COS209 - Final Project Documentation

Working Title : SheSafe
– Empowering Personal Safety/Emergency Assist Through Technology

Student Name : Chaw Thiri Win

Table of Contents

Introduction

The SheSafe app is a complete mobile safety tool that provides users with location sharing, customized safety features, and quick emergency response, and it is specifically designed for vulnerable populations. The project's objective is to develop a simple, reliable Android application with an intuitive interface that includes necessary functions including contact management, location sharing in real time, panic mode with voice recording, and emergency messaging. The program leverages Firebase as a backend server for authentication and data synchronization, in addition to local SQLite databases for offline functionality. The primary motivation behind SheSafe is the provision of readily available safety tools that assist users in emergency situations, particularly in crisis-prone locations. Comprehensive system architecture, database normalization, safe authentication procedures, and extensive testing to guarantee usability and robustness are all included in this project.

Section A

1. Project Overview

Company Background & Project Initiation

Owls Studio is a virtual technology startup focus on creating innovative solution to address real world safety concerns. The team's flagship, application, SheSafe is a mobile solution designed to support personal safety-especially for women and venerable individuals. This app provide user with emergency features such as SOS/alert, location sharing, live location sharing, and panic mode. This mobile application development have used to date tool and methodologies.

Main Objectives

- Determine and evaluate system requirements and safety needs.
- Create UI/UX and app architecture that is focused on the user.
- Create essential application functionalities with Firebase, SQLite, and Java.
- Perform comprehensive testing and validation.
- Record the system and assess the project's overall results.

Methodology Used

A hybrid development approach combining Waterfall and Agile was used:

- Waterfall: was applied during the initial planning, requirement gathering, and system designing phases.
- Agile: was used during development, testing, and evaluation, rapid prototyping, and flexible response to changes.

Change Management

Every modification to the scope or requirements of a project is managed through a formal change request procedure. Requests are examined and accepted in accordance with their viability and impact.

Software Tools Used

- **Front-end**: Java (Android SDK via Android Studio)
- **Back-end**: Firebase (Realtime DB, Authentication, Notifications), SQLite
- **UI/UX Design**: Figma, Draw.io
- **Testing**: Android Emulator, JUnit, real Android devices
- **Diagram Design**: StarUML, Draw.io

2. Requirements Definition

Problem Statement
Many people, especially women, are at risk for personal safety in both public and private settings. The practical usefulness of current mobile safety applications is limited by their frequent inability to offer discrete safety features, efficient offline functionality, or real-time emergency support.

New System Functions (Informal Overview)
By offering the following essential elements, the SheSafe app seeks to close these gaps:

- One-tap and shake-triggered SOS alerts
- Fake call simulation to covertly get out of awkward or hazardous circumstances.
- Contact management and incident logs
- PIN-based privacy protection
- Offline access using local database storage
- Real-time location sharing with emergency contacts

Hardware Requirements

- Android smartphone
- Functional GPS and accelerometer sensors
- Internet access (for cloud features like real-time sync)

Feasibility Summary

- **Technical**: All planned features are achievable using Firebase and SQLite on Android.
- **Economic**: Low development cost using open-source platforms and free tools.
- **Social**: High relevance and value to public safety, particularly for women and girls.

3. Scope of Work (Work Breakdown Structure) – SheSafe App

1. Project Initiation

1.1. Requirement Analysis

1.1.1. Conduct user research and surveys
1.1.2. Identify user types (normal, premium, admin)
1.1.3. Define essential and advanced safety features

1.2. Project Planning

1.2.1. Define project scope and objectives
1.2.2. Identify stakeholders and allocate team roles
1.2.3. Develop project schedule and milestones
1.2.4. Establish project timeline and budget

2. Design Phase

2.1. UI/UX Design

2.1.1. Design splash, permission, and welcome screens
2.1.2. Design authentication screens (login, admin login, register)
2.1.3. Design home screen and navigation bar
2.1.4. Design SOS and alert screens (share location, fake siren, incident log, shake for SOS)
2.1.5. Design location sharing and contact management screens
2.1.6. Design incident CRUD (create, read, update, delete) screens
2.1.7. Design emergency contact CRUD screens
2.1.8. Design alert/panic mode (auto location sharing, auto recording, add message)
2.1.9. Design navigation menu (profile, pin lock, safety tips, policies, about, contact)
2.1.10. Design admin panel (user management: normal/premium)

2.2. Prototype & User Feedback

2.2.1. Develop interactive prototypes
2.2.2. Conduct usability testing and collect feedback
2.2.3. Refine UI/UX based on feedback

3. Development Phase

3.1. Frontend Development

3.1.1. Implement splash, permission, welcome, and onboarding flows
3.1.2. Develop authentication (login, admin login, register)
3.1.3. Build home screen with main features (share location, fake call, incident log, contacts)
3.1.4. Implement navbar navigation (home, contact, SOS, alert, settings)
3.1.5. Develop SOS features (share location, fake siren, SOS button, shake for SOS)
3.1.6. Implement share location with contact selection
3.1.7. Develop incident log CRUD
3.1.8. Implement emergency contact CRUD
3.1.9. Build alert/panic mode (auto location, auto recording, add message)
3.1.10. Develop navigation menu screens (profile, pin lock, safety tips, privacy, terms, about, contact)
3.1.11. Implement admin features (user role management)

3.2. Backend Development

3.2.1. Set up Firebase authentication and user roles
3.2.2. Implement real-time database for incidents, contacts, and logs
3.2.3. Set up cloud messaging for alerts and notifications
3.2.4. Integrate SQLite for offline data
3.2.5. Configure admin backend for user management

3.3. Integration

3.3.1. Connect frontend and backend
3.3.2. Integrate third-party APIs (e.g., OSMDroid for maps, Firebase for auth/location)
3.3.3. Ensure data synchronization (offline/online)

4. Testing Phase

4.1. Functional Testing

4.1.1. Test each screen and feature on emulator and real devices
4.1.2. Conduct unit and integration tests (JUnit)

4.2. User Acceptance Testing

4.2.1. Recruit target users for testing
4.2.2. Collect and analyze feedback
4.2.3. Refine app based on test results

4.3. Security & Privacy Testing

4.3.1. Test PIN lock, privacy policy, and data encryption
4.3.2. Verify compliance with data protection standards

5. Deployment & Training

5.1. App Deployment

5.1.1. Prepare app for Google Play release
5.1.2. Deploy backend services

5.2. User Training & Documentation

5.2.1. Prepare user guides and FAQs
5.2.2. Provide training for admin users

6. Evaluation & Project Closure

6.1. Project Evaluation

6.1.1. Review objectives and outcomes
6.1.2. Document lessons learned and recommendations

6.2. Handover & Support

6.2.1. Handover documentation and source code
6.2.2. Set up support channels

6.3. Project Closure

6.3.1. Final project review and sign-off
6.3.2. Archive files and close project

4. Gantt Chart

ID	Task Mode	Task Name	Duration	Start	Finish
1		**TOTAL**	**43 days**	**Wed 6/11/25**	**Fri 8/8/25**
2		**1. Project Initiation**	**4 days**	**Wed 6/11/25**	**Mon 6/16/25**
3		**1.1. Requirement Analysis**	**2 days**	**Wed 6/11/25**	**Thu 6/12/25**
4		1.1.1. Conduct user research and surveys	1 day	Wed 6/11/25	Wed 6/11/25
5		1.1.2. Identify user types (normal,	0.5 days	Thu 6/12/25	Thu 6/12/25
6		1.1.3. Define essential and	0.5 days	Thu 6/12/25	Thu 6/12/25
7		**1.2. Project Planning**	**2 days**	**Fri 6/13/25**	**Mon 6/16/25**
8		1.2.1. Define project scope and objectives	0.5 days	Fri 6/13/25	Fri 6/13/25
9		1.2.2. Identify stakeholders and	0.5 days	Fri 6/13/25	Fri 6/13/25
10		1.2.3. Develop project schedule and	0.5 days	Mon 6/16/25	Mon 6/16/25
11		1.2.4. Establish project timeline and	0.5 days	Mon 6/16/25	Mon 6/16/25
12		**2. Design Phase**	**4 days**	**Tue 6/1**	**Fri 6/20**
13		**2.1. UI/UX Design**	**2 days**	**Tue 6/1**	**Wed 6/1**
14		2.1.1. Design splash, permission, and welcome screens	0.2 days	Tue 6/17/25	Tue 6/17/25
15		2.1.2. Design authentication screens (login,	0.2 days	Tue 6/17/25	Tue 6/17/25
16		2.1.3. Design home screen and	0.2 days	Tue 6/17/25	Tue 6/17/25
17		2.1.4. Design SOS and alert screens (share location, fake siren, incident log,	0.2 days	Tue 6/17/25	Tue 6/17/25
18		2.1.5. Design location sharing and contact management	0.2 days	Tue 6/17/25	Tue 6/17/25
19		2.1.6. Design incident CRUD (create, read,	0.2 days	Wed 6/18/25	Wed 6/18/25
20		2.1.7. Design emergency contact	0.2 days	Wed 6/18/25	Wed 6/18/25
21		2.1.8. Design alert/panic mode (auto location sharing, auto	0.2 days	Wed 6/18/25	Wed 6/18/25
22		2.1.9. Design navigation menu (profile, pin lock, safety tips, policies,	0.2 days	Wed 6/18/25	Wed 6/18/25
23		2.1.10. Design admin panel (user management:	0.2 days	Wed 6/18/25	Wed 6/18/25
24		**2.2. Prototype & User Feedback**	**2 days**	**Thu 6/19/25**	**Fri 6/20/25**
25		2.2.1. Develop interactive	1 day	Thu 6/19/25	Thu 6/19/25
26		2.2.2. Conduct usability testing and	0.5 days	Fri 6/20/25	Fri 6/20/25
27		2.2.3. Refine UI/UX based on feedback	0.5 days	Fri 6/20/25	Fri 6/20/25

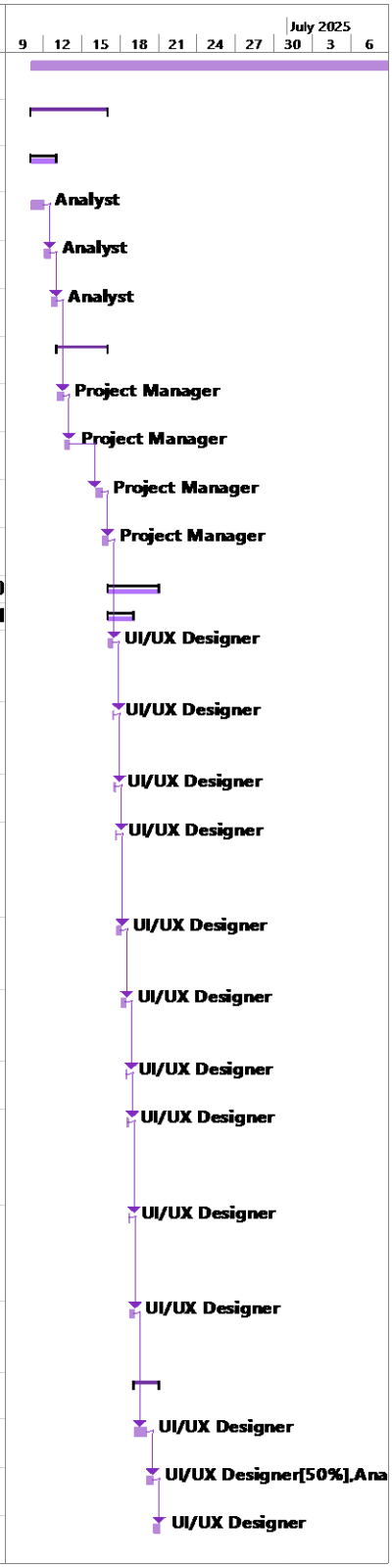

Figure A.4.1: Task Overview & Gantt Chart of SheSafe Project

ID	Task Mode	Task Name	Duration	Start	Finish
28		**3. Development Phase**	**22 days**	**Mon 6/23/25**	**Tue 7/22/25**
29		**3.1. Frontend Development**	**11 days**	**Mon 6/23/25**	**Mon 7/7/25**
30		3.1.1. Implement splash,	0.5 days	Mon 6/23/25	Mon 6/23/25
31		3.1.2. Develop authentication	0.5 days	Mon 6/23/25	Mon 6/23/25
32		3.1.3. Build home screen with main	1 day	Tue 6/24/25	Tue 6/24/25
33		3.1.4. Implement navbar navigation	1 day	Wed 6/25/25	Wed 6/25/25
34		3.1.5. Develop SOS features	2 days	Thu 6/26/25	Fri 6/27/25
35		3.1.6. Implement share location	1 day	Mon 6/30/25	Mon 6/30/25
36		3.1.7. Develop incident log	1 day	Tue 7/1/25	Tue 7/1/25
37		3.1.8. Implement emergency	1 day	Wed 7/2/25	Wed 7/2/25
38		3.1.9. Build alert/panic mode	1 day	Thu 7/3/25	Thu 7/3/25
39		3.1.10. Develop navigation menu	1 day	Fri 7/4/25	Fri 7/4/25
40		3.1.11. Implement admin	1 day	Mon 7/7/25	Mon 7/7/25
41		**3.2. Backend Development**	**7 days**	**Tue 7/8/25**	**Wed 7/16/25**
42		3.2.1. Set up Firebase	1 day	Tue 7/8/25	Tue 7/8/25
43		3.2.2. Implement real-time	2 days	Wed 7/9/25	Thu 7/10/25
44		3.2.3. Set up cloud messaging	1 day	Fri 7/11/25	Fri 7/11/25
45		3.2.4. Integrate SQLite for offline	2 days	Mon 7/14/25	Tue 7/15/25
46		3.2.5. Configure admin backend	1 day	Wed 7/16/25	Wed 7/16/25
47		**3.3. Integration**	**4 days**	**Thu 7/17/25**	**Tue 7/22/25**
48		3.3.1. Connect frontend and	1 day	Thu 7/17/25	Thu 7/17/25
49		3.3.2. Integrate third-party APIs	2 days	Fri 7/18/25	Mon 7/21/25
50		3.3.3. Ensure data synchronization	1 day	Tue 7/22/25	Tue 7/22/25

Figure A.4.2: Task Overview & Gantt Chart of SheSafe Project

ID	Task Mode	Task Name	Duration	Start	Finish
51		**4. Testing Phase**	**6 days**	**Wed 7/23/25**	**Wed 7/30/25**
52		**4.1. Functional Testing**	**2 days**	**Wed 7/23/25**	**Thu 7/24/25**
53		4.1.1. Test each screen and feature on emulator and real	1 day	Wed 7/23/25	Wed 7/23/25
54		4.1.2. Conduct unit and integration tests	1 day	Thu 7/24/25	Thu 7/24/25
55		**4.2. User Acceptance Testing**	**2 days**	**Fri 7/25/25**	**Mon 7/28/25**
56		4.2.1. Recruit target users for testing	1 day	Fri 7/25/25	Fri 7/25/25
57		4.2.2. Collect and analyze feedback	0.5 days	Mon 7/28/25	Mon 7/28/25
58		4.2.3. Refine app based on test results	0.5 days	Mon 7/28/25	Mon 7/28/25
59		**4.3. Security & Privacy Testing**	**2 days**	**Tue 7/29/25**	**Wed 7/30/25**
60		4.3.1. Test PIN lock, privacy policy, and data encryption	1 day	Tue 7/29/25	Tue 7/29/25
61		4.3.2. Verify compliance with data protection	1 day	Wed 7/30/25	Wed 7/30/25
62		**5. Deployment & Training**	**3 days**	**Thu 7/31/25**	**Mon 8/4/25**
63		**5.1. App Deployment**	**1 day**	**Thu 7/31/25**	**Thu 7/31/25**
64		5.1.1. Prepare app for Google Play	0.5 days	Thu 7/31/25	Thu 7/31/25
65		5.1.2. Deploy backend services	0.5 days	Thu 7/31/25	Thu 7/31/25
66		**5.2. User Training & Documentation**	**2 days**	**Fri 8/1/25**	**Mon 8/4/25**
67		5.2.1. Prepare user guides and FAQs	1 day	Fri 8/1/25	Fri 8/1/25
68		5.2.2. Provide training for admin	1 day	Mon 8/4/25	Mon 8/4/25
69		**6. Evaluation & Project Closure**	**4 days?**	**Tue 8/5/25**	**Fri 8/8/25**
70		**6.1. Project Evaluatio**	**1 day**	**Tue 8/5/2**	**Tue 8/5/2**
71		6.1.1. Review objectives and	0.5 days	Tue 8/5/25	Tue 8/5/25
72		6.1.2. Document lessons learned and recommendations	0.5 days	Tue 8/5/25	Tue 8/5/25
73		**6.2. Handover & Support**	**2 days**	**Wed 8/6/25**	**Thu 8/7/25**
74		6.2.1. Handover documentation and source code	1 day	Wed 8/6/25	Wed 8/6/25
75		6.2.2. Set up support channels	1 day	Thu 8/7/25	Thu 8/7/25
76		**6.3. Project Closure**	**1 day**	**Fri 8/8/25**	**Fri 8/8/25**
77		6.3.1. Final project review and sign-off	0.5 days	Fri 8/8/25	Fri 8/8/25
78		6.3.2. Archive files and close project	0.5 days	Fri 8/8/25	Fri 8/8/25
79					
80		7. End of Project	1 day?	Mon 8/11/25	Mon 8/11/25

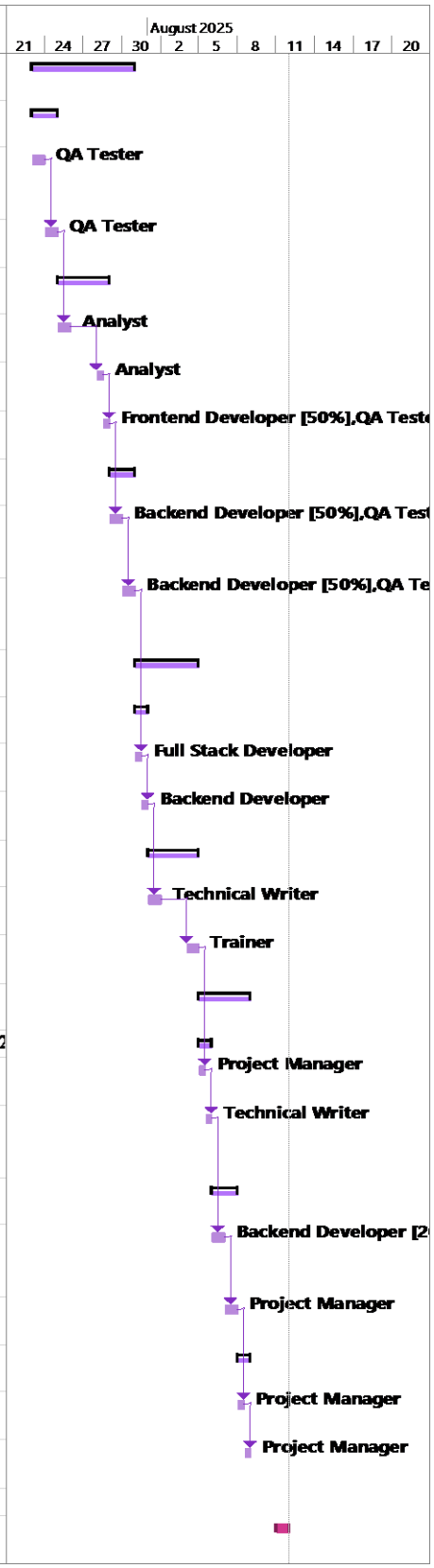

Figure A.4.3: Task Overview & Gantt Chart of SheSafe Project

5. Budgeting

Deliverable	Schedule (DAYS)	Person(s)	Rate (USD)	Budget (USD)
Requirement Gathering (Surveys, Research)	4	2 Analysts	$15/hr × 4 hrs/day	$480
UI/UX Design (Figma/Draw.io)	4	1 Designer	$15/hr × 4 hrs/day	$240
App Architecture & Planning	5	1 Project Manager, 1 Full Stack Developer	$20/hr × 4 hrs/day	$800
Frontend Development (Java/Android Studio)	10	2 Frontend Developers	$20/hr × 6 hrs/day	$2,400
Backend Development (Firebase, SQLite)	8	2 Backend Developers	$20/hr × 6 hrs/day	$1,920
Testing (Testers)	4	2 Testers	$10/hr × 4 hrs/day	$320
User Training (Preparation & Sessions)	3	2 Trainer	$10/hr × 4 hrs/day	$240
Documentation & Manuals	4	1 Writer	$10/hr × 4 hrs/day	$160
Testing Devices (Android phone purchase)	2	-	$200 (one-time)	$200
Cloud Services (Firebase, 2 months)	0	-	$25/month	$50
Miscellaneous (Internet, Power, 6 months)	-	-	$10/month	$60
Total Project Budget	**44**	-	-	**$6,870**

Table : Budgeting Table

Hardware Costs

No.	Item	Quantity/Duration	Unit Cost (USD)	Total Cost (USD)	Notes
1	Android Phone (Testing Device)	1 (one-time)	$200	$200	Needed for app testing
2	Miscellaneous (Internet, Power)	6 months	$10/month	$60	Basic connectivity & utilities

Total Hardware Cost: $260

Software Costs

No.	Item	Duration	Unit Cost (USD)	Total Cost (USD)	Notes
1	Cloud Services (Firebase)	2 months	$25/month	$50	Backend cloud usage fees

Total Software Cost: $50

Human Resource Costs

No.	Deliverable	Schedule (Days)	Personnel	Hourly Rate (USD)	Hours per Day	Total Cost (USD)
1	Requirement Gathering (Surveys, Research)	4	2 Analysts	$15	4	2 × $15 × 4 × 4 = $480
2	UI/UX Design (Figma/Draw.io)	4	1 Designer	$15	4	1 × $15 × 4 × 4 = $240
3	App Architecture & Planning	5	1 Project Manager, 1 Full Stack Dev	$20	4	2 × $20 × 4 × 5 = $800
4	Frontend Development (Java/Android Studio)	10	2 Frontend Developers	$20	6	2 × $20 × 6 × 10 = $2,400
5	Backend Development (Firebase, SQLite)	8	2 Backend Developers	$20	6	2 × $20 × 6 × 8 = $1,920
6	Testing (Testers)	4	2 Testers	$10	4	2 × $10 × 4 × 4 = $320
7	User Training (Preparation & Sessions)	3	2 Trainers	$10	4	2 × $10 × 4 × 3 = $240

Total Human Resource Cost: $6,560

Total Summary Cost

Category	Total Cost (USD)
Hardware Cost	$260
Software Cost	$50
Human Resource Cost	$6,560
Grand Total	$6,870

Section B

1. System Requirement Specifications

1.1 Functional Requirements

- **User Authentication**
  Users can securely register, log in, and reset their passwords using Firebase Authentication. Email verification is enforced for added security.
- **Emergency Messaging**
  Users can send SOS and Alert messages via SMS to pre-selected emergency contacts. Triggers include one-tap buttons or shake gestures.
- **Contact Management**
  Users can add, edit, or delete emergency contacts. Contacts can be marked as favorites for quicker access during emergencies.
- **Live Location Sharing**
  Users can share their real-time GPS location with trusted contacts and view others' shared locations on an interactive OSMDroid-based map.
- **Fake Calls and Sounds**
  The app can simulate incoming fake calls and siren sounds to help users safely get out of dangerous or uncomfortable situations.
- **Panic Mode**
  In emergency scenarios, a countdown timer initiates audio recording and automatically sends an SOS message along with the user's live location.
- **App Security**
  Includes PIN-based locking for app access and secure account deactivation with password confirmation.
- **Incident Logging**
  Locally stores logs of all emergency messages, including timestamps, contact details, and location info for later review.
- **Onboarding & Permissions**
  A guided onboarding process educates users and handles runtime permissions for SMS, Contacts, Location, and Microphone access.

1.2 Non-Functional Requirements

- **Usability**
  Features an intuitive user interface with tabbed navigation and confirmation prompts for sensitive actions.
- **Performance**
  Ensures real-time responsiveness, particularly for location updates and emergency messaging.
- **Reliability**
  Handles permission denials gracefully with fallback behaviors to prevent app crashes or user lockout.
- **Security**
  Implements password hashing, secure authentication, and encrypted communication (HTTPS). Sensitive user data is stored securely.
- **Portability**
  Compatible with Android devices that support GPS, accelerometer, SMS, and runtime permissions.

- **Maintainability**
  Follows a modular and scalable architecture for easy maintenance and future feature expansion.
- **Offline Functionality**
  Supports core features like emergency messaging and contact access without internet using local storage (SQLite).

1.3 Hardware Requirements

- An Android smartphone with:
  - GPS module
  - Accelerometer
  - Microphone
  - SMS capability
  - Internet connectivity

1.4 Software Requirements

- Android OS (API level 23 or higher)
- Firebase Authentication and Firestore for cloud-based user and location management
- SQLite for local data storage
- OSMDroid for map and location visualization
- Android SDK libraries (Location, Media, UI, etc.)

1.5 Constraints and Dependencies

- App functionality relies on runtime permissions for SMS, Contacts, Location, and Microphone access.
- Accurate GPS is necessary for reliable location sharing.
- SMS functionality depends on carrier/network support and device compatibility.
- Internet is required for cloud-based features like authentication and live location sharing.
- Firebase service availability and usage limits may impact cloud functionality.

