SafeNow Emergency Response System - Technical Specification

1. System Overview

SafeNow is a Progressive Web Application (PWA) that provides offline emergency response guidance with voice-activated controls and location-based alerts.

1.1 Core Requirements

- **Offline-first architecture**: Full functionality without internet connectivity
- **Voice-activated control**: Hands-free emergency operations
- **Location-based alerts**: GPS-enabled emergency notifications to contacts
- **Secure local storage**: Medical information stored on-device
- **Cross-platform**: Works on mobile and desktop browsers

---

2. Technology Stack

2.1 Frontend

- **Framework**: React 18+ with Vite
- **Language**: JavaScript/TypeScript (JavaScript for simplicity)
- **UI Library**: Tailwind CSS for styling
- **State Management**: React Context API (simple, no Redux needed)

2.2 PWA Technologies

- **Service Worker**: Workbox (for offline caching)
- **Manifest**: Web App Manifest for installability
- **Cache Strategy**: Cache-first for assets, network-first for dynamic data

2.3 Browser APIs

- **Web Speech API**:
  - SpeechRecognition for voice commands
  - SpeechSynthesis for audio guidance
- **Geolocation API**: Real-time location tracking
- **LocalStorage/IndexedDB**: Persistent data storage

- **Notification API**: Push notifications

2.4 Development Tools

- **Build Tool**: Vite
- **Package Manager**: npm
- **Testing**: Vitest (optional for MVP)

---

3. System Architecture

3.1 High-Level Architecture

```
┌─────────────────────────────────────────────────────────┐
│                    User Interface                        │
│  (React Components - Voice, First Aid, Alerts, Profile) │
└─────────────────────┬───────────────────────────────────┘
                      │
┌─────────────────────┴───────────────────────────────────┐
│                  Application Layer                       │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐  │
│  │ Voice        │  │ Location     │  │ Alert        │  │
│  │ Controller   │  │ Service      │  │ Manager      │  │
│  └──────────────┘  └──────────────┘  └──────────────┘  │
│  ┌──────────────┐  ┌──────────────┐                    │
│  │ First Aid    │  │ Storage      │                    │
│  │ Guide Engine │  │ Manager      │                    │
│  └──────────────┘  └──────────────┘                    │
└─────────────────────┬───────────────────────────────────┘
                      │
┌─────────────────────┴───────────────────────────────────┐
│              Browser APIs & Storage                      │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐  │
│  │ Speech API   │  │ Geolocation  │  │ LocalStorage │  │
│  └──────────────┘  └──────────────┘  └──────────────┘  │
│  ┌──────────────┐  ┌──────────────┐                    │
│  │ Service      │  │ Notification │                    │
│  │ Worker       │  │ API          │                    │
│  └──────────────┘  └──────────────┘                    │
└─────────────────────────────────────────────────────────┘
```

3.2 Component Structure

```
src/
├── components/
│   ├── Home/
│   │   ├── HomePage.jsx          # Main dashboard
│   │   └── EmergencyButton.jsx   # Quick action button
│   ├── FirstAid/
│   │   ├── FirstAidList.jsx      # Emergency scenarios list
│   │   ├── FirstAidGuide.jsx     # Step-by-step instructions
│   │   └── firstAidData.js       # Offline data source
│   ├── Voice/
│   │   ├── VoiceControl.jsx      # Voice command interface
│   │   └── VoiceCommands.js      # Command processing logic
│   ├── Alert/
│   │   ├── AlertManager.jsx      # Alert triggering UI
│   │   └── ContactsList.jsx      # Emergency contacts
│   ├── Profile/
│   │   ├── UserProfile.jsx       # Health information form
│   │   └── MedicalData.jsx       # Medical info display
│   └── Layout/
│       ├── Navigation.jsx        # App navigation
│       └── Header.jsx            # App header
├── services/
│   ├── voiceService.js           # Speech recognition logic
│   ├── locationService.js        # Geolocation handling
│   ├── storageService.js         # LocalStorage wrapper
│   ├── alertService.js           # Alert dispatch logic
│   └── notificationService.js    # Push notifications
├── context/
│   ├── AppContext.jsx            # Global app state
│   └── UserContext.jsx           # User data state
├── utils/
│   ├── constants.js              # App constants
│   └── helpers.js                # Utility functions
├── App.jsx                       # Root component
├── main.jsx                      # Entry point
└── service-worker.js             # PWA service worker
```

4. Feature Specifications

4.1 Offline Emergency Guidance

Description: Preloaded first-aid instructions for common emergencies.

