range)	No notification sent	Passed
TC_09	AlertSystem	User Distance Boundary	Severity = 0 → 1	Severity incremented to 1, notifications sent	Severity is already 1	Failed
TC_010	AlertSystem	User Distance Boundary	Severity = 1 → 2	Severity incremented to 2, extended notifications sent	Severity incremented to 2	Passed

TC_011	AlertSystem	User Distance Boundary	Severity = 3 → 4 (attempt)	Severity remains 3, "maximum severity" message shown	Severity already maximum	Passed

Feature 2: Post & Comment System

Key Boundaries Identified:

- ➔ **Comment content length**: Empty string, single character, maximum length
- ➔ **Network connectivity**: Online/offline scenarios
- ➔ **Database states**: Existing/non-existing posts and comments

Test Case ID	Component	Class	Input	Expected Output	Actual Output	Pass/Fail
BB-PC-01	Social Feature	Comment Model	Empty comment ("")	Reject with error	Cannot tap sent button	Failed
BB-PC-02	Social Feature	Comment Model	1-char comment ("A")	Comment accepted & displayed	Comment displayed	Passed
BB-PC-03	Social Feature	Comment Model	Max length comment (500 chars)	Comment accepted & stored	Comment stored successfully	Passed

5.3 Bug Detection and Solution

Bug-01: Compilation error after adding home screen widget

Incorrect Code (Before Fix)

In the original code, there was **intent handler** when home screen widget.

```
1.  package com.example.resqmob
2.  import android.content.Intent
3.  import android.os.Bundle
4.  import android.os.Handler
5.  import android.os.Looper
6.  import android.view.KeyEvent
7.  import io.flutter.embedding.android.FlutterActivity
8.  import io.flutter.embedding.engine.FlutterEngine
9.  import io.flutter.plugin.common.MethodChannel
10. class MainActivity: FlutterActivity() {
11.     private val CHANNEL = "app_widget_channel"
12.     private var flutterEngineRef: FlutterEngine? = null
13.     // sequence tracking
14.     private val sequence = mutableListOf<Int>()
15.     private val handler = Handler(Looper.getMainLooper())
16.     private val resetRunnable = Runnable { sequence.clear() }
17.     override fun configureFlutterEngine(flutterEngine: FlutterEngine) {
18.         super.configureFlutterEngine(flutterEngine)
19.         flutterEngineRef = flutterEngine
20.
21.         MethodChannel(flutterEngine.dartExecutor.binaryMessenger, CHANNEL)
22.             .setMethodCallHandler { call, result ->
23.                 if (call.method == "getInitialData") {
24.                     result.success(handleIntent(intent))
25.                 } else {
26.                     result.notImplemented() }}
27.     override fun onNewIntent(intent: Intent) {
28.         super.onNewIntent(intent)
29.         sendDataToFlutter(intent)  }
30.     override fun onKeyDown(keyCode: Int, event: KeyEvent): Boolean {
31.         when (keyCode) {
32.             KeyEvent.KEYCODE_VOLUME_DOWN -> addKey(0)
33.             KeyEvent.KEYCODE_VOLUME_UP -> addKey(1)      }
34.         return super.onKeyDown(keyCode, event) }
35.     private fun addKey(key: Int) {
36.         sequence.add(key)
37.         // reset after 3s of inactivity
```

```
38.        handler.removeCallbacks(resetRunnable)
39.        handler.postDelayed(resetRunnable, 3000)
40.        // Expected sequence = [0,0,1,0,0]
41.        val expected = listOf(0, 0, 1, 0, 0)
42.              if (sequence.size >= expected.size && sequence.takeLast(expected.size) ==
      expected) {
43.            triggerAction()
44.            sequence.clear() }    }
45.     private fun triggerAction() {
46.        flutterEngineRef?.let { engine ->
47.            MethodChannel(engine.dartExecutor.binaryMessenger, CHANNEL)
48.                .invokeMethod("onWidgetDataReceived", "ACTIVE")  }     }
49.     private fun sendDataToFlutter(intent: Intent) {
50.        handleIntent(intent)?.let { data ->
51.            flutterEngineRef?.let { engine ->
52.                MethodChannel(engine.dartExecutor.binaryMessenger, CHANNEL)
53.                    .invokeMethod("onWidgetDataReceived", data)
54.            }     } }
55.     private fun handleIntent(intent: Intent?): String? {
56.        return if (intent?.action == Intent.ACTION_VIEW) {
57.            intent.data?.getQueryParameter("data"  } else     null  }    }}
```

Issue:

```
PS D:\Projects\resqmob> flutter run
Launching lib\main.dart on sdk gphone64 x86 64 in debug mode...
Note: C:\Users\saifi\AppData\Local\Pub\Cache\hosted\pub.dev\android_intent_plus-5.3.1\android\src\main\java\dev\fluttercommunity\plus\androidintent\MethodCallHandlerImpl.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
warning: [options] source value 8 is obsolete and will be removed in a future release
warning: [options] target value 8 is obsolete and will be removed in a future release
warning: [options] To suppress warnings about obsolete options, use -Xlint:-options.
3 warnings
warning: [options] source value 8 is obsolete and will be removed in a future release
warning: [options] target value 8 is obsolete and will be removed in a future release
warning: [options] To suppress warnings about obsolete options, use -Xlint:-options.
3 warnings
e: file:///D:/Projects/resqmob/android/app/src/main/kotlin/com/example/resqmob/MainActivity.kt:28:36 Unresolved reference 'handleIntent'.
e: file:///D:/Projects/resqmob/android/app/src/main/kotlin/com/example/resqmob/MainActivity.kt:73:9 Unresolved reference 'handleIntent'.
e: file:///D:/Projects/resqmob/android/app/src/main/kotlin/com/example/resqmob/MainActivity.kt:73:31 Cannot infer type for this parameter. Please specify it explicitly.
e: file:///D:/Projects/resqmob/android/app/src/main/kotlin/com/example/resqmob/MainActivity.kt:73:37 Cannot infer type for this parameter. Please specify it explicitly.

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':app:compileDebugKotlin'.
> A failure occurred while executing org.jetbrains.kotlin.compilerRunner.GradleCompilerRunnerWithWorkers$GradleKotlinCompilerWorkAction
   > Compilation error. See log for more details

* Try:
> Run with --stacktrace option to get the stack trace.
> Run with --info or --debug option to get more log output.
> Run with --scan to get full insights.
> Get more help at https://help.gradle.org.

BUILD FAILED in 26s
Running Gradle task 'assembleDebug'...                          27.0s
Error: Gradle task assembleDebug failed with exit code 1
PS D:\Projects\resqmob>
```

➔ **Build Failed after adding HomeScreen Widget Button For Alert Trigger.**

Modified Code (After Fix)

We added a **handleIntent** Fuction to handle the Intent in **MainActivity.kt** File.

```
1.  private fun handleIntent(intent: Intent?): String? {
2.          return if (intent?.action == Intent.ACTION_VIEW) {
3.              intent.data?.getQueryParameter("data")
4.          } else {
5.              null
6.          }
7.      }
```

Retest Result (Console Output)

```
PS D:\Projects\resqmob> flutter run
Launching lib\main.dart on sdk gphone64 x86 64 in debug mode...
Running Gradle task 'assembleDebug'...                          12.8s
√ Built build\app\outputs\flutter-apk\app-debug.apk
Installing build\app\outputs\flutter-apk\app-debug.apk...       730ms
I/flutter (11413): [IMPORTANT:flutter/shell/platform/android/android_context_gl_impeller.cc(94)] Using the Impeller rendering backend (OpenGLES).
D/FlutterGeolocator(11413): Attaching Geolocator to activity
D/FlutterGeolocator(11413): Creating service.
D/FlutterGeolocator(11413): Binding to location service.
I/WindowExtensionsImpl(11413): Initializing Window Extensions, vendor API level=9, activity embedding enabled=true
Syncing files to device sdk gphone64 x86 64...                  123ms

Flutter run key commands.
r Hot reload.
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).

A Dart VM Service on sdk gphone64 x86 64 is available at: http://127.0.0.1:58492/sYPMataGkX0=/
I/example.resqmob(11413): Compiler allocated 5042KB to compile void android.view.ViewRootImpl.performTraversals()
D/FlutterGeolocator(11413): Geolocator foreground service connected
D/FlutterGeolocator(11413): Initializing Geolocator services
D/FlutterGeolocator(11413): Flutter engine connected. Connected engine count 1
I/Choreographer(11413): Skipped 35 frames!  The application may be doing too much work on its main thread.
The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9101?uri=http://127.0.0.1:58492/sYPMataGkX0=/
D/WindowLayoutComponentImpl(11413): Register WindowLayoutInfoListener on Context=com.example.resqmob.MainActivity@bfaf224, of which baseContext=android.app.ContextImpl@3112c54
D/FirebaseAuth(11413): Notifying id token listeners about user ( bEICdScHQzTnUMnja7EEKBJoZ2o1 ).
I/FLTFireBGExecutor(11413): Creating background FlutterEngine instance, with args: [--enable-dart-profiling]
I/flutter (11413): [IMPORTANT:flutter/shell/platform/android/android_context_gl_impeller.cc(94)] Using the Impeller rendering backend (OpenGLES).
D/FLTFireContextHolder(11413): received application context.
I/Choreographer(11413): Skipped 33 frames!  The application may be doing too much work on its main thread.
D/FlutterGeolocator(11413): Geolocator foreground service connected
D/FlutterGeolocator(11413): Initializing Geolocator services
D/FlutterGeolocator(11413): Flutter engine connected. Connected engine count 2
I/FLTFireMsgService(11413): FlutterFirebaseMessagingBackgroundService started!
```

➔ **Build Completed Successfully.**

Chapter 6. Deployment

6.1 Deployment Architecture & Platform Selection

- ➔ **Frontend Client (Flutter Framework):** The application is developed using Flutter, enabling a single codebase to deploy natively compiled applications across multiple platforms.
- ➔ **Mobile (Primary Target):** Released as a production-ready Android APK for wide accessibility.
- ➔ **Web (Progressive Web App - PWA):** Deployed as a PWA for quick demonstration and access via web browsers without installation.
- ➔ **Desktop (Admin Tool):** Packaged as a native Windows executable (.exe) to provide administrators with a dedicated tool for monitoring and management.

Backend & Infrastructure (Google Firebase)

The entire backend is powered by Firebase, ensuring scalability, reliability, and ease of development.

- ➔ **Authentication:** Firebase Auth handles user sign-up and login.
- ➔ **Database:** Cloud Firestore manages real-time data for users, alerts, and dynamic content.
- ➔ **Serverless Functions:** Firebase Cloud Functions handle complex backend logic.
- ➔ **Messaging:** Firebase Cloud Messaging (FCM) facilitates push notifications.
- ➔ **Hosting:** Firebase Hosting serves the web PWA and static assets.

Additional Services:

- ➔ **Google Cloud Platform (GCP)**: Provides underlying infrastructure and services for Firebase and additional APIs.

6.2 Deployment Process

➔ **Development & Build:**

- ◆ The application was built using the Flutter SDK with the Dart programming language.
- ◆ Firebase and Supabase services were integrated via their official Flutter plugins.
- ◆ Platform-specific release builds were generated:
  - APK for Android.
  - Web-compiled output for the PWA.
  - Windows executable for the desktop.

➔ **Testing**

- ◆ A comprehensive testing strategy was employed to ensure application stability and correctness.
- ◆ White-box testing techniques, including Control Flow Graph (CFG) analysis, were used to validate internal structures.
- ◆ Black-box testing techniques, such as Boundary Value Analysis (BVA), were applied to verify functionality against requirements.
- ◆ Unit and widget tests validated individual components, while integration tests ensured module interoperability.

➔ **Release**

- ◆ **Android:** The release APK is distributed via Google Drive for direct installation and is submitted to the Google Play Store for public availability.
- ◆ **Web PWA:** The web application is automatically deployed and hosted on Firebase Hosting.

- **Desktop:** The Windows executable is packaged and distributed for direct download and use by administrators.

➔ **Database & Infrastructure Setup:**

- All necessary Cloud Firestore collections (Users, Alerts, Posts, etc.) were configured with appropriate security rules.
- Supabase PostgreSQL tables were set up to manage user profiles.

➔ **Monitoring & Analytics:**

- Firebase Crashlytics is integrated for real-time error reporting and crash monitoring.
- Firebase Analytics provides insights into user engagement and feature usage, informing future development priorities.

6.3 Deployed Product URL

➔ **GitHub Repository**: https://github.com/XhAfAn1/Emergency-SOS-System

➔ **Demo APK Link**: ResQMob.apk - Google Drive

➔ **Web App:** https://resq-mob.web.app/

➔ **Desktop App: resQmob.exe**

Chapter 7. Conclusion

7.1 Learnings

The ResQMob project provided critical, hands-on experience in modern software engineering:

- ➔ **Full-Stack Proficiency:** Mastered Flutter for cross-platform development and integrated Firebase & Google cloud Service for a scalable, cloud-native backend.
- ➔ **Agile Execution:** Successfully applied Agile methodology with iterative sprints to manage evolving requirements and ensure timely delivery.
- ➔ **Rigorous QA:** Implemented formal White-Box and Black-Box testing methodologies to ensure system reliability and identify defects early.
- ➔ **Architecture for Scale:** Designed a system prioritizing low latency and security, making key decisions like data denormalization for performance.
- ➔ **User-Centric Design:** Developed a deep understanding of designing intuitive UI/UX for high-stress, critical scenarios.

7.2 Limitations

The current prototype, while functional, has acknowledged constraints that bound its real-world application:

- ➔ Connectivity Dependency: Lacks a robust offline mode, requiring a stable internet connection for all core functions.
- ➔ Preliminary Predictive Model: The ML-based risk prediction is based on limited data, affecting its accuracy and reliability.
- ➔ Isolated Ecosystem: No direct integration with official emergency services (e.g., 999), limiting response capability.
- ➔ Platform Exclusivity: Currently available only on Android, excluding iOS users.
- ➔ Unproven at Sc7.4ale: System performance and stability under mass concurrent usage remain untested.

7.3 Future Plan

The current ResQMob implementation establishes a foundational prototype. To transition into a scalable, nationwide emergency ecosystem, development will focus on three strategic pillars:

Core System Resilience & Integration

- ➔ Achieve direct API integration with national emergency services (e.g., 999) to reduce response latency.
- ➔ Implement an SMS-based offline fallback mechanism to ensure reliability without internet connectivity.
- ➔ Develop and deploy a full iOS version to ensure platform inclusivity and maximize user reach.

Advanced Intelligence & Automation

- ➔ Enhance predictive capabilities by integrating a Machine Learning model (e.g., TensorFlow Lite) for proactive risk forecasting using contextual data.
- ➔ Introduce a crowd-based verification and reputation system to automatically detect and filter false alerts, ensuring system reliability.

User Safety & Operational Control

- ➔ Launch Guardian Mode for temporary live location sharing and a Discreet Emergency Mode with a camouflage UI for covert situations.
- ➔ Develop advanced administrative monitoring tools, including live media streaming and remote device security features ("blackout mode") for critical incidents.

Implementation Phasing:

1. **Short-Term:** Offline SOS, official service integration, iOS version.
2. **Medium-Term:** ML-powered predictions, automated alert verification.
3. **Long-Term:** Advanced admin controls and public safety API for third-party integration.

This roadmap will transition ResQMob from a prototype to a reliable, nationwide public safety platform.

7.4 Cost

1. **AI Assistant (Gemini API)**

- ➔ Assume **Gemini Pro** or **Gemini 1.5 Flash** integration for SOS guidance & chatbot.
- ➔ Pricing (approx, based on Google AI):
  - ◆ **Text requests:** $0.00025 – $0.0005 per 1K tokens.
  - ◆ **1M requests/month (avg 500 tokens each)** → ~$125 – $250 (≈ 15,000 – 25,000 BDT).

2. Firebase (Scalable Backend)

- ➔ **Firestore Database**:
  - ◆ 50K document writes = Free
  - ◆ Beyond → $0.18 per 100K writes (~20 BDT).
- ➔ **Authentication**:
  - ◆ Free for first 50K/month.
- ➔ **Cloud Functions**:
  - ◆ 2M free/month, then $0.40 per million calls.
- ➔ **Storage**:
  - ◆ $0.026/GB/month (~3 BDT/GB).

Small to medium scale → **~$50–100/month (5,000–10,000 BDT)**.

3. Supabase (Postgres DB + APIs)

- ➔ Free tier: up to 500MB DB.
- ➔ Paid plan: $25–50/month (≈ 3,000–6,000 BDT).

4. SMS Service (Amar Sheba)

➔ **Per SMS Cost:** 0.3 BDT (no masking).

➔ If **10,000 SOS alerts/month →**
10,000 × 0.3 BDT = 3,000 BDT/month.

➔ If scaled to **100,000 alerts/month →**
100,000 × 0.3 BDT = 30,000 BDT/month.

5. Miscellaneous (Server, Monitoring, Updates)

➔ Domain & Hosting → ~1,000 BDT/year.

➔ Maintenance/Developer time → depends on team.

Component	Small Scale (Pilot)	Medium Scale	Large Scale
Gemini API (AI)	5,000 BDT	15,000 BDT	25,000+ BDT
Firebase	2,000 BDT	5,000 BDT	10,000 BDT
Supabase	Free	3,000 BDT	6,000 BDT
SMS (Amar Sheba)	3,000 BDT (10K msgs)	10,000 BDT	30,000 BDT
Miscellaneous	1,000 BDT	2,000 BDT	5,000 BDT
Total / month	~11,000 BDT	~35,000 BDT	~75,000+ BDT

Sign-Off,

We, the undersigned, acknowledge that this document accurately represents the requirements for the ResQMob system and authorize the development team to proceed with the design and implementation phases.

Submitted By **(Team Dynasty)**:

Abrar Khatib Lajim

Md. Saiful Islam

Umme Mukaddisa

Approved By (Stakeholder Representative / Instructor):

Yasin Sazid
Date: 4 Sept 2025

End of Document

Chapter 8. References

Road safety ML : safety road model test final version

Papers :

- Spatial Distribution of Street Crime in the Thanas of DCC area... | Download Scientific Diagram
- https://www.ijcsit.com/docs/Volume%207/vol7issue3/ijcsit20160703106.pdf?fbclid=IwY2xjawMmrjlleHRuA2FlbQIxMQABHmXCqBj8diqpGTpB1KdGENln1R6QmPlk1RwWz_G7n2T9wQP80HmIBkVhycnH_aem_H5tDaz9YIobFFZloN-ZgyA
- Dangerous Prediction in Roads by Using Machine Learning Models | IIETA

Video_Demonstration & More:

CSE412-Project