Use Case Diagram

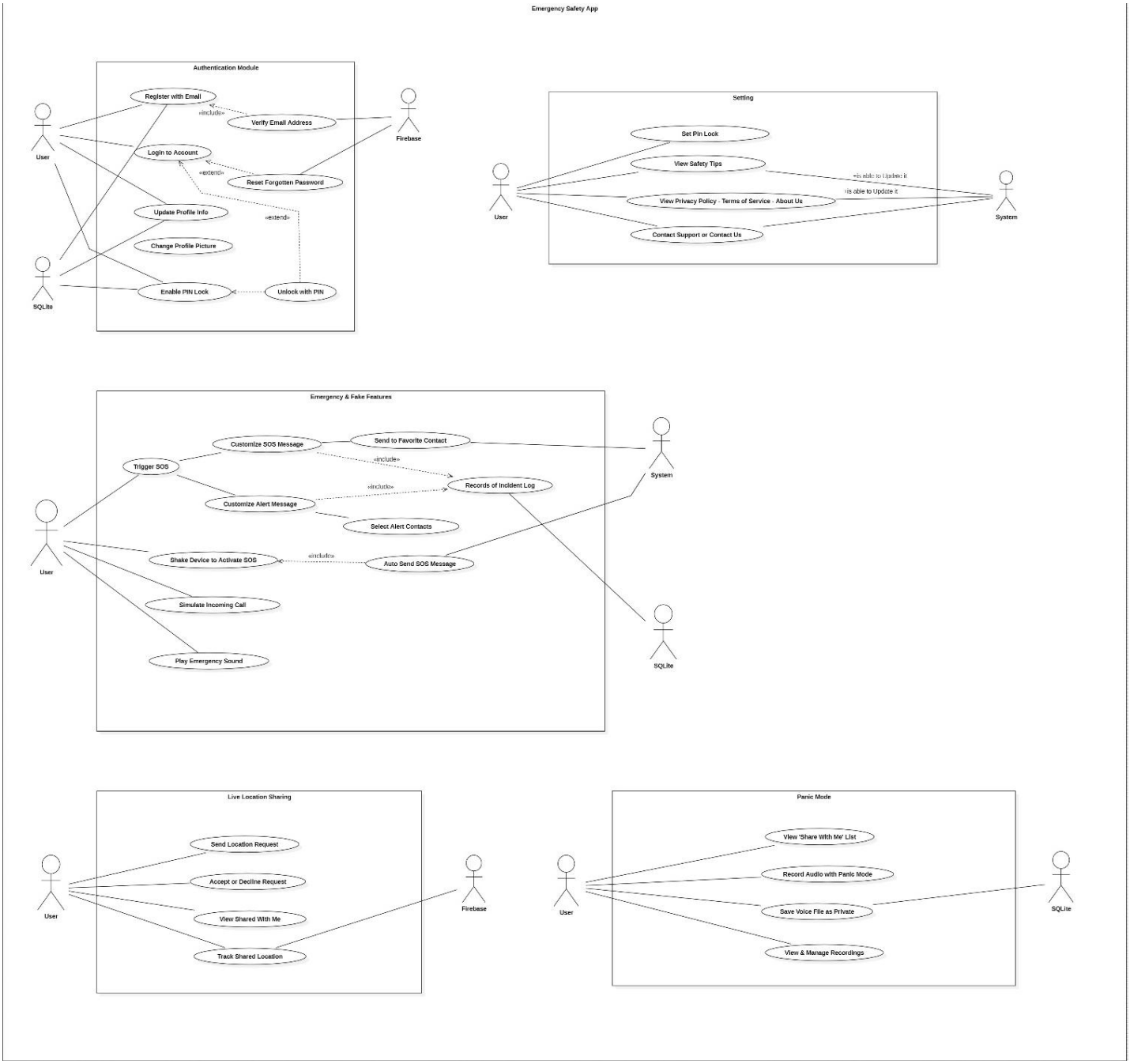

Figure B.1.1: Use Case Diagram

Class Diagram

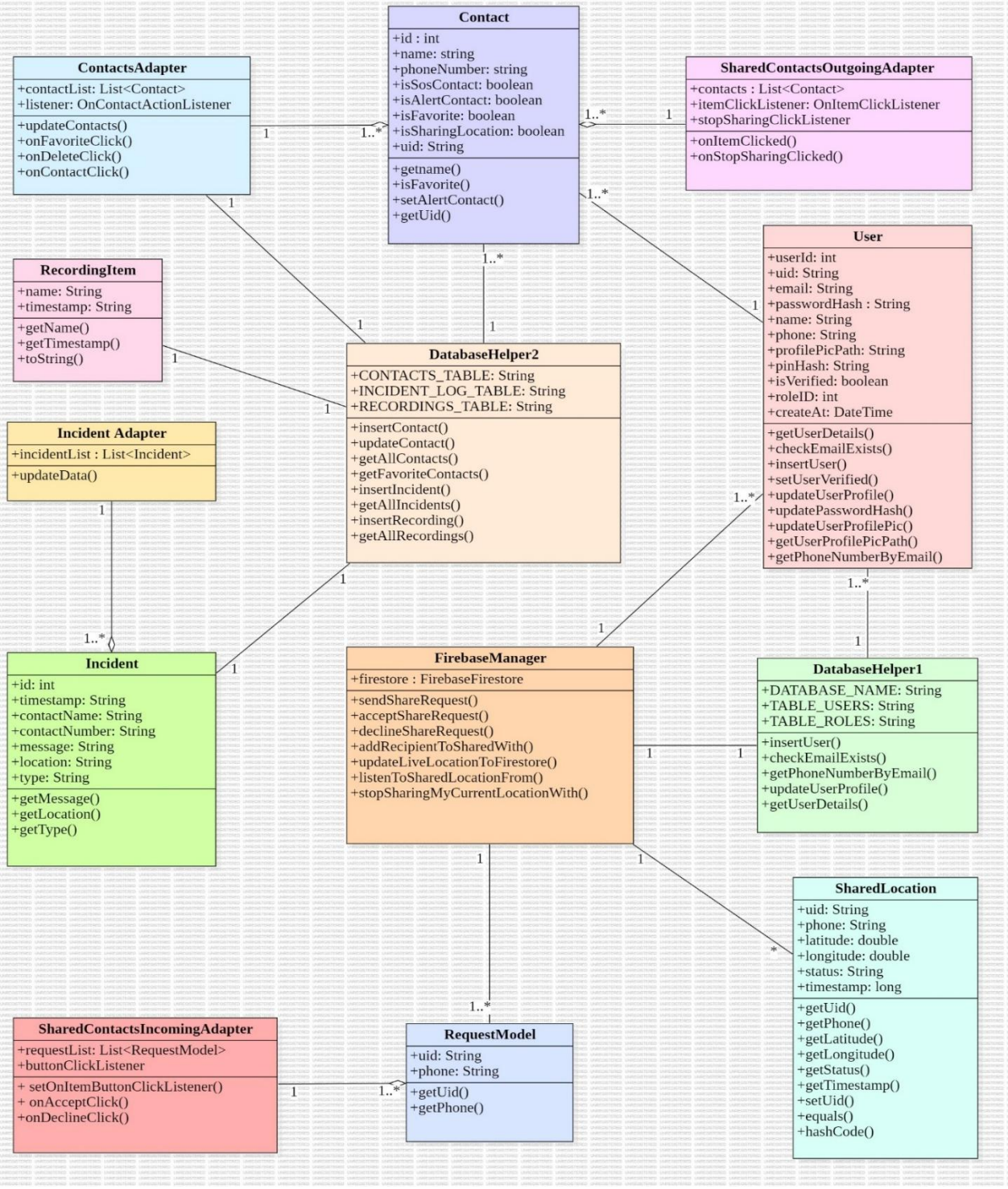

Figure B.1.2: Class Diagram

Sequence Diagram

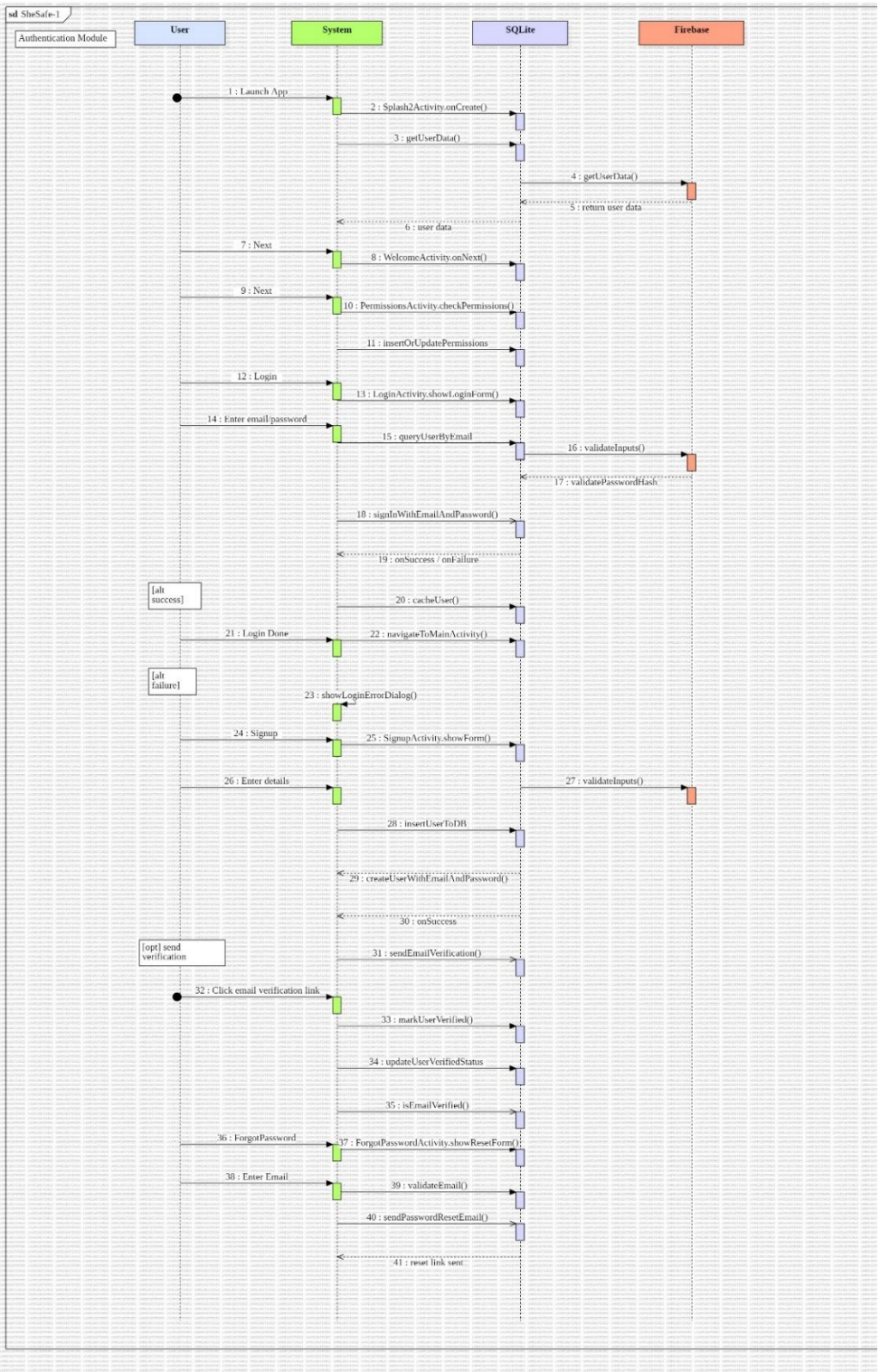

Figure B.1.3.1: Sequence Diagram

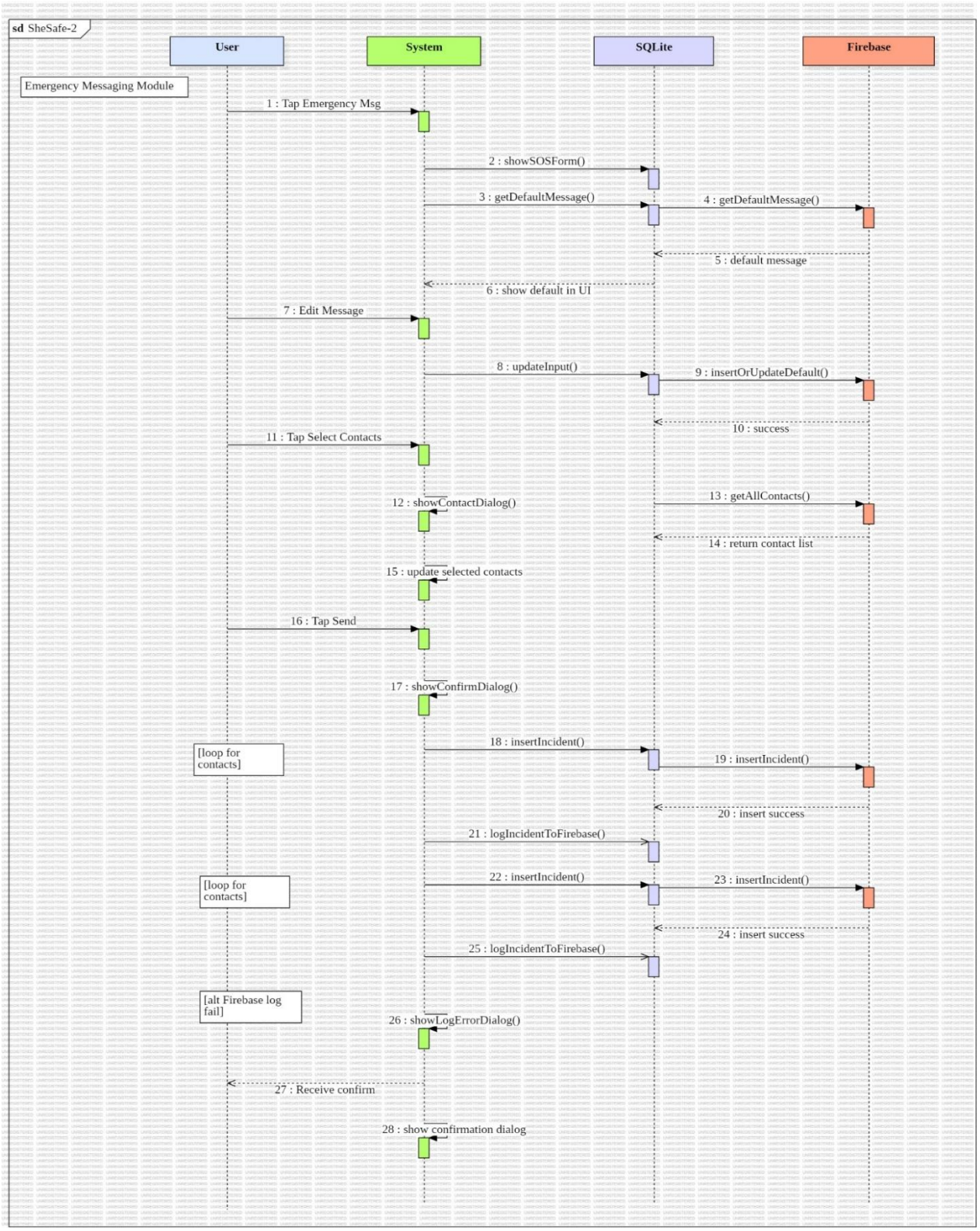

Figure B.1.3.2: Sequence Diagram

State Transition Diagram

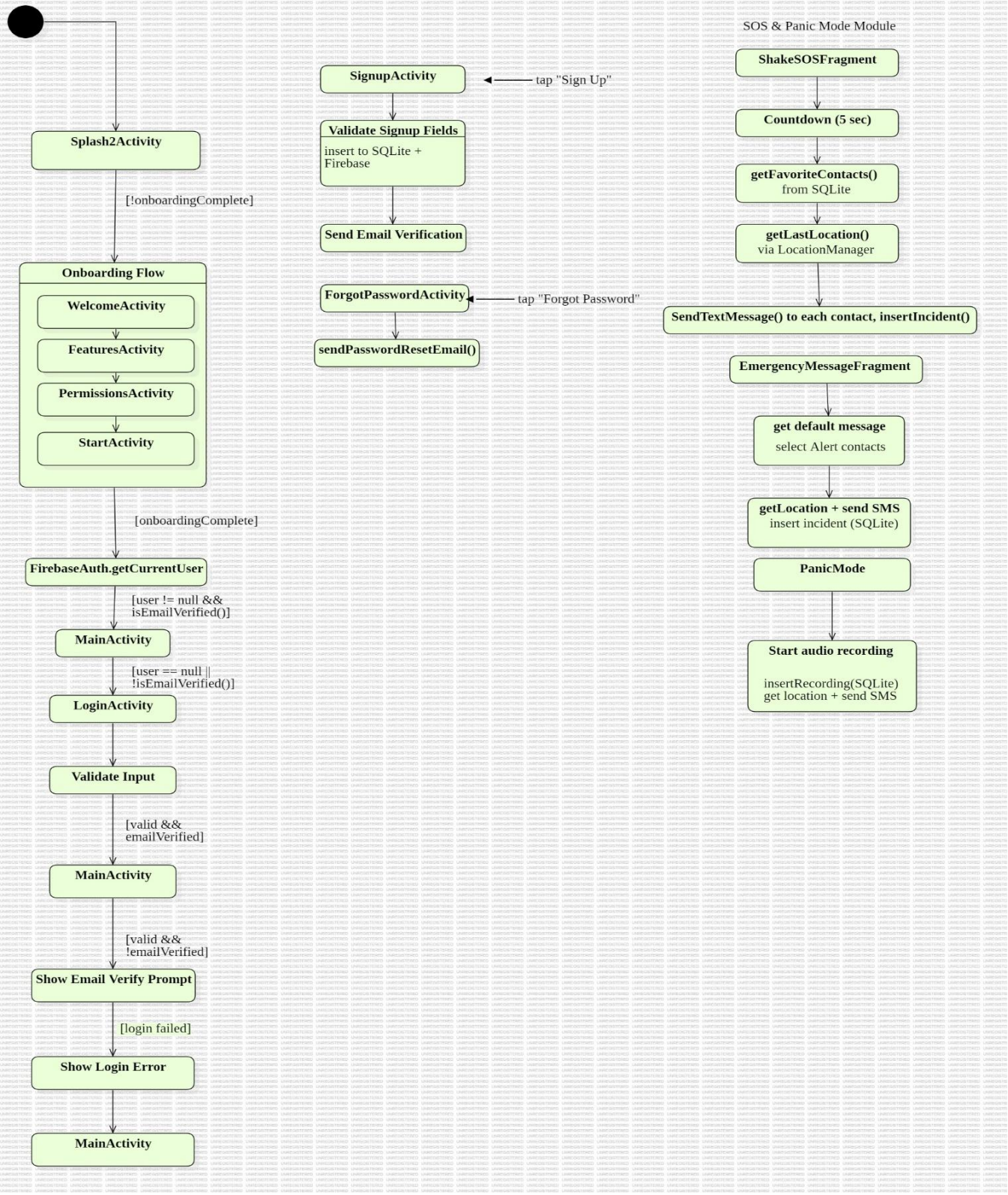

Figure B.1.4.1: State Transition Diagram

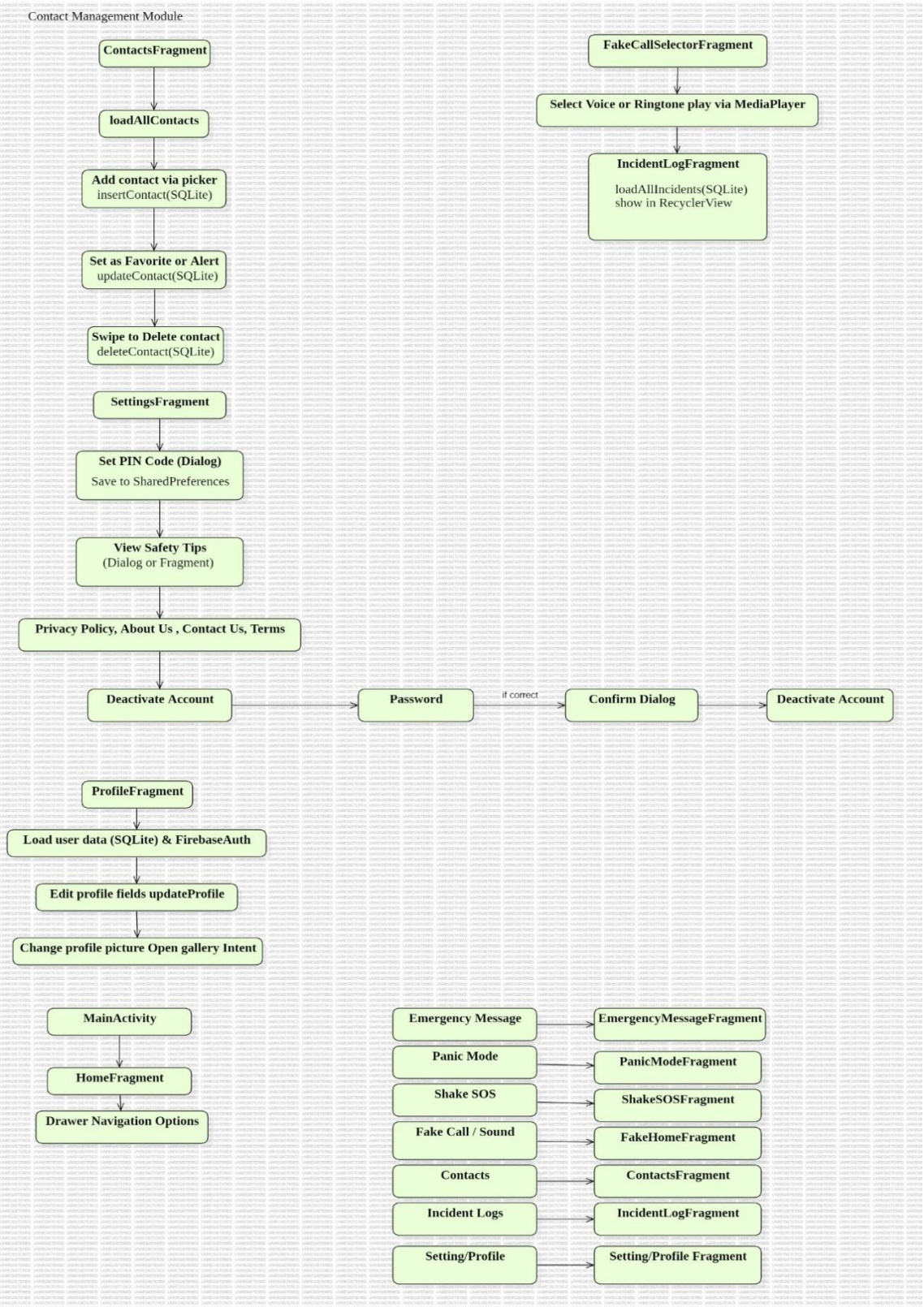

Figure B.1.4.2: State Transition Diagram

Activity Diagram

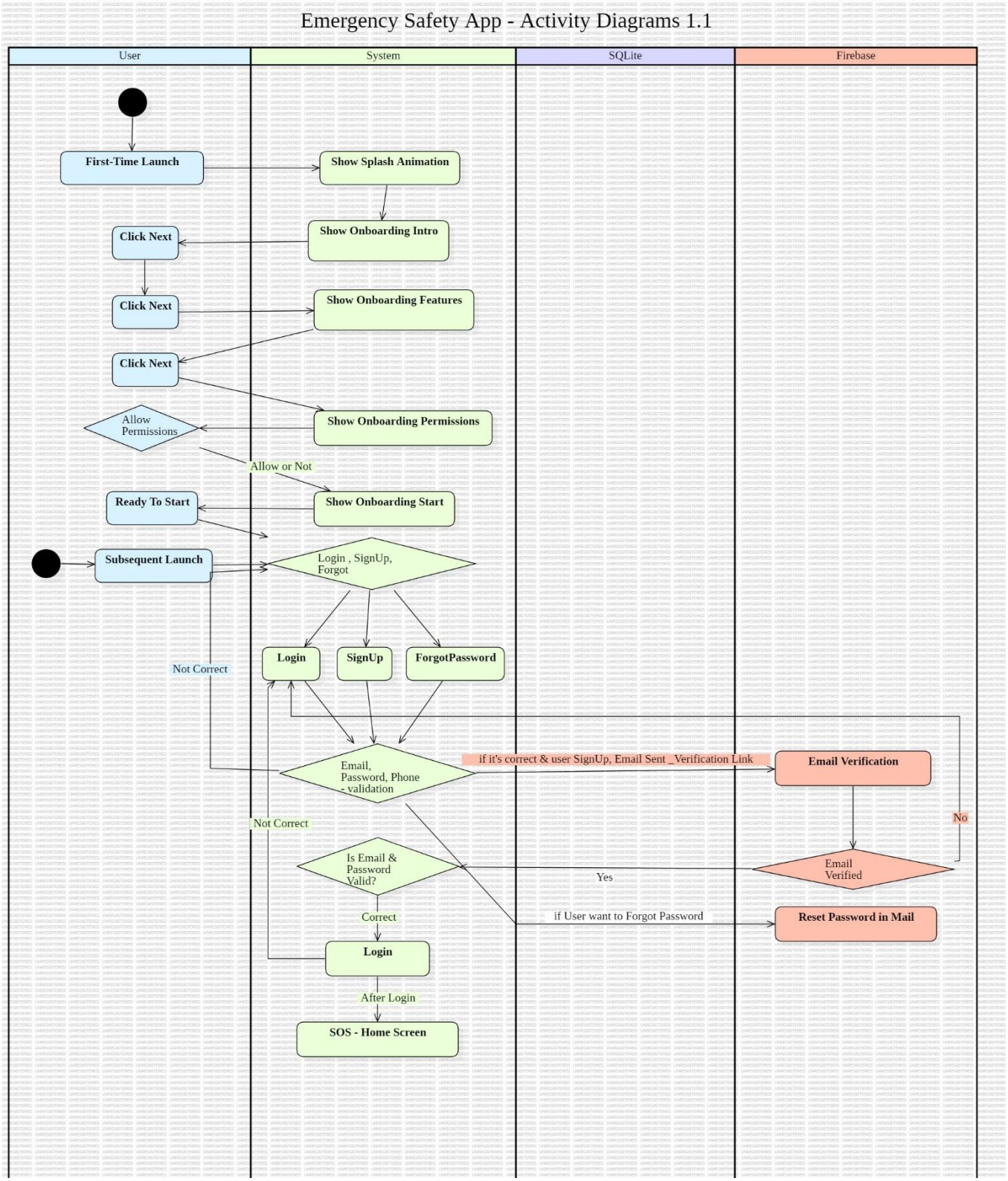

Figure B.1.5.1: Activity Diagram

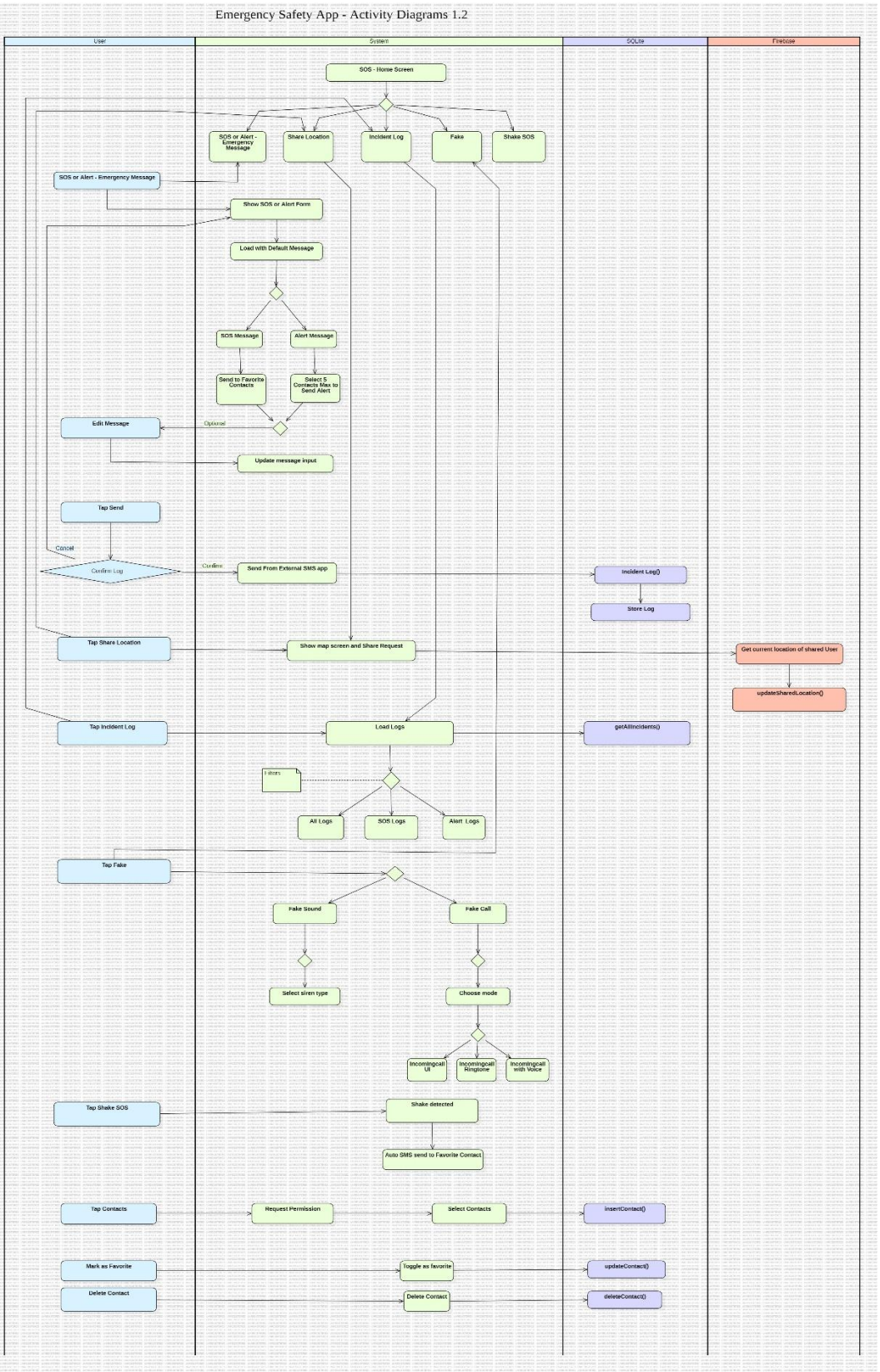

Figure B.1.5.2: Activity Diagram

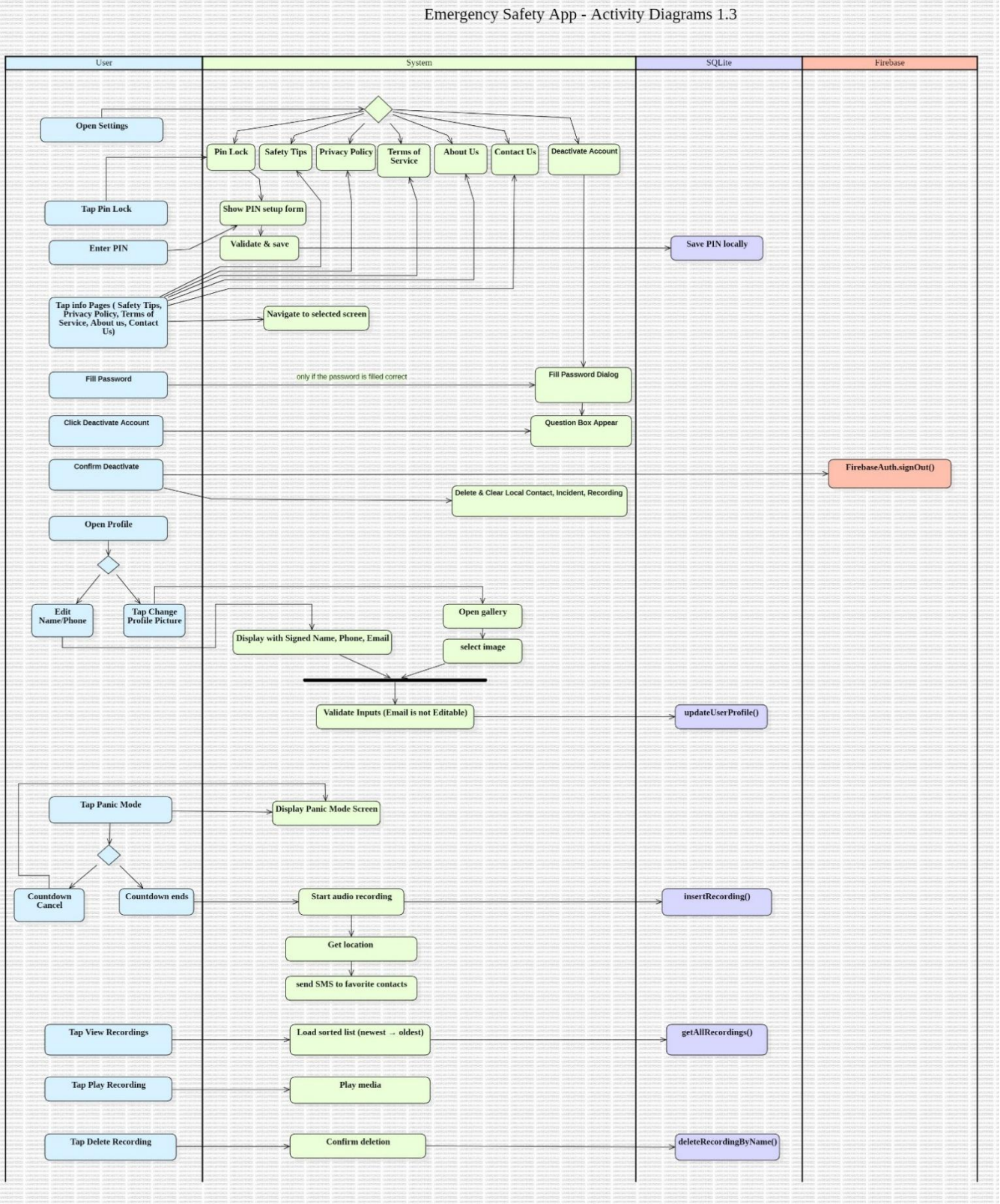

Figure B.1.5.3: Activity Diagram

Process Description

1. **User Registration & Authentication**
   - Users register using Firebase Authentication.
   - Upon successful signup, user details (name, email, hashed password, phone, profile pic path, role, verification status) are inserted into the local user's table.
   - Email verification is enforced before allowing login.
   - On login, passwords are hashed and stored locally; verification checked.
   - Firebase UID and normalized phone number are registered in Firestore users collection for further mapping and location sharing.
2. **Profile Management**
   - Users can view and update profile details: name, phone number, and profile picture.
   - Updates are saved in local SQLite users table.
   - Profile picture saved as a URI/path string.
   - Email is managed mostly through Firebase but synced/displayed locally.
3. **PIN Lock System**
   - Users set a 4-digit PIN stored securely in SharedPreferences.
   - On app start, if PIN exists but isn't verified, the app shows an unlock screen.
   - Successful PIN validation unlocks access to main app.
4. **Emergency Contacts**
   - Users add contacts from the system contact picker.
   - Contacts saved in local contacts table with boolean flags:
     - is_sos_contact
     - is_alert_contact
     - is_favorite
     - is_sharing_location
   - Contacts support multiple roles for emergency messaging and location sharing.

5. **Emergency Messaging**
   - SOS messages sent automatically to favorite contacts.
   - Alert messages sent to contacts explicitly selected for alerts.
   - If location permission granted, current location is appended as a Google Maps link.
   - Each message dispatch is logged in incident_log table with timestamp, contact info, message, location, and type.
6. **Fake Call & Sound Simulation**
   - Users can trigger fake calls:
     - Ringtone only
     - Silent screen
     - Ringtone + voice playback
   - Emergency sounds (police siren, fire engine, ambulance siren) can be played discreetly.
   - Provides realistic distraction and escape methods in emergency situations.
7. **Panic Mode**
   - User triggers panic mode, initiating a countdown then starting audio recording via MediaRecorder.
   - Sends SOS with location to favorite contacts.
   - Saves recording file info in the recordings table.
   - Allows viewing past recordings with playback and deletion functionality.
8. **Location Sharing**
   - Users share live location with selected contacts.
   - Sharing status tracked locally (contacts.is_sharing_location flag).
   - Location updates streamed to Firestore locationShares collection for subscribers.
   - Recipients can accept/decline sharing requests (also via Firestore).
   - Accepted sharings allow live location tracking on maps.

9. Incoming Location Requests

- Incoming share requests received through Firestore share_requests collection.
- Users accept or decline requests.
- Acceptance updates Firestore locationShares docs to enable live location sharing with that contact.
- Decline removes request documents.

10. Incident Log & Recording History

- Users can browse all sent emergency messages with filtering by type (SOS, Alert).
- Incident log data stored locally provides history.
- Recording history lists saved panic mode audio files with play/pause and delete options.

Data dictionary

1. Table: roles

Column	Type	PK	FK	Null	Description
role_id	INTEGER	PK		NO	Unique ID for the role
role_name	TEXT			NO	Name of the role (e.g., 'user', 'premium_user')

Usage: The rules table define user rules that can have various user types within the application Currently roles such as user and premium user service identifier for basic and advanced features in future updates the table will include additional rules to support advanced functionalities in paid fashion of the app.

2. Table: users

Column	Type	PK	FK	Null	Description
user_id	INTEGER	PK		NO	Unique user identifier
name	TEXT			NO	User's full name
email	TEXT			NO	User's email address (unique)
password_hash	TEXT			NO	Hashed user password
phone	TEXT			YES	User's phone number (optional)
profile_pic	TEXT			YES	URI or file path to the user's profile picture
pin_hash	TEXT			YES	Hashed PIN for app (optional)
is_verified	INTEGER			NO	1 if email verified, 0 otherwise
role_id	INTEGER		roles(role_id)	YES	Foreign key referencing user roles
created_at	DATETIME			YES	When user account was created

