Techwiz Documentation

Project Name: **ESavior**

Submitted By: **Aptech IIC**

Team Name: **PPC Developers**

S.No	*Names*	*Std Id*	*Batch*
01	*Kashif Raza*	*1360387*	*2202E*
02	*Hammad Abdul Rasheed*	*1362853*	*2202E*
03	*Abdul Samad*	*1357628*	*2022E*
04	*Muniba*	*1368057*	*2202E*

Index

I. Problem Definition:

In today's fast-paced world, the increasing reliance on mobile technology provides a unique opportunity to address critical emergencies in a timely and effective manner. However, many individuals still lack access to reliable and immediate help in life-threatening situations such as medical emergencies, accidents, natural disasters, and personal safety threats. Current emergency services are often fragmented, hard to access, or slow in response due to lack of real-time information and location tracking, which can exacerbate the severity of the crisis.

The primary problem faced by users is the inability to quickly connect to emergency services or notify loved ones during a crisis**. Delays in identifying the precise location of an emergency and communicating the exact nature of the situation to responders significantly hinder the chances of a successful intervention.

Additionally, there is a lack of integrated solutions that combine emergency services, real-time location sharing, automated notifications to contacts, and immediate assistance through mobile technology. This gap leaves individuals vulnerable in moments of urgency where every second counts.

The ESavior mobile app** aims to bridge this gap by providing users with a one-stop platform for emergency response services. Through features like real-time location tracking, direct communication with emergency responders, and instant alerts to predefined contacts, ESavior seeks to minimize response times and ensure that users can access help quickly and effectively.

Feel free to adjust the focus based on the specific features of your app.

II. Design Specification:

1.1 Purpose

The purpose of this document is to outline the design specifications for the *eSavior* mobile application. The app is aimed at providing users with an efficient platform for real-time monitoring and management of emergency situations, such as natural disasters, accidents, and health emergencies. The specification will cover functional and non-functional requirements, user interface design, system architecture, and data management.

1.2 Scope

This design specification focuses on the mobile app version of eSavior, targeting Android and iOS devices. It covers the user flow, feature set, performance requirements, and interaction design.

1.3 Audience

This document is intended for developers, designers, project managers, and stakeholders involved in the design and development of the eSavior mobile app.

2. System Overview

The ESavior app will:

- Allow users to report and track emergency situations.
- Provide real-time alerts based on location.
- Offer safety tips and emergency contact information.
- Enable integration with emergency response services.
- Support user roles such as general users, emergency responders, and administrators.

3. Functional Requirements

3.1 User Registration and Authentication

- Users can register via email, phone number, or social media accounts.
- Two-factor authentication for additional security.
- Password recovery mechanism via email or SMS.

3.2 Emergency Reporting

- **User input**: The app will allow users to report emergencies using a simple form, including the type of emergency, location, and additional details like photos or videos.
- **Location**: GPS functionality to automatically detect the user's location.
- **Real-time communication**: In-app messaging system to communicate with emergency services.

3.3 Alert Notifications

- Push notifications for emergency alerts, tips, and updates.
- Location-based notifications for region-specific emergencies (geo-fencing).
- Customizable notification settings (sound, vibration, or silent).

3.4 Map Integration

- Real-time map to display emergency zones, safe areas, and emergency service locations.
- Navigation assistance to nearby safe locations.

3.5 Emergency Information Hub

- Detailed safety tips categorized by emergency type (fire, flood, earthquake, health, etc.).
- Emergency contact information for local services like hospitals, police stations, and fire departments.

3.6 User Roles and Permissions

- **General Users**: Can report incidents, receive alerts, and access safety tips.
- **Emergency Responders**: Can manage incidents, coordinate response, and provide feedback.
- **Administrators**: Oversee operations, manage user roles, and handle app configurations.

4. Non-Functional Requirements

4.1 Performance

- The app must respond within 2 seconds for most actions.
- Real-time data synchronization across users should not exceed 5 seconds.

4.2 Scalability

- The system must handle up to 500,000 concurrent users during high-demand situations (e.g., during natural disasters).

4.3 Security

- End-to-end encryption for all communications between users and emergency services.
- Data protection compliant with GDPR and local laws.

4.4 Availability

- The app should have 99.99% uptime with failover systems in place to maintain operations during server outages.

4.5 Compatibility

- The app must be compatible with the latest versions of Android and iOS, with backward compatibility for versions up to two years old.

5. User Interface Design

5.1 Overview

The ESavior app interface will be designed for intuitive use, prioritizing ease of access to emergency information and reporting functionality.

5.2 Navigation Flow

- **Home Screen**: Quick access to report emergencies, view alerts, and emergency map.
- **Incident Reporting**: Step-by-step form for users to submit emergency reports.
- **Alerts Section**: Display real-time emergency alerts relevant to the user's location.
- **Safety Tips**: Categorized safety guidelines for various emergency situations.

