**Software Requirements**

**Specification**

**For**

**SafeHer**

**~G521**

Team Identities

23BD1A05BD

23BD1A663R

23BD1A6606

23BD1A663D

23BD1A662L

# Table of Contents

[Table of Contents ii](#_TOC_250033)

[Revision History ii](#_TOC_250032)

1. [Introduction 1](#_TOC_250031)
   1. [Purpose 1](#_TOC_250030)
   2. [Document Conventions 1](#_TOC_250029)
   3. [Intended Audience and Reading Suggestions 1](#_TOC_250028)
   4. [Product Scope 1](#_TOC_250027)
   5. [References 1](#_TOC_250026)
2. [Overall Description 2](#_TOC_250025)
   1. [Product Perspective 2](#_TOC_250024)
   2. [Product Functions 2](#_TOC_250023)
   3. [User Classes and Characteristics 2](#_TOC_250022)
   4. [Operating Environment 2](#_TOC_250021)
   5. [Design and Implementation Constraints 2](#_TOC_250020)
   6. [User Documentation 2](#_TOC_250019)
   7. [Assumptions and Dependencies 3](#_TOC_250018)
3. [External Interface Requirements 3](#_TOC_250017)
   1. [User Interfaces 3](#_TOC_250016)
   2. [Hardware Interfaces 3](#_TOC_250015)
   3. [Software Interfaces 3](#_TOC_250014)
   4. [Communications Interfaces 3](#_TOC_250013)
4. [System Features 4](#_TOC_250012)
   1. [System Feature 1 4](#_TOC_250011)
   2. [System Feature 2 (and so on) 4](#_TOC_250010)
5. [Other Nonfunctional Requirements 4](#_TOC_250009)
   1. [Performance Requirements 4](#_TOC_250008)
   2. [Safety Requirements 5](#_TOC_250007)
   3. [Security Requirements 5](#_TOC_250006)
   4. [Software Quality Attributes 5](#_TOC_250005)
   5. [Business Rules 5](#_TOC_250004)
6. [Other Requirements 5](#_TOC_250003)

[Appendix A: Glossary 5](#_TOC_250002)

[Appendix B: Analysis Models 5](#_TOC_250001)

[Appendix C: To Be Determined List 6](#_TOC_250000)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

1. **Introduction**

**Abstract**

SafeHer is a mobile safety application specifically designed to support and protect women in emergency SafeHer is a mobile app designed to improve women's safety by offering a fast and dependable way to alert trusted contacts in emergencies. With a single press of a panic button or a simple gesture like shaking the phone, the app instantly sends an SMS alert containing the user’s live GPS location to pre-selected emergency contacts.

What sets SafeHer apart is its offline functionality — it works even without an internet connection, making it highly reliable in remote areas or places with poor network coverage. This ensures that help can be reached regardless of connectivity.

Users can manage their emergency contact list, customize the alert message, and choose their preferred trigger method. The app uses local storage to save contact information securely on the device, ensuring privacy and availability at all times.

In addition to core features, SafeHer can access GPS services, send SMS alerts, and display nearby help using maps. For users with internet access, it also integrates with services like MongoDB to log emergency events, analyze usage patterns, and report crashes to improve performance.

Whether at home, traveling, or in unfamiliar surroundings, SafeHer gives women a simple yet powerful tool to stay safe and connected when it matters most.

**1.1 Purpose**

The purpose of this **Software Requirements Specification (SRS)** is to clearly define what the **SafeHer – Women Security SMS Service App** is expected to do, how it should function, and the environment in which it will operate. This document serves as a shared point of reference for **developers, designers, testers, and stakeholders**, ensuring everyone involved in the project is aligned with the same vision and objectives.

This document helps:

* Provide **clear and structured guidance** on the features and functionality of the SafeHer app
* Avoid **confusion or misunderstandings** regarding how the system should behave
* Set **measurable expectations** for **reliability, usability, performance**, and **offline capability**
* Promote **smooth collaboration** among technical and non-technical teams involved in development
* **In simple terms, this SRS covers:**
* What the **SafeHer app** is and what it aims to solve
* The **core features** and how they are expected to work (like SOS triggering, GPS, SMS, offline behavior)
* How the app should behave in **different environments**, such as with or without internet access
* The **technical and non-functional requirements** necessary to make the app reliable, efficient, and user friendly

**1.2 Document Conventions**

To keep this document clear, organized, and easy to follow, the following formatting and naming conventions are used throughout:

· Bold Text is used to highlight important terms, system components, screen names, or section headings.