Usage: The users table is designed to store essential information about each user, including authentication data and profile details. Email and password_hash ensure secure access while tracking whether a user is verified by their email.

3. Table: emergency_contacts

Column	Type	PK	FK	Null	Description
contact_id	INTEGER	PK		NO	Unique identifier for the emergency contact
user_id	INTEGER		users(user_id)	NO	Owner user of this emergency contact
contact_name	TEXT			NO	Name of the emergency contact
contact_phone	TEXT			NO	Phone number of the emergency contact
is_favorite	INTEGER			NO	1 if marked as favorite, else 0

Usage: This table holds the emergency contacts for users, allowing quick access during critical situations. Each contact can be marked as a favorite, facilitating streamlined access in emergencies.

4. Table: contacts

Column	Type	PK	FK	Null	Description
id	INTEGER	PK		NO	Unique contact identifier
name	TEXT			YES	Contact name
phone_number	TEXT			YES	Contact phone number
is_sos_contact	INTEGER			NO	1 if marked as SOS contact, else 0
is_alert_contact	INTEGER			NO	1 if marked as alert contact, else 0
is_favorite	INTEGER			NO	1 if marked as favorite, else 0
is_sharing_location	INTEGER			NO	1 if user is sharing live location with this contact, else 0

Usage: Stores user contacts with multiple flags denoting their roles (SOS, alert, favorite, sharing location).

5. Table: default_messages

Column	Type	PK	FK	Null	Description
type	TEXT	PK		NO	Message type identifier (e.g., SOS_DEFAULT, ALERT_DEFAULT)
message_text	TEXT			YES	The saved default emergency message

Usage: The default_messages table is used to store predefined templates for emergency messages, ensuring users can quickly send alerts. By customizing these messages, users can tailor their emergency responses to suit different situations.

6. Table: incident_log

Column	Type	PK	FK	Null	Description
id	INTEGER	PK		NO	Unique incident ID
timestamp	TEXT			YES	Date and time the message was sent
contact_name	TEXT			YES	Name of the contact who received the message
contact_number	TEXT			YES	Phone number of the contact
message	TEXT			YES	Content of the message
location	TEXT			YES	Location string sent with the message (URL or text)
incident_type	TEXT			YES	Type of incident, e.g., "SOS" or "ALERT"

Usage: Logs outgoing emergency message incidents, their recipients, and message types.

7. Table: recordings

Column	Type	PK	FK	Null	Description
id	INTEGER	PK		NO	Unique recording ID
name	TEXT			YES	Recording filename or name
path	TEXT			YES	Filepath or URI to the audio file
timestamp	TEXT			YES	Date and time when the recording was saved

Usage: Stores information about emergency audio recordings.

Normalization

Normalization is a database design which is used to organize table and their relationship to minimize redundancy and dependency by dividing larger tables into smaller ones and also related tables. It help to improve the data integrity and reduce a normalize and ensure efficient data storage and retrieval.

Roles

role_id	role_name

User

user_id	name	email	password_hash	phone	profile_pic	pin_hash	is_verified	role_id	created_at

Emergency_Contacts

contact_id	user_id	contact_name	contact_phone	is_favorite

App Contacts

id	name	phone_number	is_sos_contact	is_alert_contact	is_favorite	is_sharing_location

Default Messages

type	message_text

Incident Log

id	timestamp	contact_name	contact_number	message	location	incident_type

Recordings

id	name	path	timestamp

No.	Table Name	1NF	2NF	3NF	Primary Keys	Dependencies / Notes
1	Users	✔	✔	✔	user_id	role_id is a foreign key to Roles. All other fields depend directly on user_id.
2	Roles	✔	✔	✔	role_id	Contains only atomic fields. No transitive dependencies.
3	Emergency Contacts	✔	✔	✔	contact_id	user_id is a foreign key to Users. All fields depend directly on contact_id.
4	Contact(App)	✔	✔	✔	id	Flags (is_sos_contact, is_alert_contact, etc.) are atomic and refer only to contact.
5	Default Messages	✔	✔	✔	type	No foreign keys. Each message is atomic and keyed by type.
6	Incident Log	✔	✔	✔	id	No transitive dependencies. Contact info is denormalized intentionally for logging.
7	Recordings	✔	✔	✔	id	Each recording has atomic fields. No foreign keys or derived attributes.

Table : Normalization Table

Entity Relationship Diagram

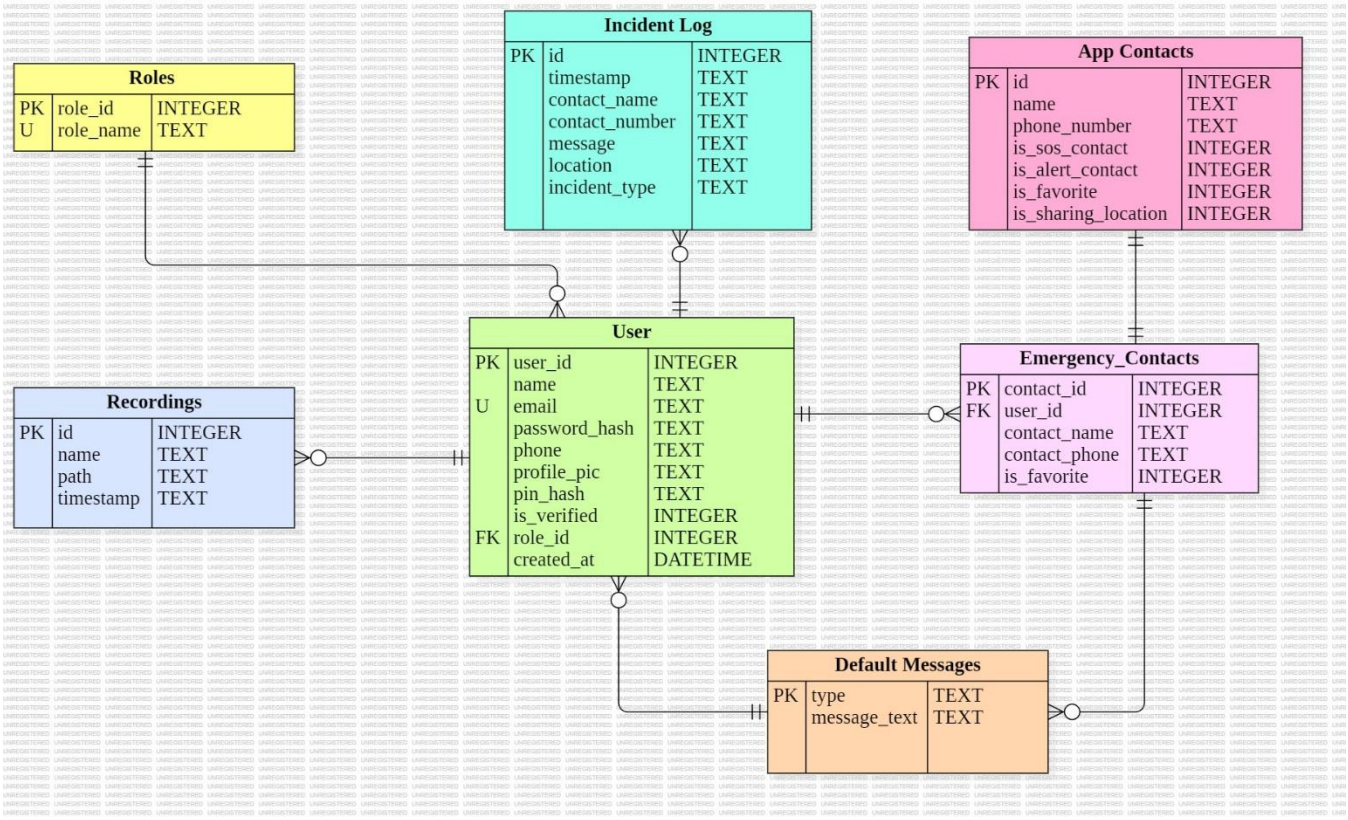

Figure B.1.6 : Entity Relationship Diagram

Section C

1. System Design

1.1. Database Design

A. Database Table from Database Helper 1

Table: android_metadata

	locale
	Filter
1	en_US

Table A.1 : Android Metadata

Table: contacts

Filter in any column

	id	name	phone_number	is_sos_contact	is_alert_contact	is_favorite	is_sharing_location
	Fil...	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Sneha Ella	+959781303620	0	0	0	0
2	2	Sameer	09940923255	0	1	1	0
3	3	Sandayar	09790496575	0	1	0	0
4	4	Aster	09781304080	0	1	1	0

Table A.2 : Contacts

Table: default_messages

Filter in any column

	type	message_text
	Filter	Filter
1	SOS_DEFAULT	Emergency! I need help immediately. Please come to my location.
2	ALERT_DEFAULT	This is an alert message. I might be in trouble. Please check on me.