5.3 UI Elements

- **Buttons**: Large, clear buttons for primary actions like "Report Emergency" and "Send Alert."
- **Icons**: Icons for each type of emergency to ensure quick identification (e.g., fire, flood, medical).
- **Map**: Interactive map with color-coded markers for emergencies and safe zones.

5.4 Colors and Themes

- Red and orange tones for emergency alerts.
- Blue and green tones for safe zones and confirmations.

6. System Architecture

6.1 Frontend

The mobile app will be developed using **React Native**, ensuring a cross-platform experience for Android and iOS users. Core components include:

- **Navigation**: React Native Navigation for smooth transitions between screens.
- **State Management**: Redux for managing app-wide state, such as user data and alerts.
- **Real-time Communication**: WebSocket or Firebase for real-time updates.

6.2 Backend

The backend system will be hosted on cloud infrastructure (e.g., AWS or Google Cloud) with the following components:

- **REST API**: To handle requests from the app.
- **Database**: A NoSQL database (e.g., Firebase Firestore) for fast data access.
- **Notification Service**: Firebase Cloud Messaging (FCM) for push notifications.

6.3 Data Flow

1. **User actions**: Users interact with the mobile app, submitting reports and receiving alerts.
2. **API requests**: The app communicates with the backend via RESTful APIs for real-time operations.
3. **Data storage**: Incident data and user profiles are stored securely in the database.
4. **Notifications**: The system triggers notifications based on location and event priority.

7. Data Management

7.1 Data Types

- **User Data**: User profiles, preferences, location, and interaction history.
- **Incident Reports**: Emergency type, time, location, media (photos/videos), and description.
- **Alerts**: Real-time updates and notifications, categorized by region and emergency type.

7.2 Data Security

- All sensitive data will be encrypted in transit (HTTPS) and at rest.
- User data access will be limited by role-based access control (RBAC).

8. Testing and Quality Assurance

8.1 Unit Testing

- Each module will undergo unit testing to ensure individual components work as expected.

8.2 Integration Testing

- Full integration testing will be carried out to ensure seamless interaction between app components and backend services.

8.3 User Acceptance Testing (UAT)

- Testing with a small group of real users to ensure the app is intuitive and meets their needs.

8.4 Performance Testing

- Load testing will be conducted to ensure the app performs well under high user demand.

9. Deployment and Maintenance

9.1 Deployment Strategy

- Initial deployment will be carried out in phases, with a beta release followed by full production rollout on the Google Play Store and Apple App Store.

9.2 Ongoing Maintenance

- Regular updates to ensure compatibility with new OS versions.
- 24/7 monitoring of system uptime and real-time issue reporting.

III. Diagrams:

a)Data Flow Diagram:

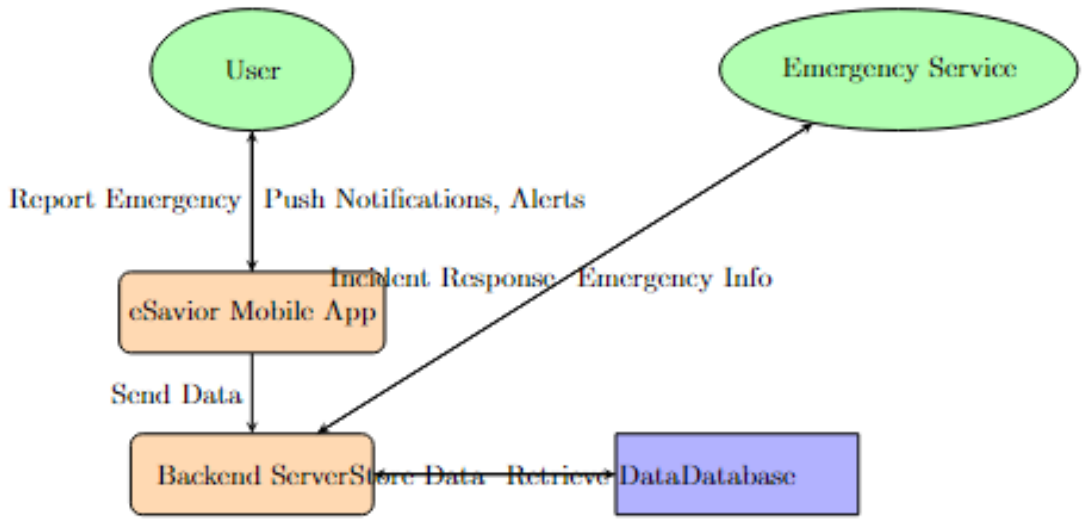

Figure 1: Data Flow Diagram for eSavior Mobile Application

b)Flow Chart: