Software Requirements Specification

1. **Objective:**

To clearly document all functional and non-functional requirements of the system, ensuring a shared understanding among stakeholders and serving as a reference for design, development, testing, and maintenance.

**1. Introduction**

This section introduces the project, explaining its purpose and audience.

### ****1.1 Purpose****

The purpose of this application, **ZapSafe**, is to provide users with a next-gen AI-powered personal safety companion. The app uses AI, ML, and Deep Learning to continuously monitor environmental cues such as threatening sounds, aggressive gestures, sudden crowd movement, or unsafe vibes. Whenever a potential risk is detected, ZapSafe instantly alerts users with alarms, vibrations, and live location sharing to emergency contacts.

Example Functionalities:

* Detect threatening sounds and gestures in real time.
* Allow users to trigger invisible SOS signals.
* Generate fake calls to help users escape dangerous situations.
* Share live GPS location and harassment reports with trusted contacts and nearby community members.

### ****1.2 Document Conventions****

* **AI** – Artificial Intelligence
* **ML** – Machine Learning
* **DL** – Deep Learning
* **GPS** – Global Positioning System
* **SOS** – Emergency Help Signal
* **UI** – User Interface
* **API** – Application Programming Interface

### ****1.3 Intended Audience and Reading Suggestions****

* **Developers** – Implement AI features, mobile UI, and backend APIs.
* **QA Testers** – Validate performance, accuracy of AI models, and app usability.
* **Business Analysts** – Ensure the app meets market and societal safety needs.
* **End Users** – Individuals seeking real-time personal safety.
* **Law Enforcement & NGOs** – Can integrate alerts and reports into their systems.

### ****1.4 Product Scope****

ZapSafe is an AI-powered safety app that proactively detects danger and provides instant alerts. Unlike traditional SOS apps that require manual activation, ZapSafe reacts automatically.

Key features include:

* **AI-powered sound and gesture recognition.**
* **Invisible SOS triggers (power button, volume keys, gestures).**
* **Fake call simulation to escape threats.**
* **Crowd-backed harassment reporting.**
* **Cross-platform compatibility (Android, iOS, Web).**

### ****1.5 References****

* Research on **AI sound & gesture recognition** datasets (UrbanSound8K, ESC-50, MediaPipe).
* **GDPR compliance** for data privacy.
* **PCI-DSS** for secure transactions.
* Industry reports on **personal safety apps** (bSafe, Safetipin, Life360).

## ****2. Overall Description****

### ****2.1 Product Perspective****

ZapSafe is a **multi-user safety platform** integrating individuals, emergency contacts, law enforcement, and the community.

* Works as a **mobile-first app** with optional web dashboard.
* Integrates with third-party services like **Google Maps (GPS tracking)** and **Firebase (authentication/notifications)**.
* Provides **AI-driven proactive detection**, unlike traditional manual SOS apps.

### ****2.2 Product Features****

* **User Registration & Login** (via phone, email, or social login).
* **AI-powered Danger Detection** (sound and gesture recognition).
* **Fake Call Simulation** (custom caller ID, realistic ringtone).
* **Invisible Distress Signal** (hardware triggers, secret words).
* **Live GPS Tracking** (real-time location sharing).
* **Crowd-Backed Harassment Reporting** (notify nearby users).
* **Report Generation & History Logs** (for admins and users).

### ****2.3 User Classes and Characteristics****

* **End Users (Students, Women, Elderly)** – Trigger SOS, receive alerts, view history.
* **Emergency Contacts** – Receive live location and distress updates.
* **Crowd Users** – Get nearby alerts and validate/report incidents.
* **Law Enforcement / NGOs** – Access alerts for quick response.
* **Admin** – Manage system logs, security, and compliance.
* **Primary Stakeholders:** End-users, emergency contacts.
* **Secondary Stakeholders:** Law enforcement, NGOs.
* **Tertiary Stakeholders:** App store providers, researchers (datasets).

### ****2.4 Operating Environment****

* **Mobile App:** Android 9+ / iOS 13+
* **Web Dashboard:** Chrome, Firefox, Safari
* **Database:** PostgreSQL / MongoDB
* **Hosting:** AWS / Google Cloud / Firebase
* **AI Frameworks:** TensorFlow Lite, Scikit-learn

### ****2.5 Design and Implementation Constraints****

* Must support **at least 50,000 concurrent users**.
* AI models must run **in real time (<100ms inference)**.
* All communications must be encrypted (SSL/TLS).
* Compliance with **regional data privacy laws**.
* Continuous monitoring should not cause **excessive battery drain**.

### ****2.6 Assumptions and Dependencies****

* Users must have a **stable internet connection** for real-time alerts.
* Mobile devices must support **GPS, microphone, and motion sensors**.
* Dependence on **Google Maps API** for GPS tracking.
* Dependence on **Firebase/third-party services** for notifications.
* Future versions may integrate Mapbox (offline maps) to reduce reliance on Google Maps

## ****3. Specific Requirements****

### ****3.1 Functional Requirements****

1. **User Registration & Authentication**
   * Register via phone/email/social login.
   * OTP verification for new accounts.
2. **AI-powered Sound & Gesture Monitoring**
   * Detect distress sounds (screams, glass breaking).
   * Detect SOS gestures (hand raise, shaking).
3. **Invisible Distress Signal**
   * Triggered by power button press, volume keys, or keywords.
   * Sends live GPS + audio/video to contacts.
4. **Fake Call Simulation**
   * Generate realistic fake incoming call.
   * Allow scheduling or instant trigger.
5. **Crowd-Backed Harassment Reporting**
   * Nearby users receive silent notifications.
   * Option to validate or escalate to authorities.
6. **Report Generation & History Logs**
   * Users can view past SOS alerts.
   * Admins generate reports for analysis.

### ****3.2 External Interface Requirements****

* **User Interfaces:** Mobile-first design, dark/light mode, minimal clicks.
* **Hardware Interfaces:** Uses GPS, mic, camera, and sensors.
* **Software Interfaces:** Google Maps API, Firebase Auth, Push Notifications.
* **Communication Interfaces:** SMS, FCM push notifications, encrypted REST APIs.

### ****3.3 Performance Requirements****

* Danger detection inference: **<100ms latency**.
* Alerts must reach contacts within **5 seconds**.
* Should handle **100,000+ alerts per day**.
* Continuous monitoring should not drain more than 10–15% battery over 8 hours of active use

### ****3.4 Security Requirements****

* All data encrypted (AES-256).
* Strong authentication with OTP + optional biometric.
* Anonymization of crowd reports.
* Compliance with **GDPR** and **local safety regulations**.
* System must comply with emergency communication regulations in target deployment regions (e.g., TRAI in India, FCC in the US)

### ****3.5 Software Quality Attributes****

* **Availability:** 99.9% uptime.
* **Scalability:** Supports thousands of concurrent users.
* **Usability:** Simple UI, minimal steps to send SOS.
* **Reliability:** Ensure low false positives/negatives in AI detection.

### ****3.6 Other Non-Functional Requirements****

* **Legal Compliance:** Must adhere to regional safety & privacy laws.
* **Localization:** Multi-language support.
* **Accessibility:** Voice commands for visually impaired users.
* Codebase should follow modular architecture (separate AI, UI, backend services) for easier updates

## ****3.7 Use Cases****

**Use Case 1: Silent SOS Trigger**

* **Actor:** User
* **Precondition:** User has configured trusted contacts.
* **Trigger:** User presses the power button 3 times.
* **System Response:** App sends live GPS and audio stream to emergency contacts.
* **Postcondition:** Contacts receive the SOS alert within 5 seconds.

**Use Case 2: Fake Call Simulation**

* **Actor:** User
* **Precondition:** Fake call feature is enabled in settings.
* **Trigger:** User performs secret gesture or key combination.
* **System Response:** App simulates a realistic incoming call with chosen caller ID and ringtone.
* **Postcondition:** User can answer the call to exit the situation naturally.

**Use Case 3: Crowd-Backed Alert**

* **Actor:** User in distress.
* **Precondition:** App location permissions are active.
* **Trigger:** App detects a scream + aggressive gesture.
* **System Response:** App sends silent alerts to nearby users within 200–500m.
* **Postcondition:** Nearby users receive notification and can validate or escalate incident.

## ****3.8 Reliability Requirements****

* The system should recover from a crash or restart within **10 seconds**.
* Backup servers must automatically take over in case of downtime.
* Database backups should be created at least **once every 24 hours**.
* AI models should maintain **>85% precision** and **>80% recall** for threat detection.
* System should tolerate temporary internet outages (≤30s) by queuing alerts for later delivery
* Unit testing for AI modules (accuracy, precision, recall).
* Stress testing (large number of simultaneous SOS alerts).
* Usability testing with real users (ease of triggering SOS).

## ****4. Future Enhancements****

* **Smartwatch Integration:** Allow SOS triggers from wearables.
* **AI Video-based Threat Detection:** Integrate CCTV feeds for recognizing violence or harassment.
* **Safety Heatmaps:** Highlight unsafe areas using crowd-sourced data.
* **Police/NGO API Integration:** Direct reporting to authorities.
* **Multilingual Voice Commands:** Voice-activated SOS for accessibility.
* Federated Learning: Improve AI models on-device without sending raw user data to the cloud, enhancing privacy

## ****5. Appendix****

* **Datasets Used:** UrbanSound8K (sound detection), ESC-50 (environmental sounds), MediaPipe (gesture recognition).
* **Glossary:**
  + **SOS Trigger** – Any action by the user or app to send distress signal.
  + **Crowd Alert** – Notification sent to nearby app users for assistance.
  + **Fake Call** – Simulated incoming call to help user escape threatening situations.
* High-level system architecture diagram (to be included in design phase)
* **FCM – Firebase Cloud Messaging.**
* **DoS – Denial of Service (relevant in risk assessment).**
* **TFLite – TensorFlow Lite (AI framework for on-device inference).**