Table A.3 : Default Messages

Table: incident_log

	id	timestamp	contact_name	contact_number	message	location	incident_type
1	1	2025-08-07 17:14:28	Sameer	09940923255	Emergency! I need help immediately. Please come to my location.	Location: https://maps.google.com/?q=16.7901489,96.1965971	SOS
2	2	2025-08-07 17:14:28	Aster	09781304080	Emergency! I need help immediately. Please come to my location.	Location: https://maps.google.com/?q=16.7901489,96.1965971	SOS
3	3	2025-08-07 17:14:43	Sameer	09940923255	This is an alert message. I might be in trouble. Please check on me.	Location: https://maps.google.com/?q=16.7901459,96.1966012	ALERT
4	4	2025-08-07 17:14:43	Sandayar	09790496575	This is an alert message. I might be in trouble. Please check on me.	Location: https://maps.google.com/?q=16.7901459,96.1966012	ALERT
5	5	2025-08-07 17:14:43	Aster	09781304080	This is an alert message. I might be in trouble. Please check on me.	Location: https://maps.google.com/?q=16.7901459,96.1966012	ALERT

Table A.4 : Incident Log

Table: recordings

	id	name	path	timestamp
1	1	panic_20250807_171459.mp3	/storage/emulated/0/Android/data/com.example.sampleotfp/files/Music/panic_20250807_171459.mp3	2025-08-07 17:15:06
2	2	panic_20250807_171512.mp3	/storage/emulated/0/Android/data/com.example.sampleotfp/files/Music/panic_20250807_171512.mp3	2025-08-07 17:15:17

Table A.5 : Recordings

Table: sqlite_sequence

	name	seq
1	contacts	4
2	incident_log	5
3	recordings	2

Table A. 6 : SQLite Sequence

B. Database Table from Database Helper 2

Table: android_metadata

	locale
	Filter
1	en_US

Table B. 1 : Android Metadata

Table: emergency_contacts

contact_id	user_id	contact_name	contact_phone	is_favorite

Table B. 2 :Emergency Contacts

Table: roles

	role_id	role_name
	Filter	Filter
1	1	user
2	2	premium_user

Table B. 3 : Roles

Table: sqlite_sequence

	name	seq
	Filter	Filter
1	roles	2
2	users	1

Table B. 4 : SQLite Sequence

Table: users

Filter in any column

	user_id	name	email	password_hash	phone	profile_pic	pin_hash	is_verified	role_id	created_at
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	User102	sampleuser000102@gmail.com	5053d0ef5668e0f953d63efad7ced568e2bf...	+959783557560	default_pfp	NULL	1	1	2025-08-07 10:41:02

Table B. 5 : Users

C. Database Table from Firebase

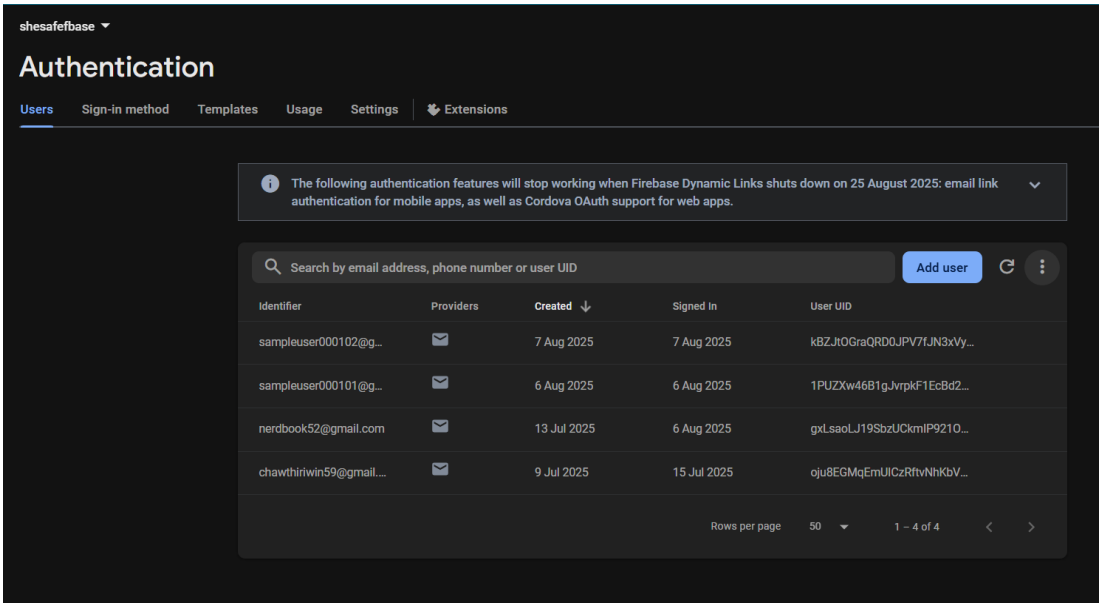

Table C.1 : Authentication

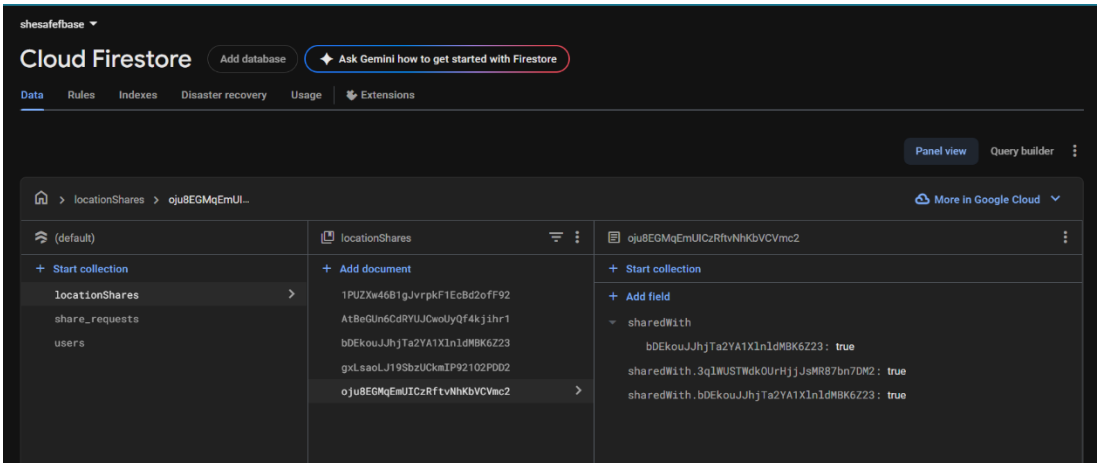

Table C.2 : Cloud Firestore - 1

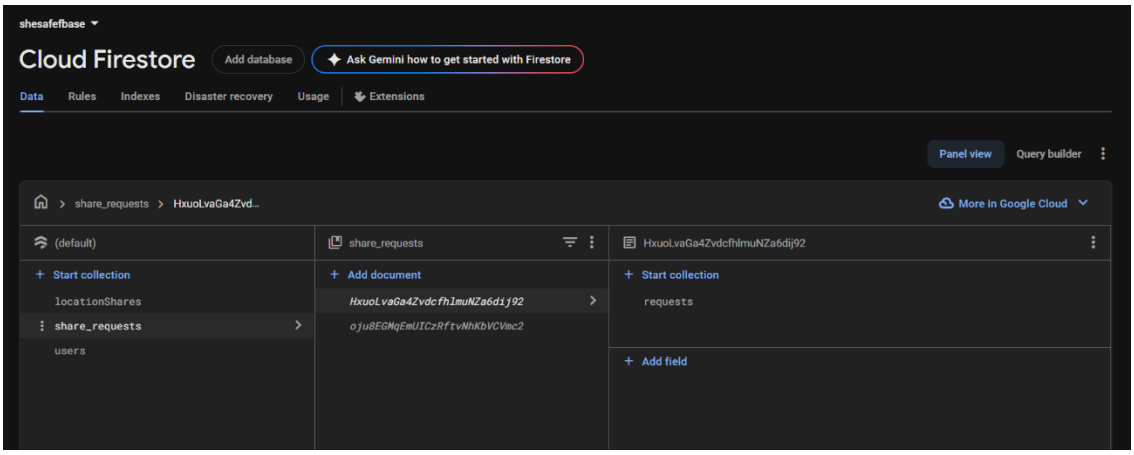

Table C.3 : Cloud Firestore - 2

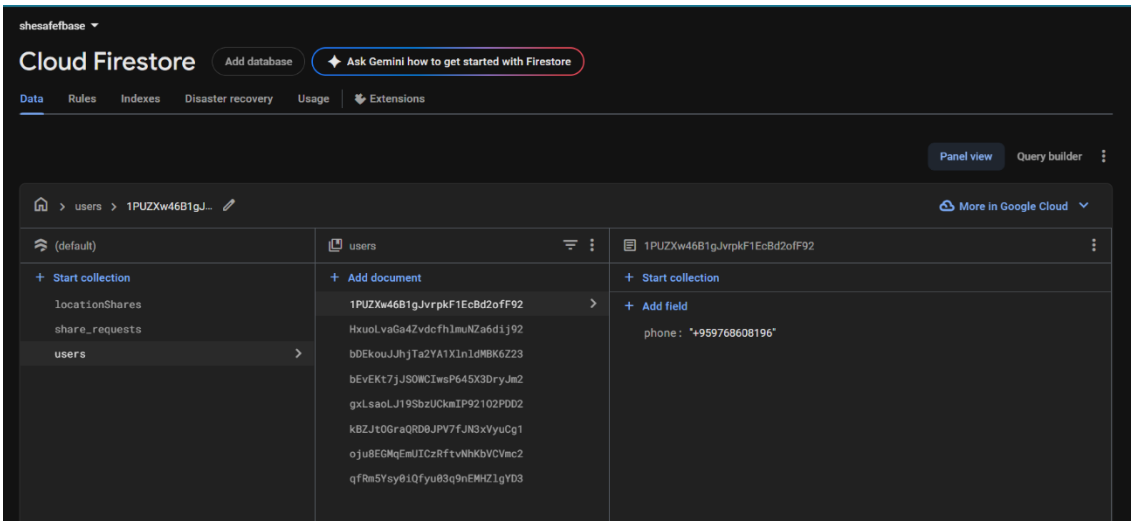

Table C.3 : Cloud Firestore - 3

1.2. Hierarchy Chart

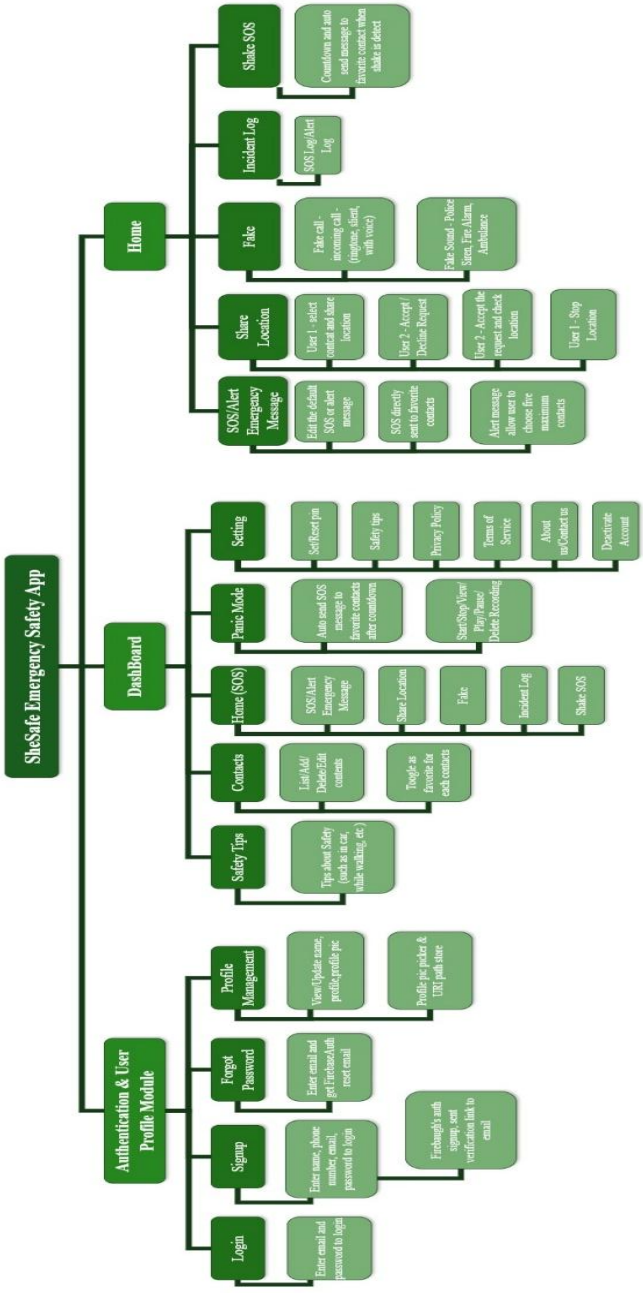

Table 1.2.1 : Hierarchy Chart

1.3. Mini-specifications

1) **Authentication and User Management**

- Activities: LoginActivity, SignupActivity, ForgotPasswordActivity
- Features:

  - Firebase email/password login.
  - Local password hashing with SHA-256.
  - Sign Up Email verification before Login.
  - Password reset via Firebase.
  - Profile management with picture and contact info (ProfileFragment).

2) **Onboarding Flow**

- Activities: Splash2Activity, WelcomeActivity, FeaturesActivity, PermissionsActivity, StartActivity
- Manages first-time launch, animations, permission prompts, and onboarding status.

3) **Emergency Messaging**

- Fragment: EmergencyMessageFragment
- Features:

  - Create/save SOS and Alert messages.
  - SOS auto-sends to favorites; Alert lets user select contacts.
  - Handles permissions for SMS and location.
  - Live GPS fetching with retry and timeout.
  - SMS sending with fallback to default SMS app.
  - Logs events into incident history.

4) **Contact Management**

- Fragment: ContactFragment
- Adapters: ContactsAdapter, ConfirmedIncomingAdapter, SharedContactsIncomingAdapter, SharedContactsOutgoingAdapter

  - Uses SQLite to store emergency contacts and share statuses.
  - Permission handling for reading contacts.
  - Add/delete contacts, mark as favorite, and multi-select.
  - Manage per-contact sharing location status.

5) **Location Sharing and Live Map**

- Fragments: ShareLocationFragment, SharedWithMeFragment, LiveLocationMapFragment

  - Real-time updates via Firebase Firestore.
  - OSMDroid used for showing location markers.
  - Allows starting/stopping live location sharing with trusted contacts.

6) **Panic and Shake SOS Modes**

- Fragments: PanicModeFragment, ShakeSOSFragment

  - Panic Mode records audio and sends SOS after countdown.
  - Shake detection triggers countdown and SOS with location.
  - Handles permissions and logs incidents.
  - Includes audio history viewing.

7) **Fake Call and Sound Simulation**

- Fragments: FakeHomeFragment, FakeCallSelectorFragment, FakeCallFragment, FakeSoundFragment

  - Simulates incoming calls with ringtone/voice options.
  - Plays police/fire/ambulance sirens.
  - Includes help popups and guided UI.

8) **Incident Log and History**

- Fragment: IncidentLogFragment

  - Lists sent SOS/Alert messages with filters.
  - RecordingHistoryFragment shows audio log history.

9) **Settings and Security**

- Fragment: SettingFragment

  - PIN lock management via SetPinLockFragment and UnlockPinFragment.
  - Access to legal and info pages: Privacy Policy, Terms, About Us, Safety Tips, Contact Us.
  - Account deactivation requires password confirmation.

Database Specifications:

- **DatabaseHelper1**:
  - Stores user login credentials and linked emergency contacts.
  - Manages user roles and authentication.
- **DatabaseHelper2**:
  - Stores:

    - Emergency contacts with status flags (SOS, Alert, Favorite, SharingLocation).
    - Default emergency messages.
    - Incident logs with location/message info.
    - Audio recording metadata.

Third-Party and System Integrations:

- Firebase Authentication and Firestore for user/session management and location data.
- Android runtime permissions for SMS, location, contacts, and audio.
- OSMDroid for map display and markers.
- Android Sensors (accelerometer) for shake detection.
- Media APIs for recording and playback of audio.

UI and UX Features:

- MainActivity manages tab-based bottom navigation.
- Modular UI with Fragment transitions.
- Help dialogs and animated onboarding flow.
- Safety tips and expandable settings.
- Confirmation dialogs for critical actions (e.g., sending SOS or account deactivation).

1.4. Interface design

Figure 1 : On-Boarding Screen 1

Figure 2 : On-Boarding Screen 2

Figure 3 : On-Boarding Screen 3

Figure 4 : On-Boarding Screen 4

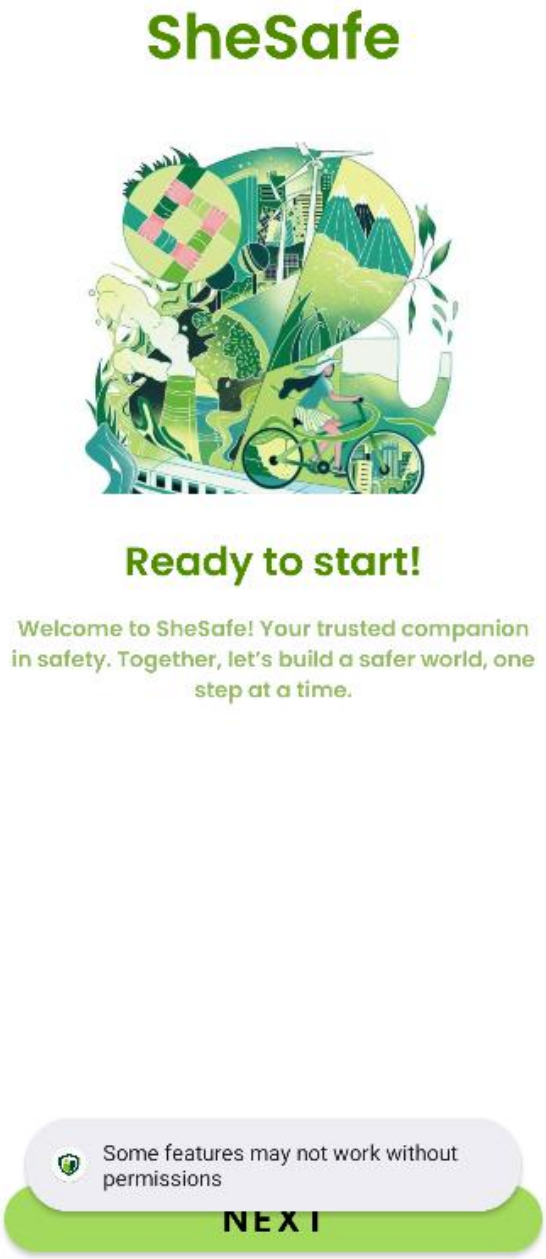

Figure 5 : Ready to Start Screen

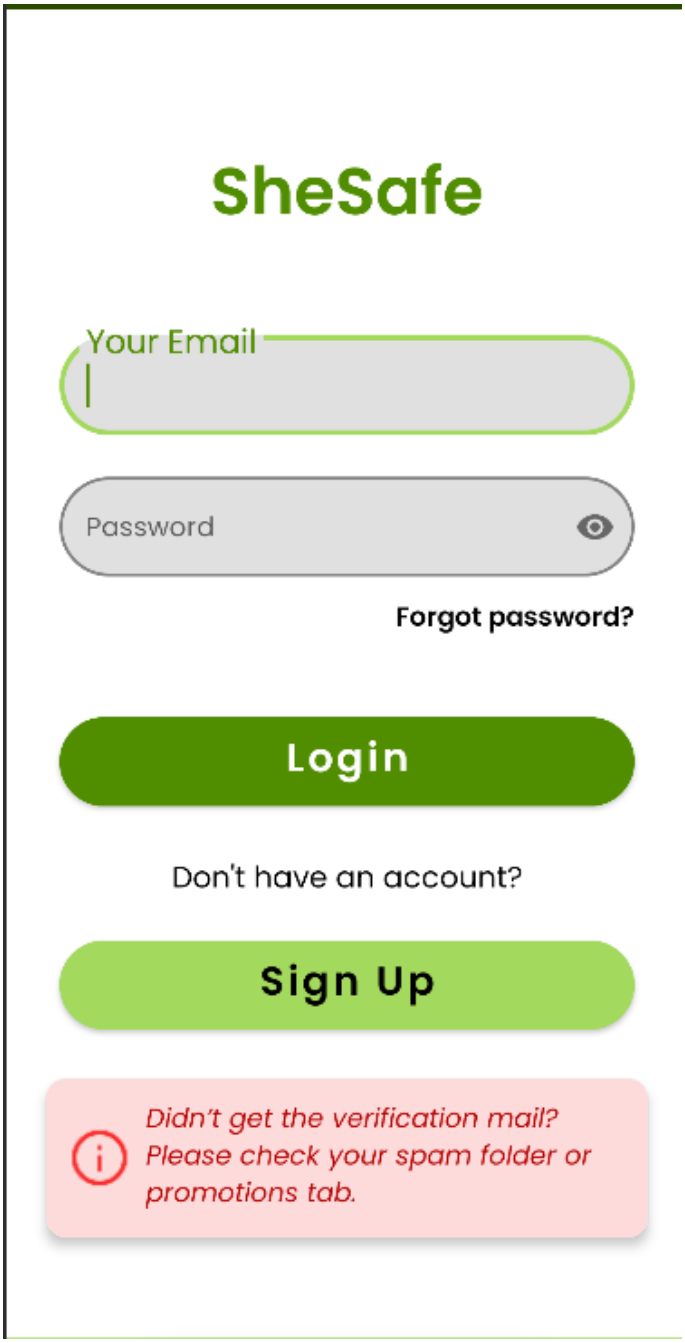

Figure 6 : Login Screen

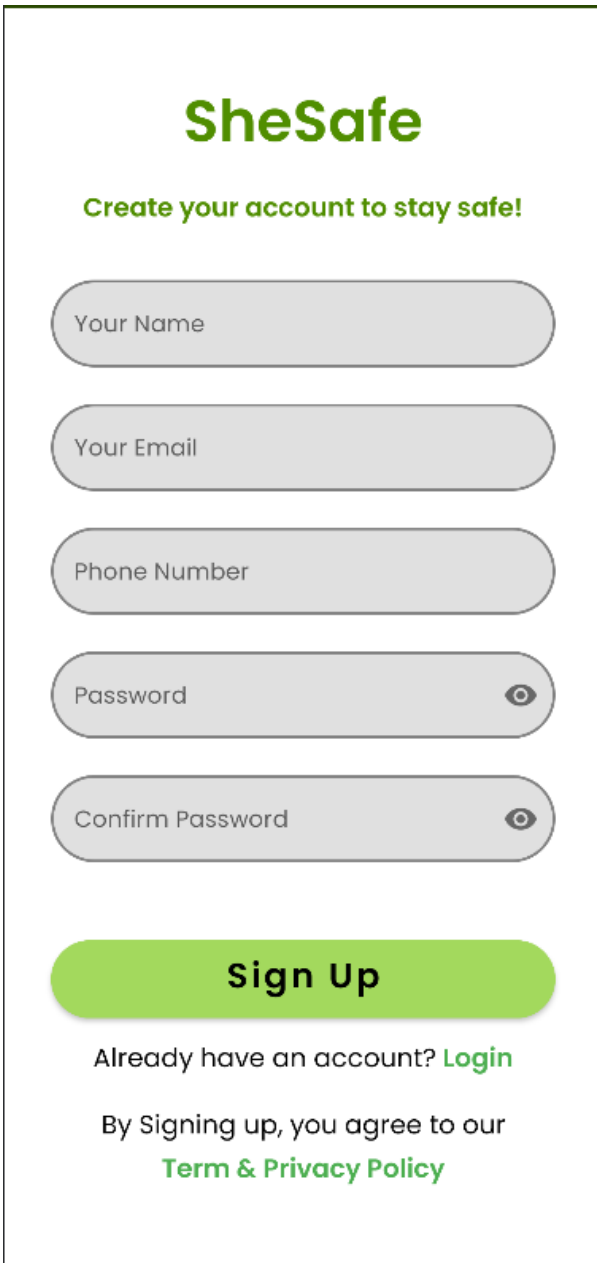

Figure 7 : SignUp Screen

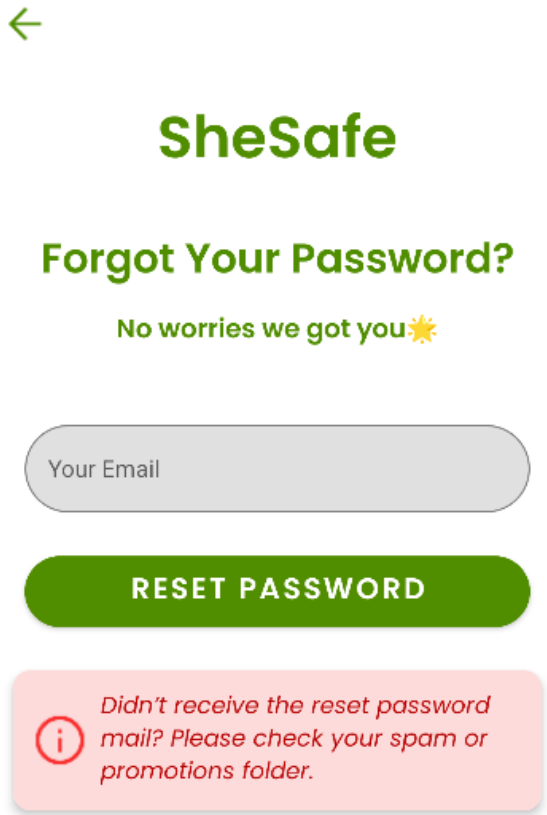

Figure 8 : Forgot Password Screen

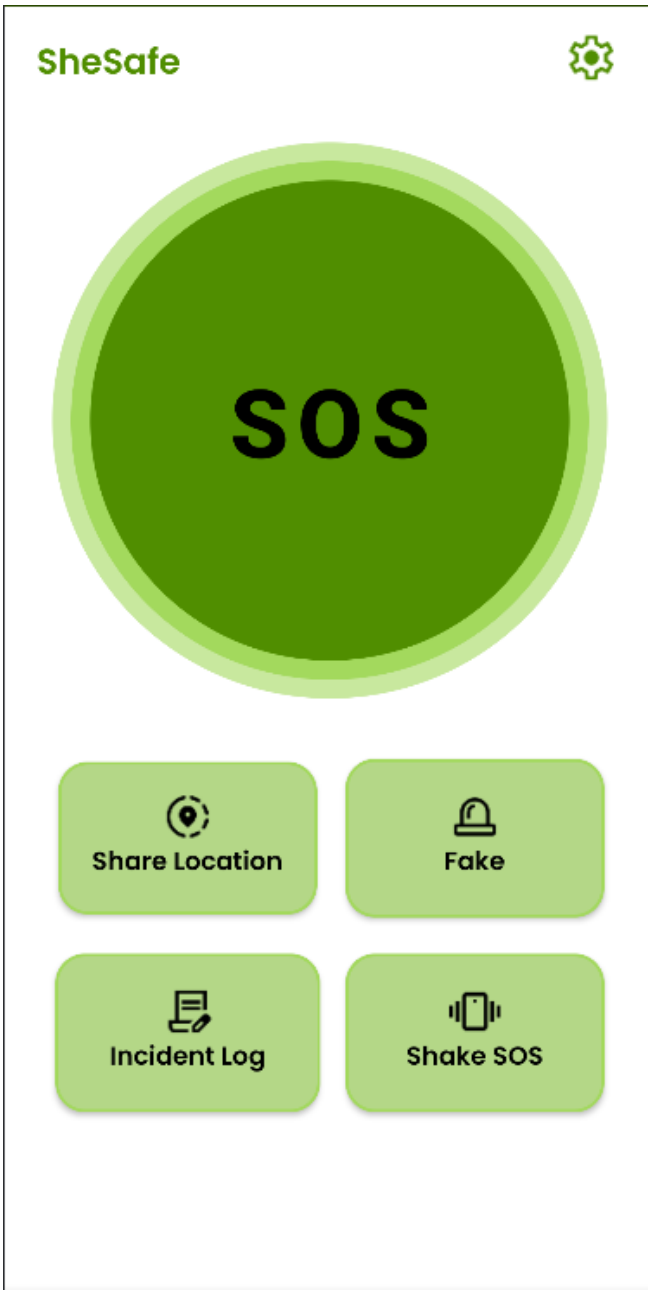

Figure 9 : Home Screen

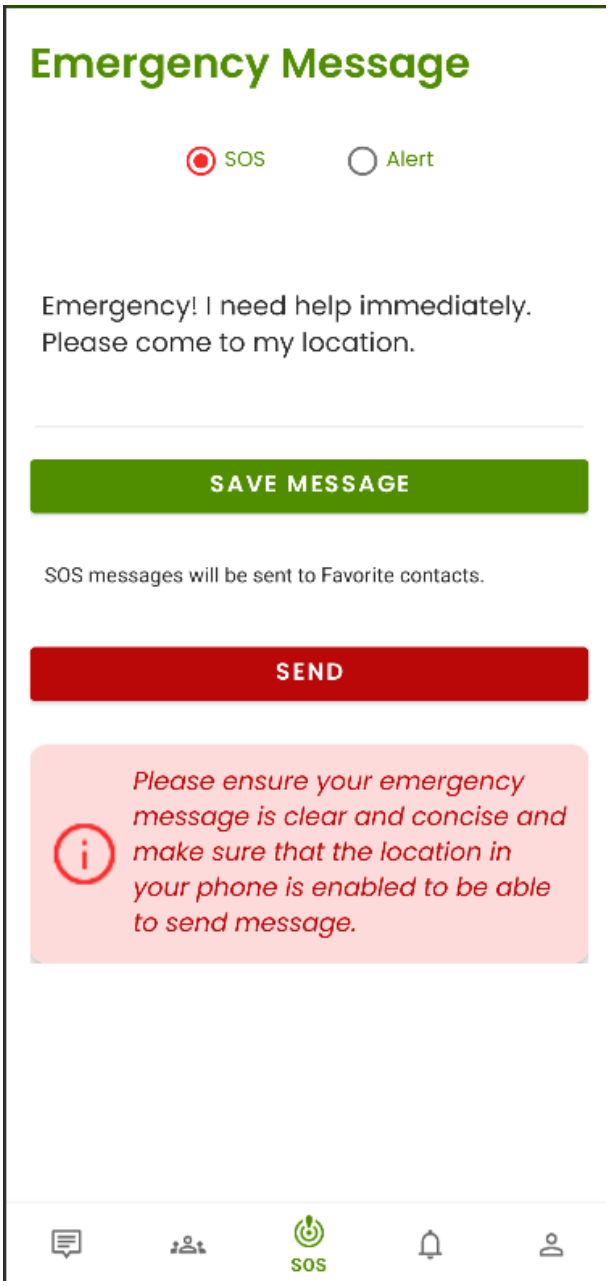

Figure 10 : Emergency Message SOS Screen

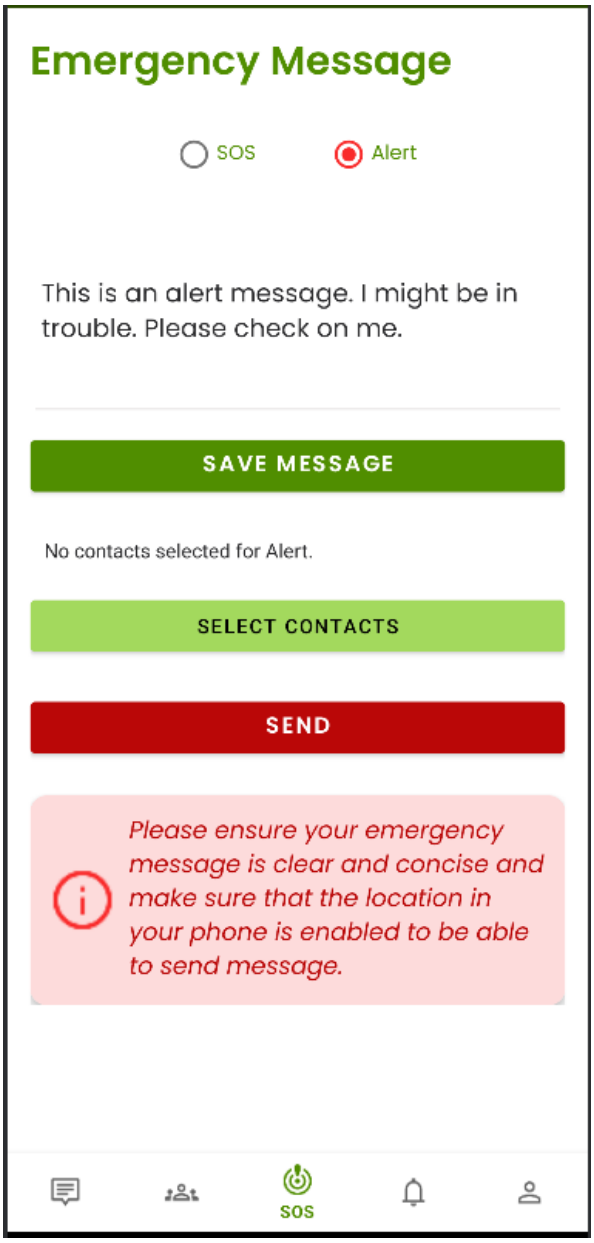