· Italic Text is used for emphasis or to refer to UI elements, buttons, trigger options, or labels (e.g., SOS Button, Settings screen, Few other gestures).

· Monospaced Text is used to indicate code snippets, file paths, API names, or commands (e.g., sendSOS(), SafeHerService).

· Bullet points and numbered lists are used to break down complex information for better clarity and readability Throughout this document:

Throughout this document:

· **“User”** refers to the person using the **SafeHer mobile application**, typically women who want to quickly alert trusted contacts in case of danger.

· **“System”** refers to the **SafeHer mobile app** as a whole, including all its modules such as SMS sending, GPS tracking, trigger detection, and background service.

· **“Trigger”** refers to any action that activates the SOS alert, such as shake gesture, power button press, or tapping the SOS button on the screen.

· **“Emergency Contacts”** refers to the list of phone numbers stored by the user that receive the SOS message.

**1.3 Intended Audience**

This document is intended for **everyone involved in the planning, design, development, testing, deployment, and usage** of the **SafeHer mobile application**. It serves as a comprehensive guide to help each role understand the app’s **goals, features, system behavior, and technical requirements**.

The primary audience includes:

· **Developers** – To understand the technical specifications, system architecture, and integration of native modules such as SMS, GPS, and gesture/power button triggers, especially under offline conditions.

· **UI/UX Designers** – To create a clear, accessible, and stress-free user interface suited for emergency situations where speed and simplicity are critical.

· **QA Testers** – To develop test plans that simulate real-world emergency scenarios, validate offline and online SOS functionality, and ensure system reliability.

· **Project Managers** – To oversee timelines, manage priorities like offline-first behavior, allocate resources, and ensure progress aligns with safety-focused objectives.

· **Stakeholders and Product Owners** – To verify that the app addresses user needs, aligns with the vision of empowering women's safety, and provides tangible real-world value.

· **Security Analysts** – To assess how the system handles sensitive data (emergency contacts, GPS location), and to review permissions and background processes to ensure privacy and security compliance.

· **End-Users (indirectly)** – While not the primary readers of this document, their safety needs and usage conditions are central to all requirements, shaping the design and functionality of the application.

**1.4 Product Scope**

**SafeHer** is a mobile application focused on enhancing **women's safety** by enabling users to send **emergency SOS alerts** quickly and discreetly—even **without internet access**. The system combines **gesture recognition**, **hardware triggers**, **GPS tracking**, and **native SMS capabilities** to notify trusted contacts with the user’s **real-time location** during distress.

The primary goal of **SafeHer** is to provide an **offline-capable, fast-response personal safety tool**, especially useful in areas with poor network connectivity or where users cannot access traditional emergency services quickly.

### **SafeHer is especially helpful for people who want to:**

* Instantly alert family or friends in dangerous or uncomfortable situations
* Share their **real-time location** via SMS during emergencies
* Use **non-obvious activation methods** (shake/power button) for discreet alerts
* Stay protected while traveling, commuting alone, or living independently

#### Key Features of SafeHer:

* **Emergency SOS Trigger**  
  Users can send an SOS alert with their location via SMS by:

1. Tapping a large **SOS button**
2. **Shaking the phone**
3. **Pressing the power button 3 times**

* **Offline Functionality**  
  Works without internet by using the phone’s **native GPS and SMS services**
* **Live Location Sharing**  
  Sends a **Google Maps link** to emergency contacts to view user’s location
* **Contact Management**  
  Users can add, edit, or delete up to **5 emergency contacts** who will receive the alert
* **Silent Mode & Fake Call**  
  SOS can be triggered silently, and optionally a **fake incoming call** can play to distract or comfort the user
* **Custom Emergency Message**  
  Users can define their own alert message sent via SMS