Implementation:

- Store first-aid data as JavaScript objects in `firstAidData.js`
- Include scenarios: CPR, bleeding control, burns, fractures, choking, shock
- Each scenario contains:
  - Title and description
  - Step-by-step instructions (array of strings)
  - Warning/cautions
  - Visual indicators (icons)

Data Structure:

```javascript
{
  id: "cpr",
  title: "CPR (Cardiopulmonary Resuscitation)",
  category: "cardiac",
  icon: "heart",
  description: "For unresponsive person not breathing",
  steps: [
    "Check for responsiveness - tap and shout",
    "Call for help or activate emergency alert",
    "Place person on firm surface, face up",
    // ... more steps
  ],
  warnings: ["Do not perform if person is breathing normally"],
  duration: "Continue until help arrives"
}
```

User Flow:

1. User selects emergency type from list
2. App displays step-by-step guide
3. User navigates through steps (Next/Previous buttons)

4. Voice synthesis reads steps aloud (optional)

4.2 Voice-Activated Control

Description: Hands-free operation using voice commands.

Implementation:

- Use Web Speech API (SpeechRecognition)
- Support commands:
  - "Emergency" / "Help" → Trigger emergency alert
  - "Start CPR" → Open CPR guide
  - "Call contact" → Initiate contact alert
  - "Where am I" → Speak current location
  - "Stop listening" → Deactivate voice control

Voice Service Architecture:

```javascript
```

```
class VoiceService {
  constructor() {
    this.recognition = null;
    this.synthesis = window.speechSynthesis;
    this.isListening = false;
  }

  startListening() {
    // Initialize SpeechRecognition
    // Set continuous: true
    // Handle results and errors
  }

  processCommand(transcript) {
    // Match transcript to commands
    // Execute corresponding action
  }

  speak(text) {
    // Use SpeechSynthesis to read text
  }
}
```

User Flow:

1. User taps microphone button or says wake word
2. System starts listening (visual indicator shown)
3. User speaks command
4. System processes and executes command
5. System provides voice/visual feedback

Edge Cases:

- Browser doesn't support Speech API → Show manual controls only
- Noisy environment → Provide visual confirmation before action
- Accidental activation → Require confirmation for critical actions

---

4.3 Location-Based Alerts

Description: Send GPS location to emergency contacts via SMS/WhatsApp.

Implementation:

- Use Geolocation API to get current coordinates
- Format location as Google Maps link
- Generate SMS/WhatsApp deep links for sharing
- Store emergency contacts in LocalStorage

Alert Flow:

```
User triggers alert
   ↓
Get current location (Geolocation API)
   ↓
Format message: "EMERGENCY! I need help at [location link]"
   ↓
Generate deep links for each contact
   ↓
Open SMS/WhatsApp with pre-filled message
   ↓
Log alert in local history
```

Contact Structure:

```javascript
{
  id: "uuid",
  name: "John Doe",
  phone: "+263771234567",
  relationship: "Family",
  preferredMethod: "whatsapp" // or "sms"
}
```

Deep Link Formats:

- SMS: `sms:+263771234567?body=EMERGENCY! ...`
- WhatsApp: `https://wa.me/263771234567?text=EMERGENCY! ...`

User Flow:

1. User triggers emergency alert (button or voice)
2. App requests location permission (if not granted)

3. App retrieves GPS coordinates
4. App generates message with location link
5. App opens messaging apps for each contact
6. User confirms sending from messaging app

Edge Cases:

- Location unavailable → Send alert without location
- No contacts configured → Prompt to add contacts first
- Permission denied → Request permission with explanation

4.4 Secure Health Data Storage

Description: Store critical medical information locally on device.

Implementation:

- Use LocalStorage for simple key-value pairs
- Encrypt sensitive data (optional for MVP)
- Store medical profile including:
  - Full name, age, blood type
  - Allergies
  - Medical conditions
  - Medications
  - Emergency contacts

Data Structure:

```
javascript
```

```
{
 personal: {
  name: "Jane Smith",
  age: 28,
  bloodType: "O+",
  photo: null // base64 if needed
 },
 medical: {
  allergies: ["Penicillin", "Peanuts"],
  conditions: ["Asthma"],
  medications: ["Ventolin inhaler"],
  lastUpdated: "2025-01-15"
 },
 emergencyContacts: [
  // Array of contact objects
 ]
}
```

Storage Service:

```javascript
class StorageService {
 save(key, data) {
  localStorage.setItem(key, JSON.stringify(data));
 }

 get(key) {
  const data = localStorage.getItem(key);
  return data ? JSON.parse(data) : null;
 }

 saveUserProfile(profile) {
  this.save('safeNowProfile', profile);
 }

 getUserProfile() {
  return this.get('safeNowProfile');
 }
}
```

User Flow:

1. User navigates to Profile section

2. User fills in medical information form
3. User saves profile (stored locally)
4. Data persists across sessions
5. User can edit/update anytime

---

5. Progressive Web App (PWA) Implementation

5.1 Service Worker

Purpose: Enable offline functionality and asset caching.

Implementation Strategy:

- Use Workbox library for simplified setup
- Cache static assets (HTML, CSS, JS, images)
- Cache first-aid data
- Network-first for dynamic content

Service Worker Structure:

```javascript
// service-worker.js
import { precacheAndRoute } from 'workbox-precaching';
import { registerRoute } from 'workbox-routing';
import { CacheFirst, NetworkFirst } from 'workbox-strategies';

// Precache build assets
precacheAndRoute(self.__WB_MANIFEST);

// Cache static resources
registerRoute(
  ({request}) => request.destination === 'image',
  new CacheFirst({cacheName: 'images'})
);

// Cache first-aid data
registerRoute(
  ({url}) => url.pathname.includes('/api/'),
  new NetworkFirst({cacheName: 'api-cache'})
);
```

5.2 Web App Manifest

File: `public/manifest.json`

```json
{
  "name": "SafeNow Emergency Response",
  "short_name": "SafeNow",
  "description": "Offline emergency response and first aid guide",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#dc2626",
  "orientation": "portrait",
  "icons": [
    {
      "src": "/icons/icon-192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/icons/icon-512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

5.3 Offline Detection

Implementation:

```javascript
// Detect online/offline status
window.addEventListener('online', () => {
  // Show "You're back online" notification
});

window.addEventListener('offline', () => {
  // Show "You're offline" notification
  // Inform user all features still work
});
```

6. User Interface Design

6.1 Design Principles

- **Large Touch Targets**: Minimum 44x44px for buttons
- **High Contrast**: Ensure readability in stress situations
- **Simple Navigation**: Maximum 3 taps to any feature
- **Clear Visual Feedback**: Confirm all actions
- **Accessibility**: Support screen readers

6.2 Color Scheme

- **Primary (Emergency)**: Red (🟥 #dc2626)
- **Secondary (Info)**: Blue (🟦 #2563eb)
- **Success**: Green (🟩 #16a34a)
- **Warning**: Amber (🟧 #f59e0b)
- **Neutral**: Gray scale

6.3 Key Screens

Home Screen:

- Large "Emergency Alert" button (center)
- Voice control toggle
- Quick access cards: First Aid, Profile, Contacts
- Connection status indicator

First Aid Screen:

- Search/filter emergencies
- Category icons (cardiac, trauma, respiratory, etc.)
- List of emergency scenarios
- Tap to view detailed guide

Guide Detail Screen:

- Emergency title and icon
- Step counter (Step 1 of 8)

- Large text instructions
- Next/Previous navigation
- Voice playback button

Profile Screen:

- Form fields for medical data
- Emergency contacts list
- Add/Edit/Delete contacts
- Save button

Alert Screen (triggered state):

- "Sending Emergency Alert" message
- Location map preview
- List of contacts being alerted
- Cancel button (if accidental)

---

7. Data Flow Diagrams

7.1 Emergency Alert Flow

```
User Action (Button/Voice)
   ↓
Request Location Permission
   ↓
Get GPS Coordinates
   ↓
Retrieve Emergency Contacts from Storage
   ↓
Format Alert Message with Location
   ↓
Generate Deep Links (SMS/WhatsApp)
   ↓
Open Native Messaging Apps
   ↓
User Sends Messages
   ↓
Log Alert Event Locally
```

```
    ↓
Show Confirmation to User
```

7.2 Voice Command Flow

```
User Activates Voice Control
    ↓
Check Speech API Support
    ↓
Start Speech Recognition
    ↓
Show "Listening" Indicator
    ↓
Capture Audio Input
    ↓
Convert Speech to Text
    ↓
Match Text to Command Library
    ↓
Execute Matched Command
    ↓
Provide Visual/Audio Feedback
    ↓
Continue Listening or Stop
```

7.3 First Aid Guide Flow

```
User Opens First Aid Section
    ↓
Load First Aid Data from Local Source
    ↓
Display Emergency Categories
    ↓
User Selects Emergency Type
    ↓
Load Step-by-Step Guide
    ↓
Display Step 1
    ↓
User Navigates (Next/Previous)
    ↓
[Optional] Voice Reads Instructions
```