Figure 11 : Emergency Message Alert Screen

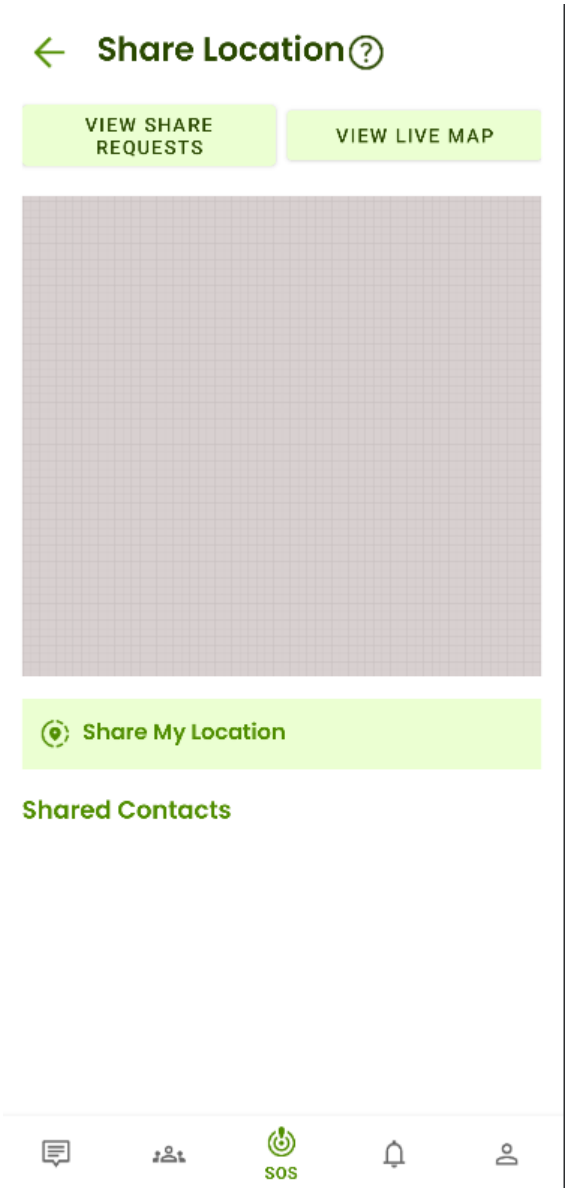

Figure 12 : Share Location User 1 Screen

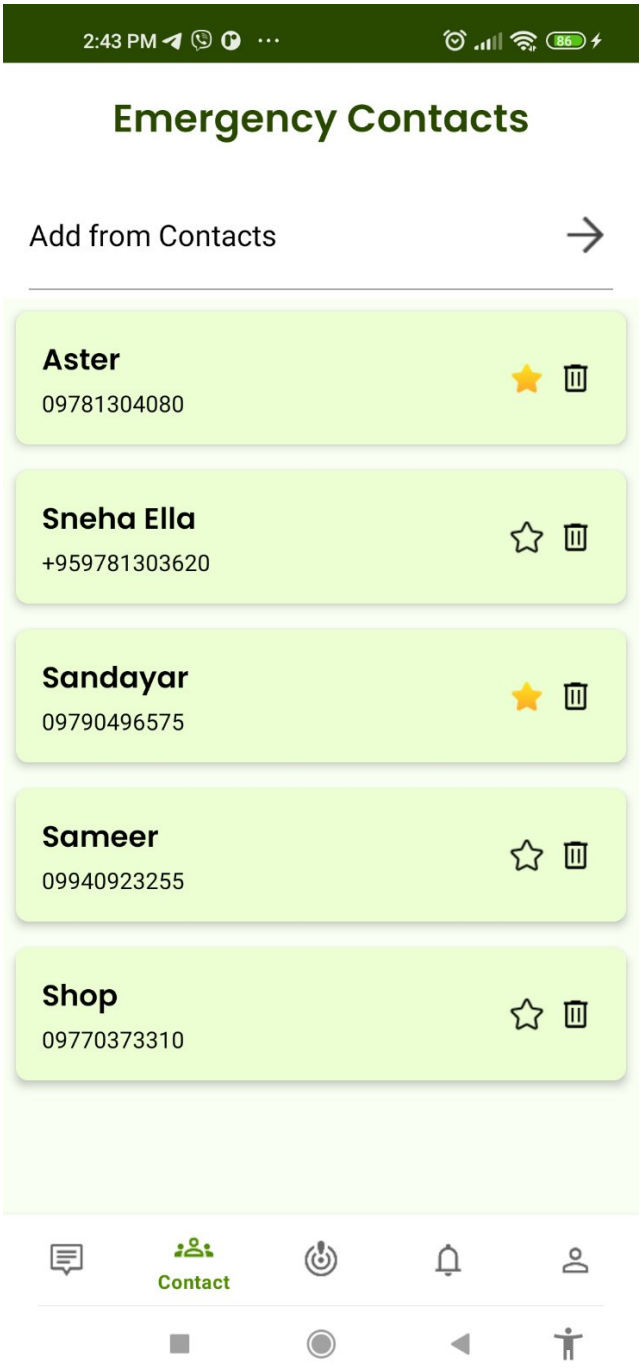

Figure 13 : Emergency Contacts Screen

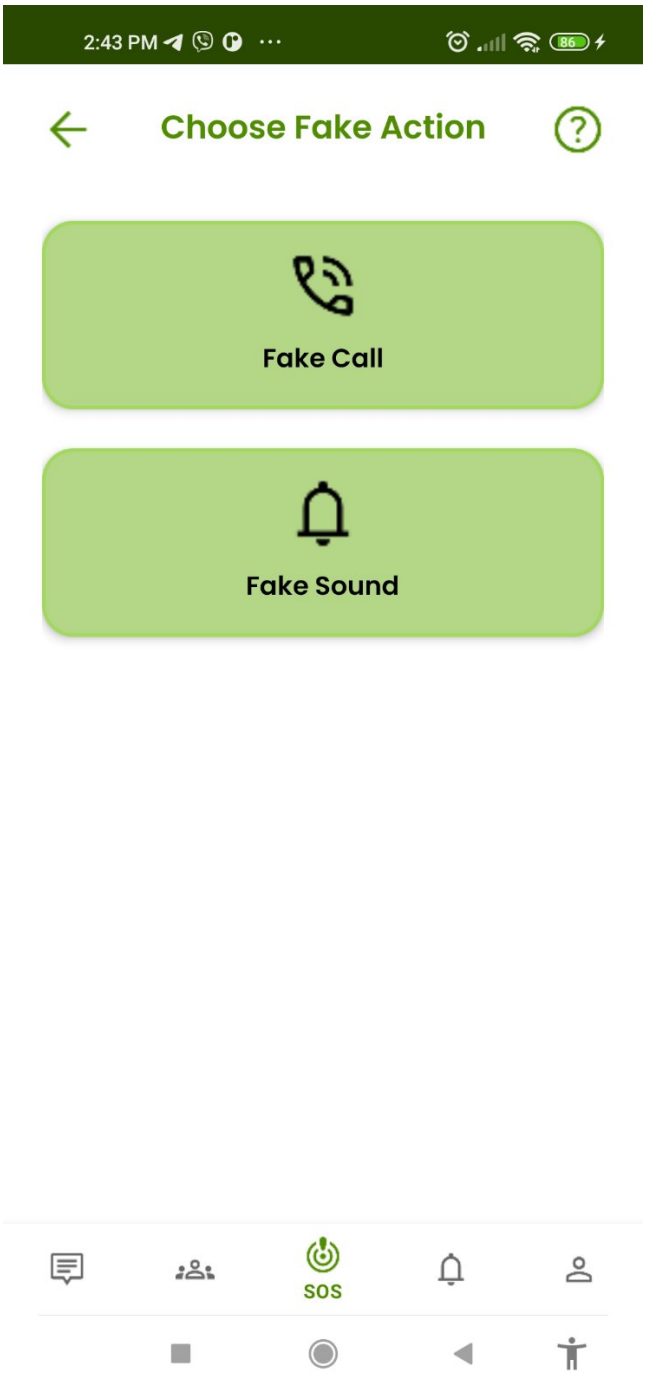

Figure 14 : Fake Home Screens

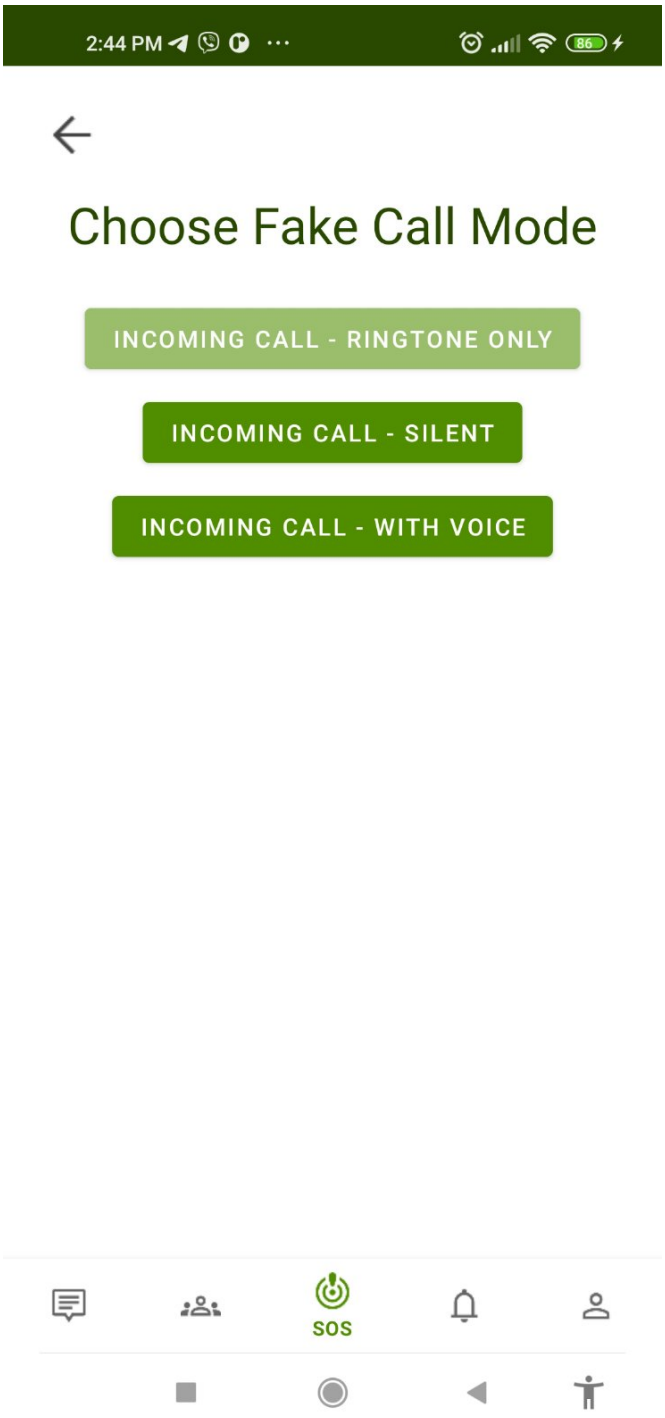

Figure 15 : Fake Call Screen

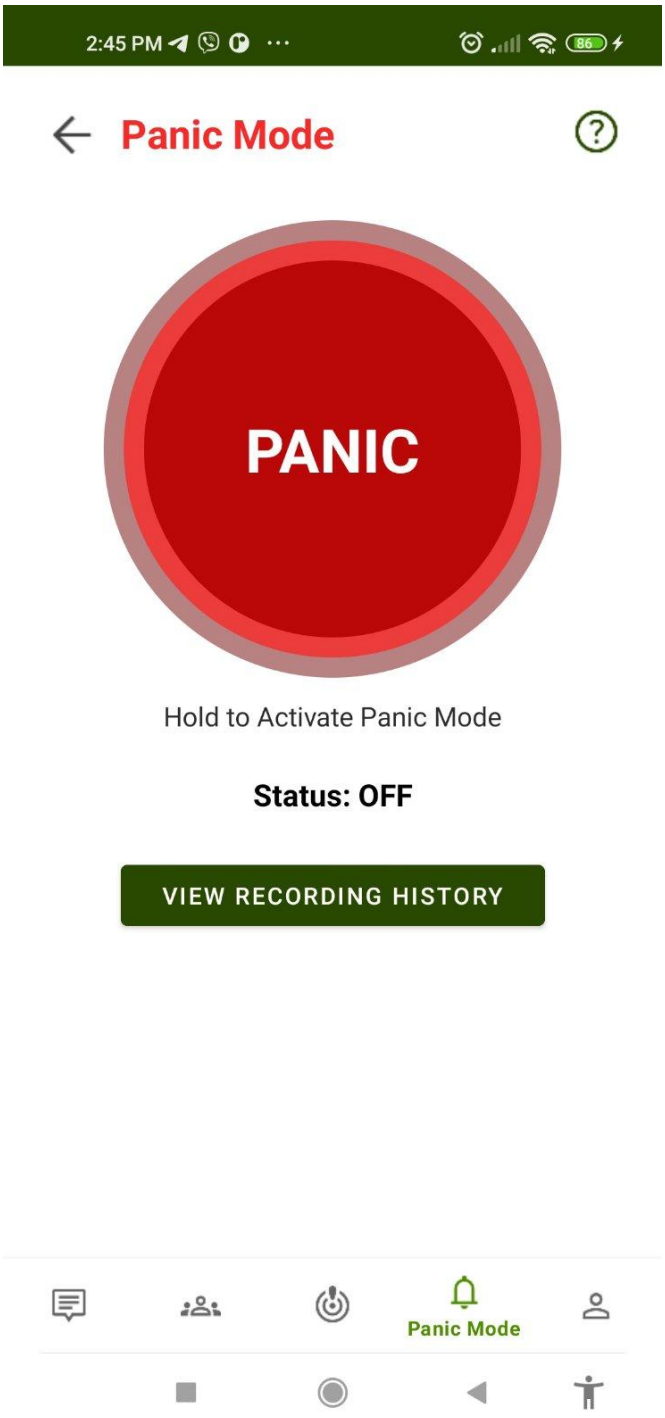

Figure 16 : Panic Mode Screen

Figure 17 : Recording History Screen

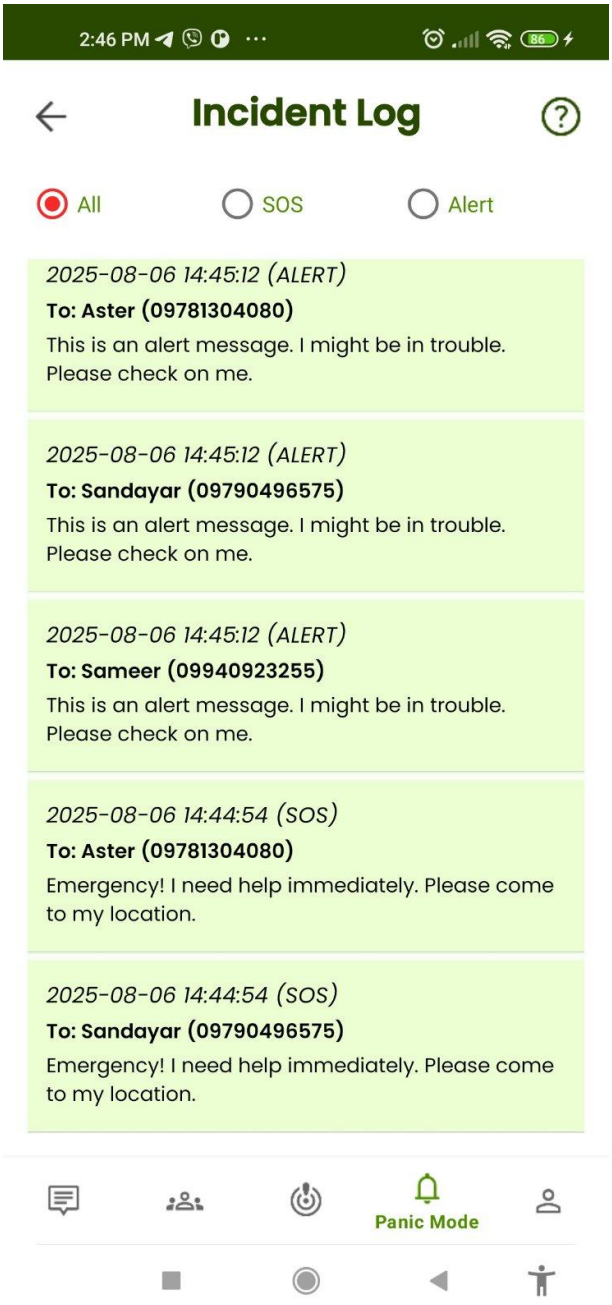

Figure 18 : Incident Log(All) Screen

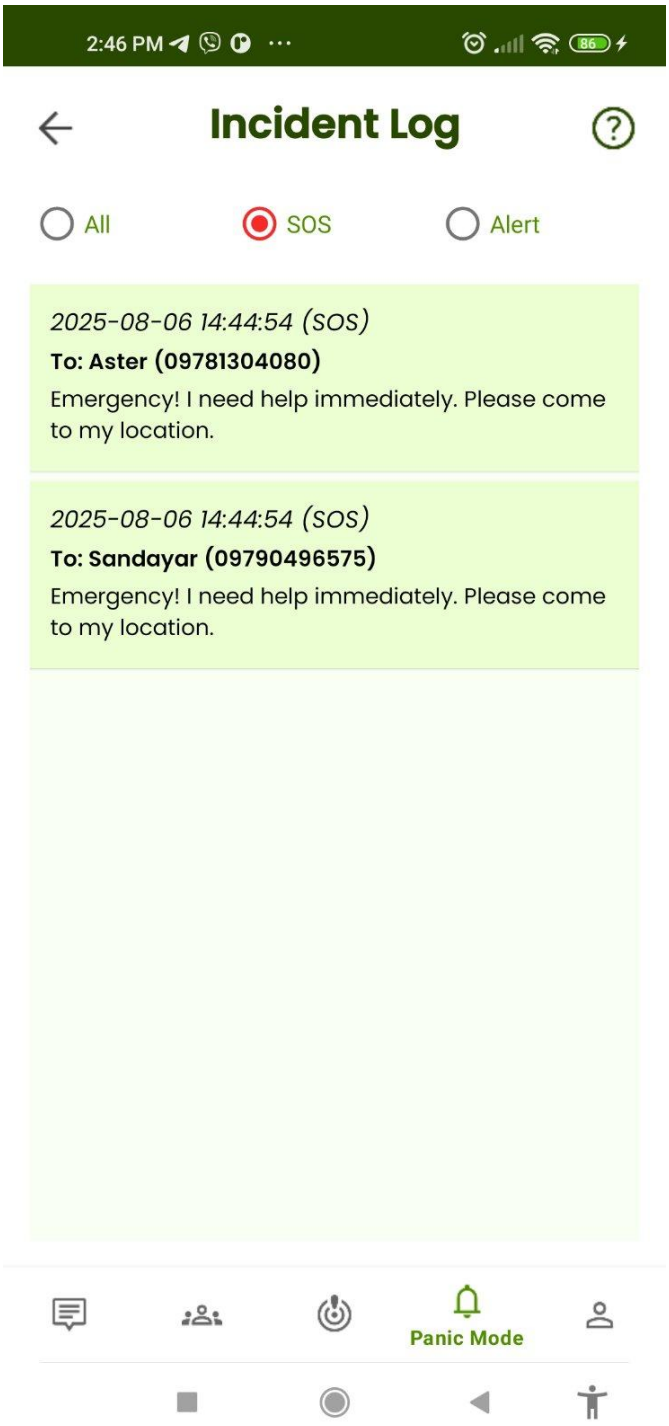

Figure 19 : Incident Log(SOS) Screen