**1.5 References**

* **React Native Documentation** – https://reactnative.dev/docs/getting-started  
  (Frontend framework used for building cross-platform mobile apps)
* **Mongo DB(Atlas) --** [**https://www.mongodb.com/**](https://www.mongodb.com/)
* **(**Storing the user’s information including the login, instances of the user)
* **React Navigation Docs** – https://reactnavigation.org/docs/getting-started/  
  (Routing and navigation in React Native apps)
* **Geolocation & Permissions (React Native APIs)** –  
  https://reactnative.dev/docs/permissionsandroid  
  (Accessing GPS location, managing runtime permissions)
* **SQLite --** <https://www.sqlite.org/>

(Used to store the emergency contacts, Message content, sync status and GPS location)

* **SMS and Call Integration** –
* · react-native-sms / react-native-sms-x (for sending SMS)
* react-native-call or react-native-immediate-phone-call (for emergency calls)
* **Shake Gesture Detection** –  
  <https://www.npmjs.com/package/react-native-shake>  
  (Detecting phone shake gestures for triggering safety features)

**2. Overall Description**

**2.1 Product Perspective**

To support the development of the SafeHer mobile application and ensure alignment with best practices, several technical resources and industry standards serve as foundational references. These tools, frameworks, and guidelines not only guide the development process but also help maintain quality, security, and usability throughout the system lifecycle.

Key components of Ectopia include:

**Frontend Development**

* **React Native Documentation** – https://reactnative.dev/docs/getting-started  
  (Used for building the mobile UI and native features like camera, GPS, and SMS integration)

**Backend & APIs**

* **Node.js Documentation** – <https://nodejs.org/en/docs>  
  (JavaScript runtime environment optionally used for backend logic, API integrations, or custom services)
* **Express.js Documentation** – <https://expressjs.com/>  
  (Optional lightweight web framework used to create REST APIs if backend server is needed)

**Database & Authentication**

* **Mongo DB(Atlas)** – [**https://www.mongodb.com/**](https://www.mongodb.com/)  
  (MongoDB Atlas is a cloud-based NoSQL database service used to securely store and manage application data with high scalability and flexibility.)

**SafeHer is built as a mobile-first platform using React Native**, designed specifically for Android (and optionally iOS), with cloud-based services powered by **MongoDB**. The system is optimized for **offline functionality**, ensuring users can access core safety features even without an internet connection. Its **modular architecture** supports easy maintenance, real-time updates, and future expansion—including gesture-based triggers, emergency SMS/calls, GPS tracking, and secure user data handling.

**2.2 Product Functions**

· **User Registration and Login**

· **Onboarding and Personalization**

· **Gesture-Based Emergency Trigger**

· **Offline Emergency Actions**

· **GPS Location Tracking**

· **Emergency Contact Management**

· **Safety Mode in Background**

· **Settings & Personal Profile**

· **Security & Session Management**

**2.3 User Classes and Characteristics**

The SafeHer mobile application is intended to serve a wide spectrum of female users who may face safety risks in everyday life. The user base is divided into distinct classes based on lifestyle, environment, and use-case needs. Each class shares common characteristics that influence how they interact with the app’s features.

### **1. General Women Users**

Women of all ages needing quick access to SOS features for safety. Require offline functionality and discreet activation methods.

### **2. Students (School/College)**

Young women commuting or on campus. Need fast, silent SOS (e.g., shake or power button) and simple UI.

### **3. Working Professionals**

Women traveling for work or during late hours. Prefer customizable alerts and background-running services.

### **4. Elderly Women**

Older women living alone. Require large buttons, simple navigation, and reliable offline alerts.

### **Common Traits**

· Depend on GPS + SMS (no internet needed)

· Prioritize privacy and security

· Need one-tap or gesture-based emergency alerts

**2.4 Operating Environment**

| **Category** | **Details** |
| --- | --- |
| **Platform** | Android mobile devices (phones only) |
| **OS Requirement** | Android 7.0 (Nougat) or higher |
| **Hardware Required** | - GPS sensor- Accelerometer sensor (for shake detection)- Cellular network for SMS |
| **Connectivity** | Works **without Internet** – uses **SMS** and **GPS** only |
| **App Permissions** | - Location access- SMS access- Background service permission- Contact access |
| **Trigger Methods** | - Shake gesture- Power button press (e.g., 3 times)- SOS button tap |
| **User Interface** | Minimal UI – large SOS button, simple settings |
| **Battery Efficiency** | Runs in background with **low power usage** |

**2.5 Design and Implementation Constraints**

The SafeHer app is subject to several design and implementation constraints due to its safety-critical and offline nature:

* **Platform Constraint**: The application is developed exclusively for Android devices. iOS is not supported due to limitations in background services and SMS sending.
* **Offline Functionality**: The app must function completely offline. It relies on native GPS and SMS services, without requiring internet access.
* **Permissions Dependency**: Full functionality requires users to grant SMS, location (GPS), and background activity permissions during setup.
* **Hardware Constraints**: The app depends on the device having a working GPS module, an accelerometer sensor (for shake detection), and an active SIM card with SMS capability.
* **Battery Usage Constraint**: The application must operate efficiently in the background with minimal battery consumption. Continuous polling or active location tracking is avoided.
* **Contact Limitation**: To ensure reliable and quick SMS dispatch, users can add a maximum of five emergency contacts.
* **Trigger Method Constraint**: The SOS can only be triggered through supported methods, including a visible UI button, shake gesture, or power button press (which may vary based on device manufacturer support).
* **Security Constraints**: All emergency contact data is stored locally and securely on the device. No cloud storage or sync is used to protect user privacy.
* **UI Simplicity Constraint**: The user interface must remain minimal and responsive, avoiding complex layouts or animations that could slow down performance during emergencies.

**2.6 User Documentation**

· **Register & Setup**

Open the app → Grant SMS & Location permissions

Add up to **5 emergency contacts**

Set a **custom emergency message** (e.g., “I’m in danger. Please help!”)

· **Triggering Emergency Alert**

Press **power button 3 times**, or **Shake** the phone, or Tap the **big SOS button**  
→ An **SMS with your GPS location** is instantly sent to saved contacts

· **Offline Operation**

No internet needed

Works using **device GPS + native SMS**

· **Silent Mode & Fake Call**

Sends alerts **silently**

Optionally triggers a **fake call** as distraction

· **Contact & Settings Management**

Add/edit/delete emergency contacts

Choose preferred SOS trigger

Test SOS feature anytime

**2.7 Assumptions and Dependencies**

· **Device Permissions**:  
It is assumed that the user will grant **location** and **SMS** permissions for the app to function.

· **SMS Functionality**:  
The app depends on the device’s **native SMS service** to send alerts. No internet is required.

· **Location Accuracy**:  
Accurate GPS location depends on the device’s **location services** being enabled and functioning properly.

· **Battery & Background Services**:  
It is assumed that the device has sufficient **battery** and allows **background services** so the app can run continuously.

· **Android Device Compatibility**:  
The app is built for **Android phones only** (using React Native). It may not function on iOS unless separately developed.

· **Contact Storage**:  
Emergency contact data is stored **locally** on the device and may be lost if the app is uninstalled.

· **3. External Interface Requirements**

**3.1 User Interfaces**

· **Home Screen**

· **Contacts Management Screen**

· **Settings Screen**

**3.2 Hardware Interfaces**

The **SafeHer** app relies on the following hardware components available on most Android smartphones:

* **GPS Module**

Required to capture real-time user location.

Works without internet (uses satellite GPS data).

* **Accelerometer & Gyroscope**

Detects shake gestures to trigger emergency alerts.

* **Power Button Hardware**

Used to detect quick multi-press (e.g., press 3 times) to silently trigger SOS.

* **SIM Card and GSM Module**

Required to send SMS to emergency contacts.

Ensures offline operation via mobile network.

* **Battery & Power Management**

App is optimized to run silently in the background with minimal power consumption.

**3.3 Software Interfaces**

· **Frontend Interface**

Developed using **React Native** for cross-platform Android support.

Handles user interface elements like the SOS button, settings, and contact management.

Uses React Native libraries to access hardware features such as GPS, accelerometer, and SMS.

· **Device Location Interface**

Uses **React Native Location** or **@react-native-community/geolocation** to get the user's current GPS coordinates.

Works without internet by relying on the phone’s built-in GPS hardware.

· **SMS Interface**

Integrates with **React Native SMS** or native modules to send emergency text messages directly to pre-set contacts.

Embeds live location links (e.g., Google Maps) in the SMS.

· **Shake & Button Trigger Interface**

Uses **react-native-shake** for detecting shake gestures.

For power button detection, native Android modules or custom code can be used to listen for hardware button presses (e.g., 3 times)

**3.4 Communication Interfaces**

· **Frontend–Backend Communication:** The **React Native frontend** interacts directly with **MongoDB services** via SDKs (Realtime Database). All requests are secured over **HTTPS** using MongoDB’s built-in authentication tokens.

· **Emergency SMS Communication:** SafeHer uses the **device’s native SMS manager** to send emergency messages to pre-set contacts. The message includes the **user's GPS location** as a Google Maps link. This works **offline** and only requires SMS permissions for once.

· **Location Services Communication:** The app accesses the device’s **GPS** hardware via React Native Location APIs. No external server is required for location fetching, enabling **offline support**.

· **Database Communication:** SafeHer uses **MongoDB & SQLite** to store user data, emergency contact lists, and app preferences. All operations are performed through MongDB with **real-time syncing** when online.

· **Security:** All communication uses **HTTPS**, and **JWT for secure** user sessions.

**4. System Features**

**4.1 Emergency SOS Alert System**

* **Description and Priority**:  
  This is the **core feature** of the SafeHer app. It enables users to send an emergency alert via SMS to trusted contacts, including their real-time GPS location.  
  **Priority**: High
* **Stimulus/Response Sequences**:

1. User triggers SOS via panic button or gesture.
2. App captures real-time GPS coordinates.
3. App composes and sends an SMS to emergency contacts.
4. A beep sound is received to the emergency contacts.

* **Functional Requirements**:

1. Send SOS SMS to emergency contacts.
2. Manual panic button for alert.
3. Include current GPS location in the alert.

**4.2 Contact Management System:**

· **Description and Priority**:  
Allows users to add, edit, or delete up to 5 emergency contacts to whom SOS alerts will be sent.  
**Priority**: High

· **Stimulus/Response Sequences**:

1. User opens contact management screen.
2. Adds or edits contact numbers.
3. Contacts saved for future alerts.

· **Functional Requirements**:

1. Store and manage emergency contacts.

**4.3 Gesture-Based SOS Trigger:**

* **Description and Priority**:  
  Enables a discreet way to trigger SOS using gestures like phone shaking or pressing the power button 3 times or long press button.  
  **Priority**: Medium
* **Stimulus/Response Sequences**:

1. App detects the specific gesture.
2. Triggers SOS alert automatically.

* **Functional Requirements**:

1. Gesture-based emergency activation.

### **4.4 GPS Location Tracking**

* **Description and Priority**:  
  App fetches real-time GPS location of the user and includes it in the SOS message.

**Priority**: High

* **Stimulus/Response Sequences**:

1. SOS is triggered.
2. GPS data is fetched from the device.
3. Coordinates are appended to the alert.

* **Functional Requirements**:

1. Fetch and include current location.

### **4.5 Offline Operation**

* **Description and Priority**:  
  Ensures that all core functionalities (like sending SMS and fetching GPS) work without internet.  
  **Priority**: High
* **Stimulus/Response Sequences**:

1. User triggers SOS in offline mode.
2. App still fetches location and sends SMS using device services.

* **Functional Requirements**:

1. Operate even without internet.

### **4.6 Permission Management**

* **Description and Priority**:  
  Handles all necessary runtime permissions like SMS, location, and background services.  
  **Priority**: High
* **Stimulus/Response Sequences**:

1. On first launch or feature usage, app prompts for permissions.
2. User accepts or denies.
3. App functions based on granted access.

* **Functional Requirements**:

1. Manage and request permissions.

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

* **Real-time Location Updates:** App must capture current location with less than 10-meter accuracy (subject to device GPS capabilities).
* **App Load Time:** Application must initialize and become usable in under 2 seconds on a standard mobile device.
* **Response Time:** Emergency alerts (SMS + GPS location) must be sent within 2–3 seconds of triggering the panic button or gesture.
* **Concurrent Users:** System must handle at least 10,000 active users without performance degradation (Firebase ensures horizontal scalability).
* **Offline Reliability:** Alert system must work without internet, leveraging native SMS protocols.
* **Battery Efficiency:** App should minimize battery usage by optimizing location fetch and background services.

**5.2 Safety Requirements**

1. Emergency features (panic button/gesture) must not fail under low battery, no network, or screen lock conditions.
2. App must send alerts even if the app is minimized or running in the background.
3. Prevent accidental alerts with confirmation triggers or double-press gestures.
4. Include a test mode to avoid accidental SMS during app testing/demonstration.
5. Must follow government/local telecom regulations for emergency SMS sending.

**5.3 Security Requirements**

* **Data Privacy:** All user data (contacts, location, profile) stored securely via MongoDB, SQLite.
* **User Authentication**: JWT Authentication (email/password,) ensures identity protection.
* **Communication Security:** All communication (cloud sync, login) over HTTPS with SSL encryption.
* **Permission Management:** App must request and explain use of sensitive permissions (location, SMS, contacts).

**5.4 Software Quality Attributes**

* **Usability**: Intuitive and minimal UI, accessible to all users (especially in panic situations).
* **Reliability**: Critical features like SMS alerts and location sharing must have 99.9% reliability.
* **Maintainability**: Modular React Native codebase allows quick updates, bug fixes, and feature expansion.
* **Portability**: Single codebase supports both Android and iOS using React Native as a **Cross platform**.
* **Accessibility**: Support for screen readers, large fonts, and color contrast for visually impaired users.

**5.5 Business Rules**

* Only registered users can use the panic feature.
* Up to 5 emergency contacts can be configured per user.
* SMS alerts are rate-limited to 1 alert per 30 seconds to prevent spam or misfires.
* A beep sound is included to alert the included contacts for 1-2 mins.
* Alert history (timestamps, GPS logs) is visible to the user & admin.
* Alerts include user name, device battery percentage (<10%), GPS link, and custom emergency message.