```
    ↓
User Completes or Exits Guide
```

---

8. Security & Privacy

8.1 Data Privacy

- All data stored locally on device only
- No server-side data transmission
- No user accounts or authentication required
- No tracking or analytics

8.2 Permissions

- **Location**: Required for emergency alerts
- **Microphone**: Required for voice commands
- **Notifications**: Optional for alerts

8.3 Data Security

- LocalStorage is accessible only to the app
- Consider encryption for sensitive medical data (future enhancement)
- No third-party data sharing

---

9. Browser Compatibility

9.1 Target Browsers

- Chrome/Edge 90+ (primary)
- Safari 14+ (iOS support)
- Firefox 88+

9.2 Feature Detection

- Check Speech API support
- Check Geolocation API support
- Graceful degradation if unsupported

9.3 Fallbacks

- No Speech API → Manual controls only
- No Geolocation → Allow manual location entry
- No Service Worker → Online mode only

---

10. Performance Requirements

- **Initial Load**: < 3 seconds on 3G
- **Offline Load**: < 1 second
- **Voice Recognition Latency**: < 500ms
- **Location Retrieval**: < 5 seconds
- **App Size**: < 5MB total

---

11. Testing Strategy

11.1 Unit Tests

- Voice command parsing
- Storage service functions
- Alert message formatting

11.2 Integration Tests

- Voice command to action execution
- Alert flow from trigger to message send
- Offline data retrieval

11.3 Manual Testing

- Test on actual mobile devices
- Test offline functionality (airplane mode)
- Test voice commands in quiet/noisy environments
- Test location accuracy
- Test across different browsers

11.4 User Testing

- Test with non-technical users
- Measure time to complete key tasks
- Gather feedback on UI clarity

---

12. Deployment

12.1 Hosting

- **Platform**: Netlify, Vercel, or GitHub Pages
- **HTTPS**: Required for PWA features
- **Custom Domain**: Optional

12.2 Build Process

```bash
npm run build
# Generates optimized production build
# Output: dist/ folder
```

12.3 PWA Requirements

- Served over HTTPS
- Includes valid manifest.json
- Registers service worker
- Has at least 192x192 and 512x512 icons

---

13. Future Enhancements (Out of Scope for MVP)

- Integration with emergency services (911/999/112)
- Medical ID QR code generation
- Bluetooth beacon for location tracking
- Multi-language support
- Wearable device integration
- Community emergency alerts
- Video-based first aid guides

- AI-powered symptom checker

---

14. Dependencies

14.1 Production Dependencies

```json
{
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-router-dom": "^6.20.0",
  "workbox-precaching": "^7.0.0",
  "workbox-routing": "^7.0.0",
  "workbox-strategies": "^7.0.0"
}
```

14.2 Development Dependencies

```json
{
  "vite": "^5.0.0",
  "@vitejs/plugin-react": "^4.2.0",
  "tailwindcss": "^3.4.0",
  "autoprefixer": "^10.4.16",
  "postcss": "^8.4.32"
}
```

---

15. API Reference

15.1 Storage Service API

```javascript
```

```
// Save user profile
storageService.saveUserProfile(profileData);

// Get user profile
const profile = storageService.getUserProfile();

// Save emergency contacts
storageService.saveContacts(contactsArray);

// Get emergency contacts
const contacts = storageService.getContacts();

// Clear all data
storageService.clearAll();
```

15.2 Voice Service API

```javascript
// Start listening for commands
voiceService.startListening();

// Stop listening
voiceService.stopListening();

// Speak text aloud
voiceService.speak("Follow these instructions");

// Check if supported
const supported = voiceService.isSupported();
```

15.3 Location Service API

javascript

```
// Get current location
locationService.getCurrentLocation()
  .then(coords => {
    // coords: { latitude, longitude, accuracy }
  });

// Get location as Google Maps link
const link = locationService.getMapLink(latitude, longitude);

// Check if supported
const supported = locationService.isSupported();
```

15.4 Alert Service API

```javascript
// Send emergency alert
alertService.sendAlert(contacts, location);

// Get alert history
const history = alertService.getHistory();

// Clear alert history
alertService.clearHistory();
```

---

16. Glossary

- **PWA**: Progressive Web App - web application that works offline and can be installed
- **Service Worker**: Background script that enables offline functionality
- **Web Speech API**: Browser API for speech recognition and synthesis
- **Geolocation API**: Browser API for accessing device location
- **LocalStorage**: Browser storage for persistent key-value data
- **Deep Link**: URL that opens a specific app with pre-filled data
- **Manifest**: JSON file that defines PWA properties
- **Cache-First**: Strategy that serves cached content before network
- **Network-First**: Strategy that tries network before falling back to cache

---

Document Version

- **Version**: 1.0
- **Date**: October 29, 2025
- **Status**: Draft for Implementation