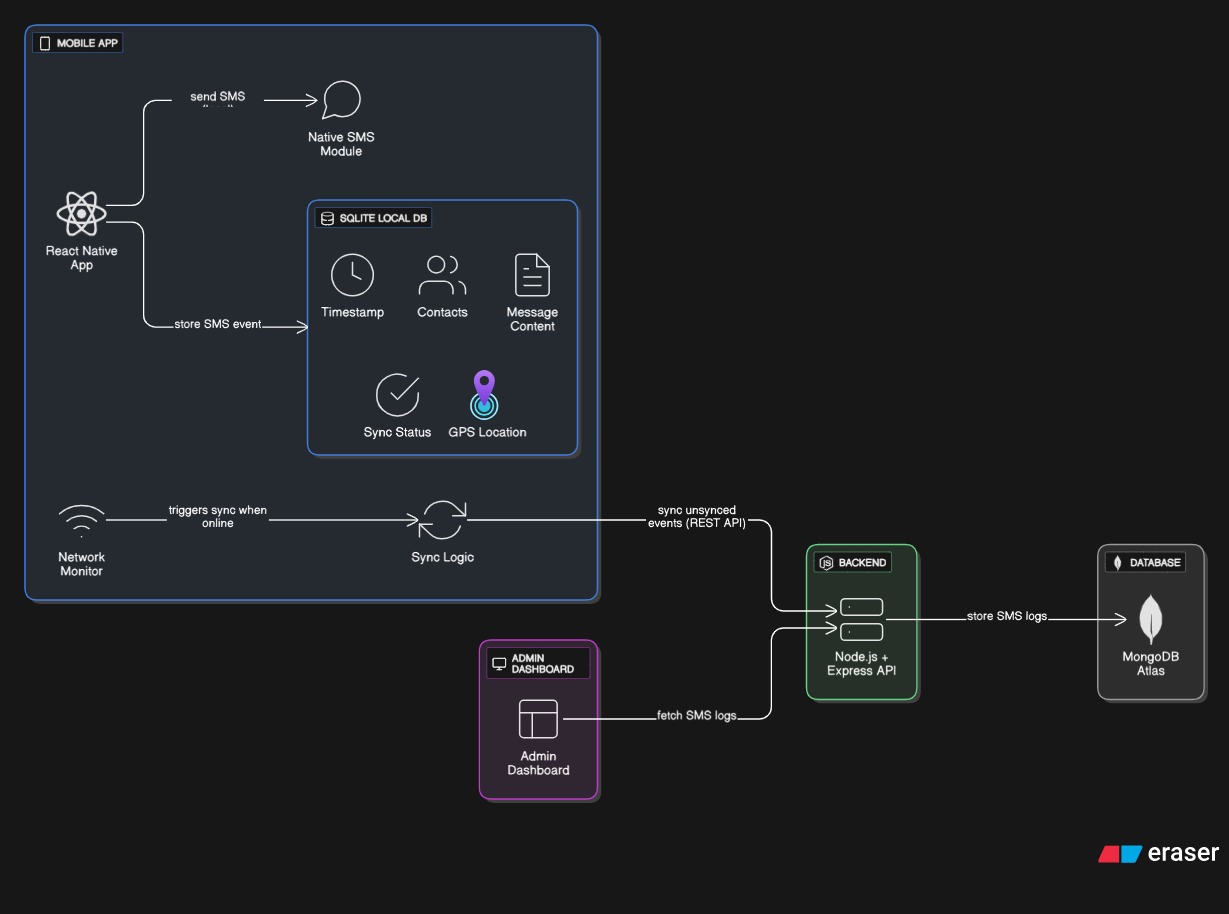
**6. Other Requirements**

* Platform Requirements: React Native (for Android & iOS); MongoDB (Atlas)
* Legal & Compliance: Must comply with privacy laws (Indian IT Act for national use).
* Localization: App should support multi-language UI in at least English, Hindi, and Telugu (future enhancement).

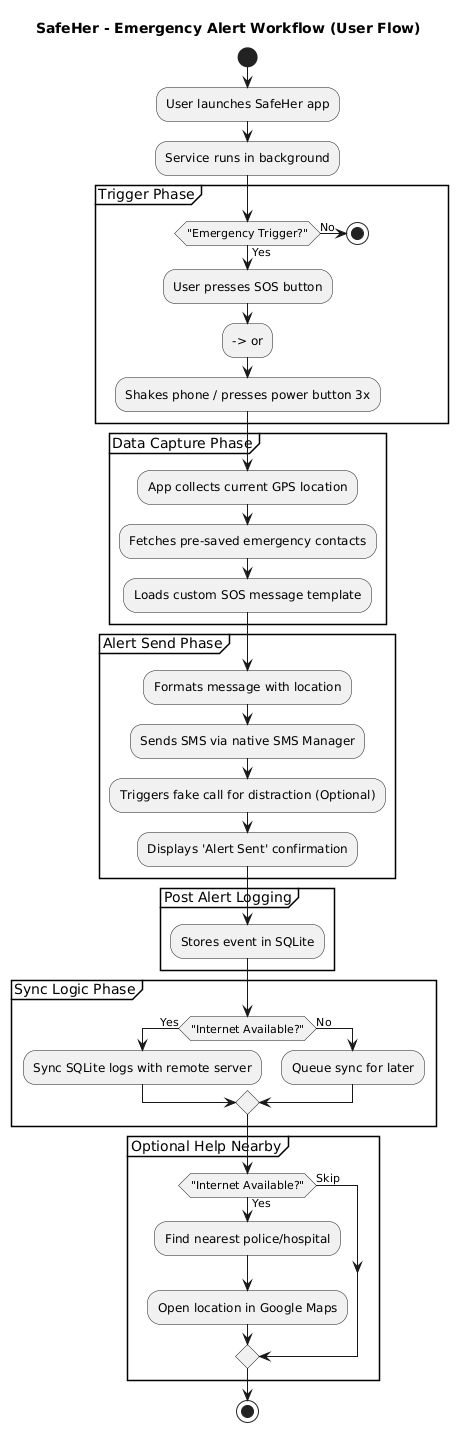
Permissions Required:

* SEND\_SMS and READ\_SMS for alert functionality
* ACCESS\_FINE\_LOCATION for real-time location sharing
* READ\_CONTACTS or manual entry for trusted contact selection

**Architecture Diagram:**



**Work Flow**

****