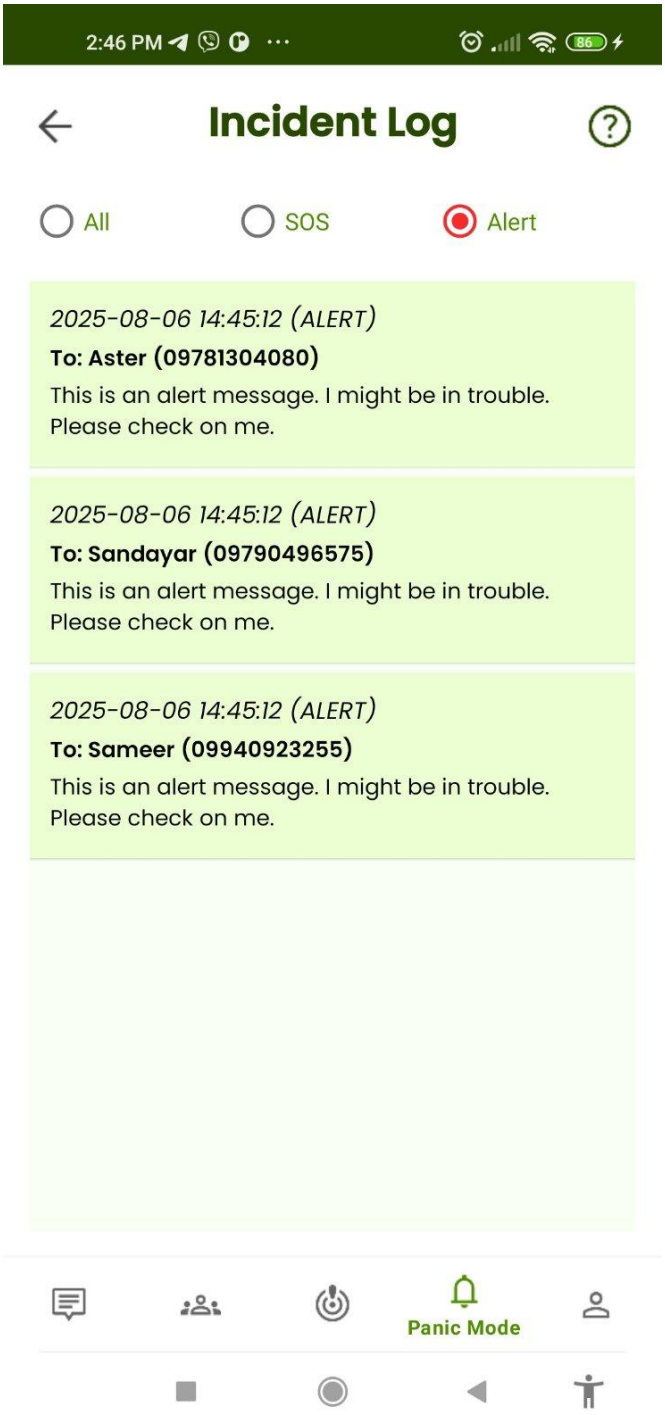

Figure 20 : Incident Log(Alert) Screen

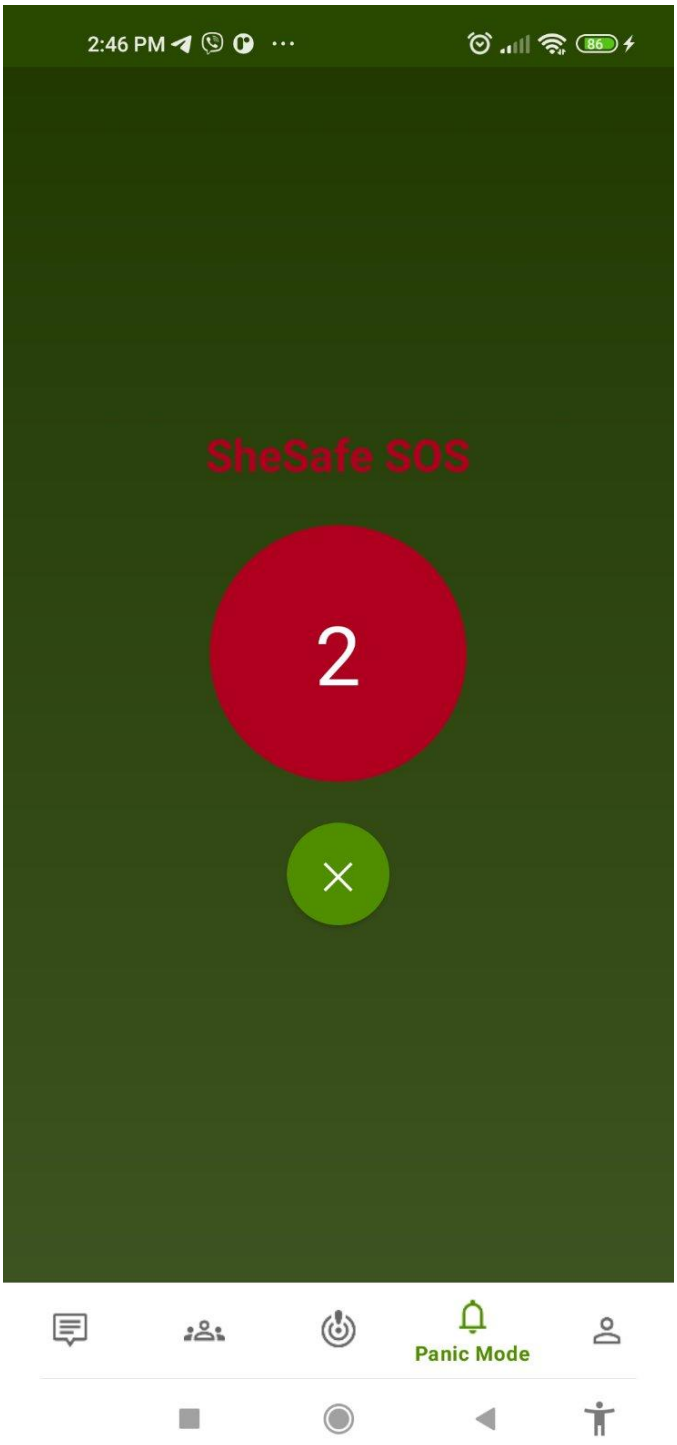

Figure 21 : Shake SOS Countdown Screen

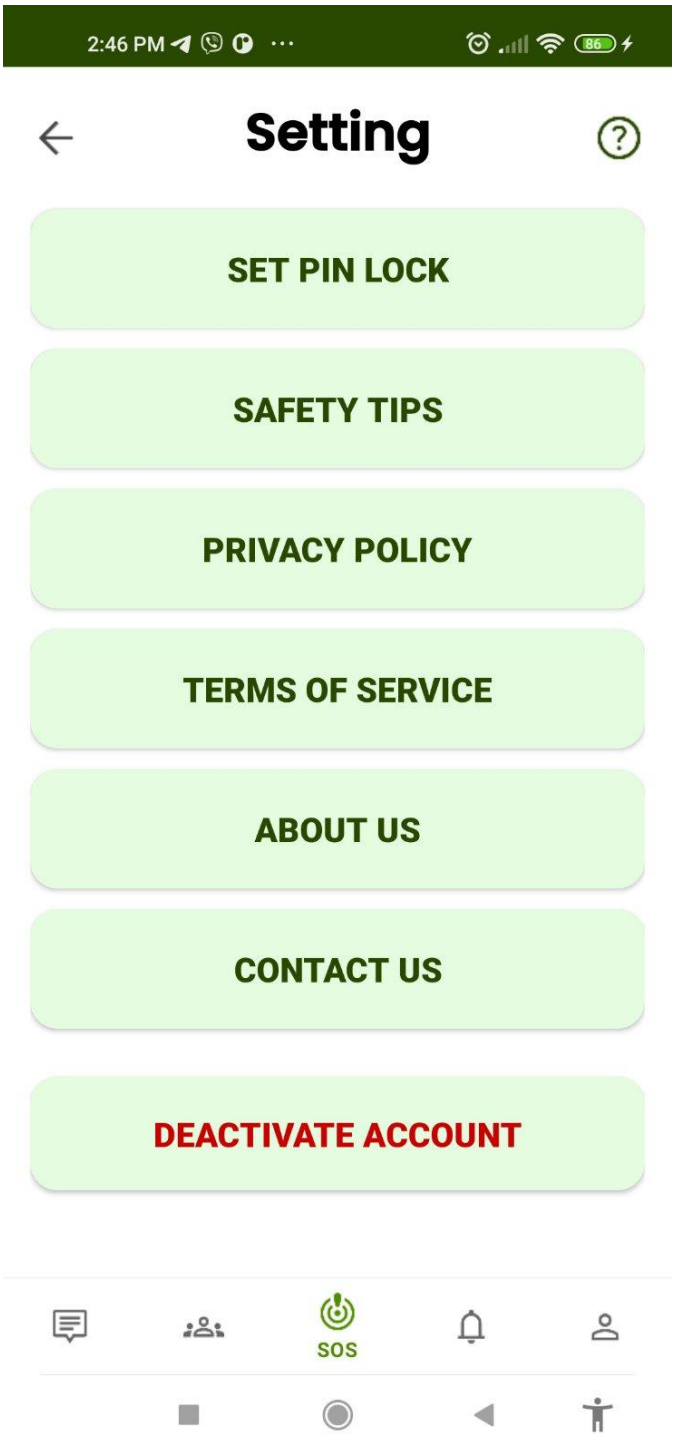

Figure 22 : Setting Home Screen

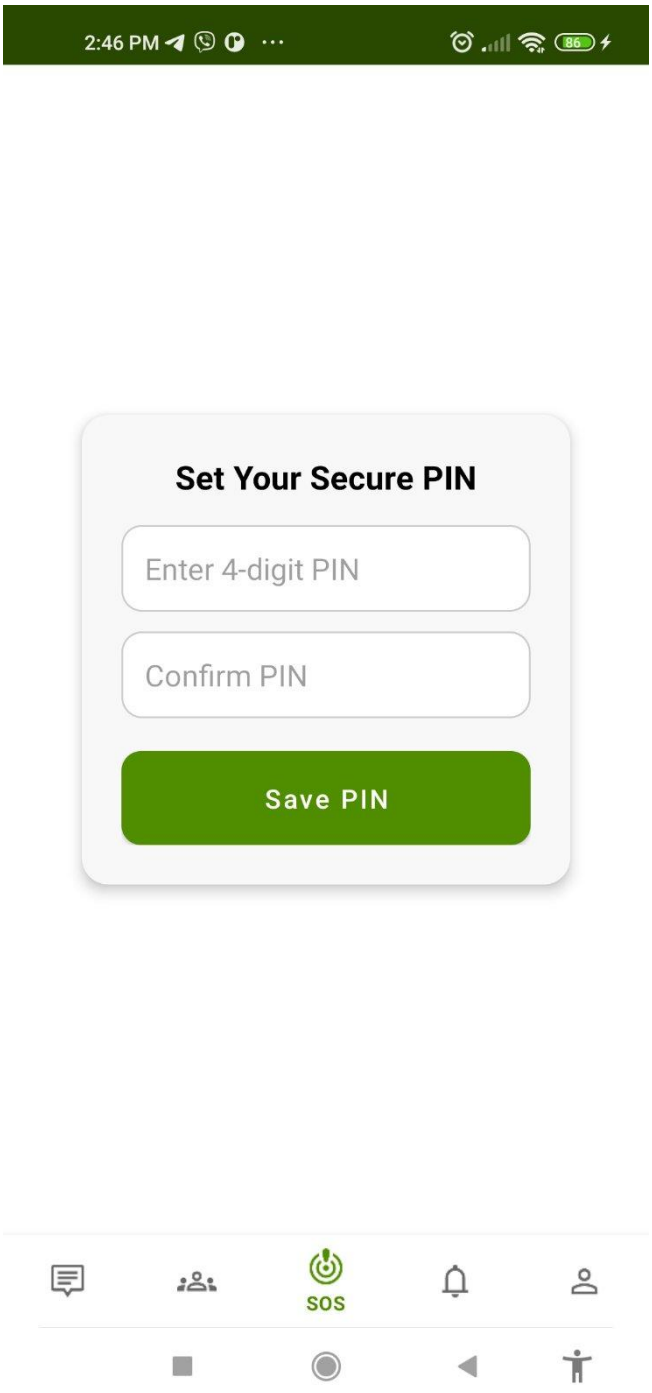

Figure 23 : Add Pin Screen

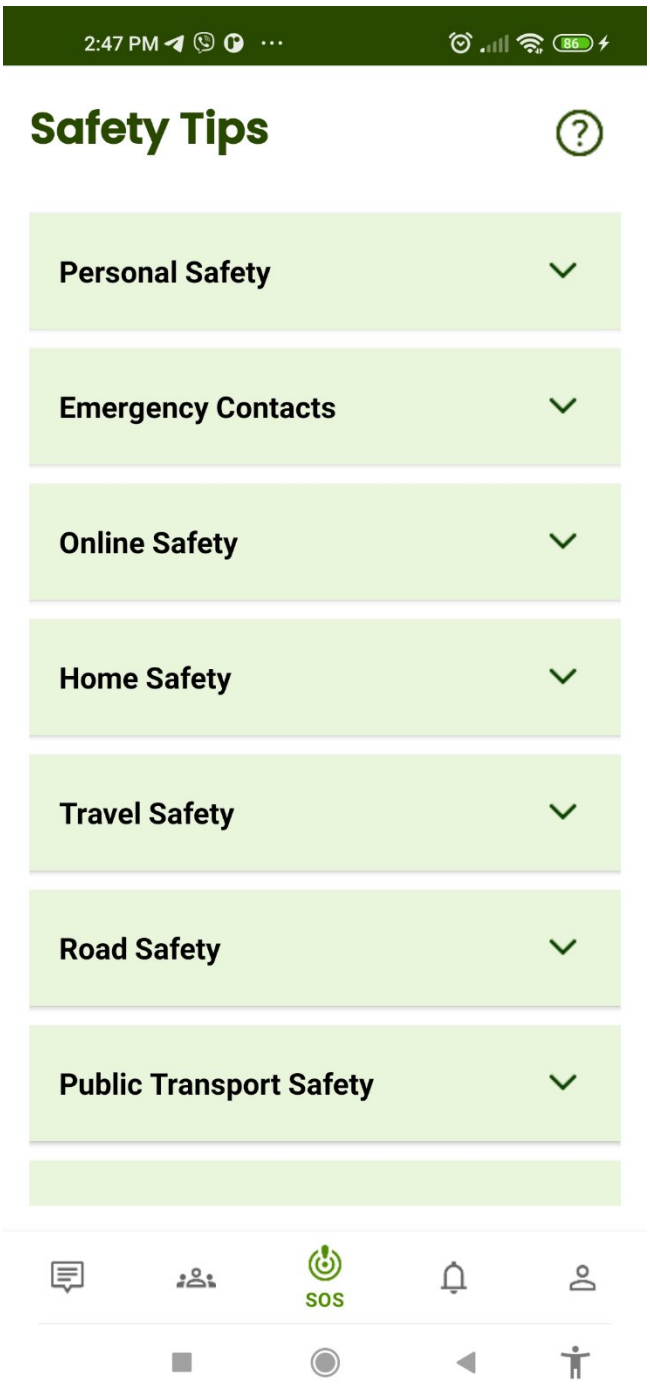

Figure 24 : Safety Tips Screen 1

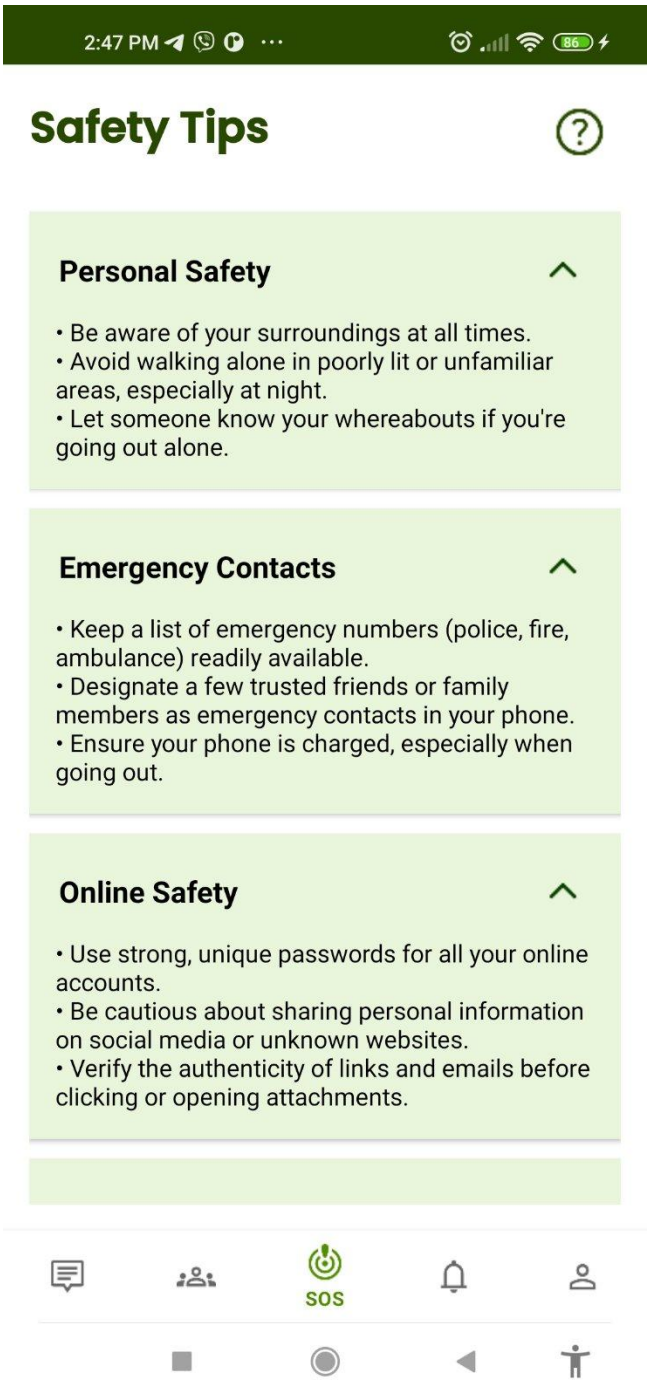

Figure 25 : Safety Tips Screen 2

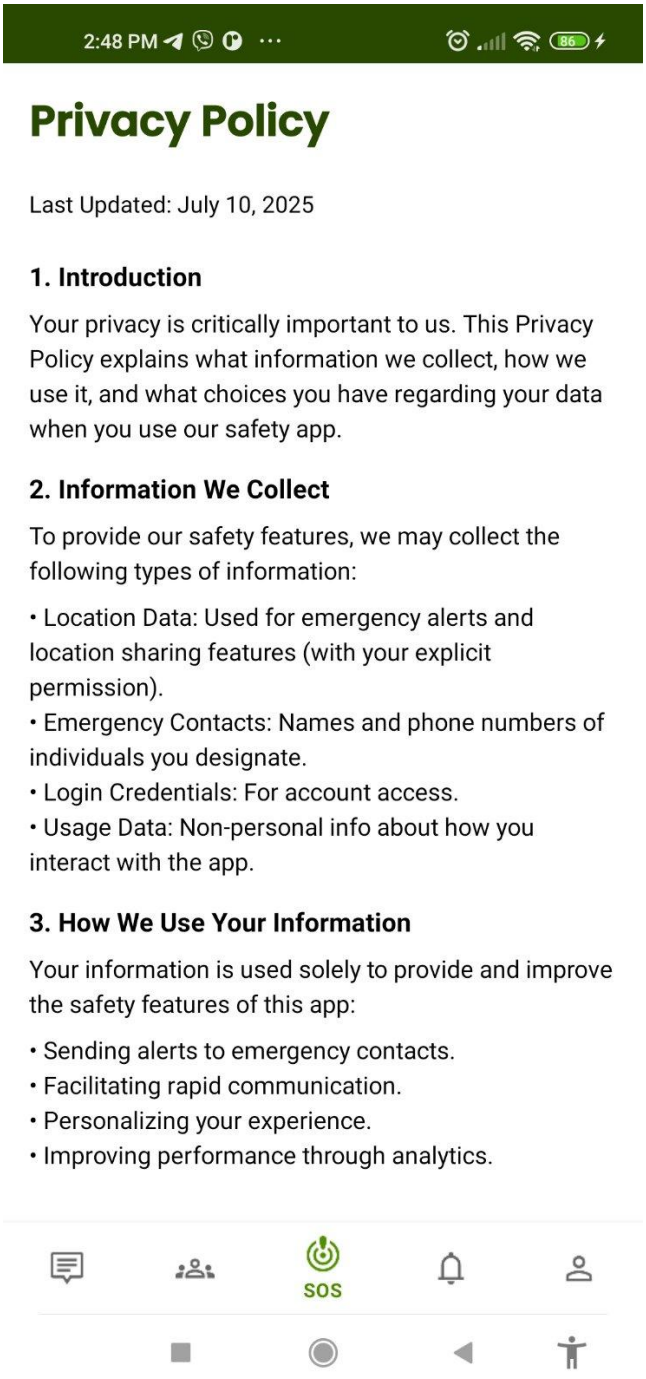

Figure 26 : Privacy Policy

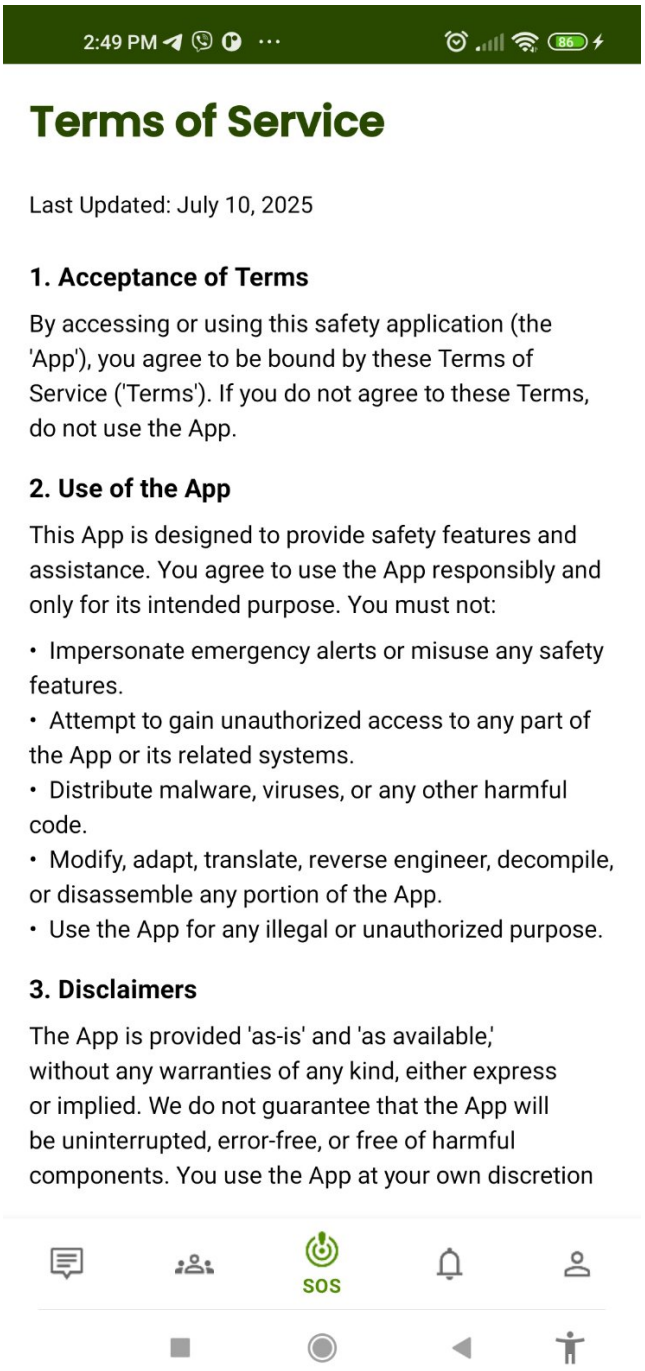

Figure 27 : Terms of Service Screen

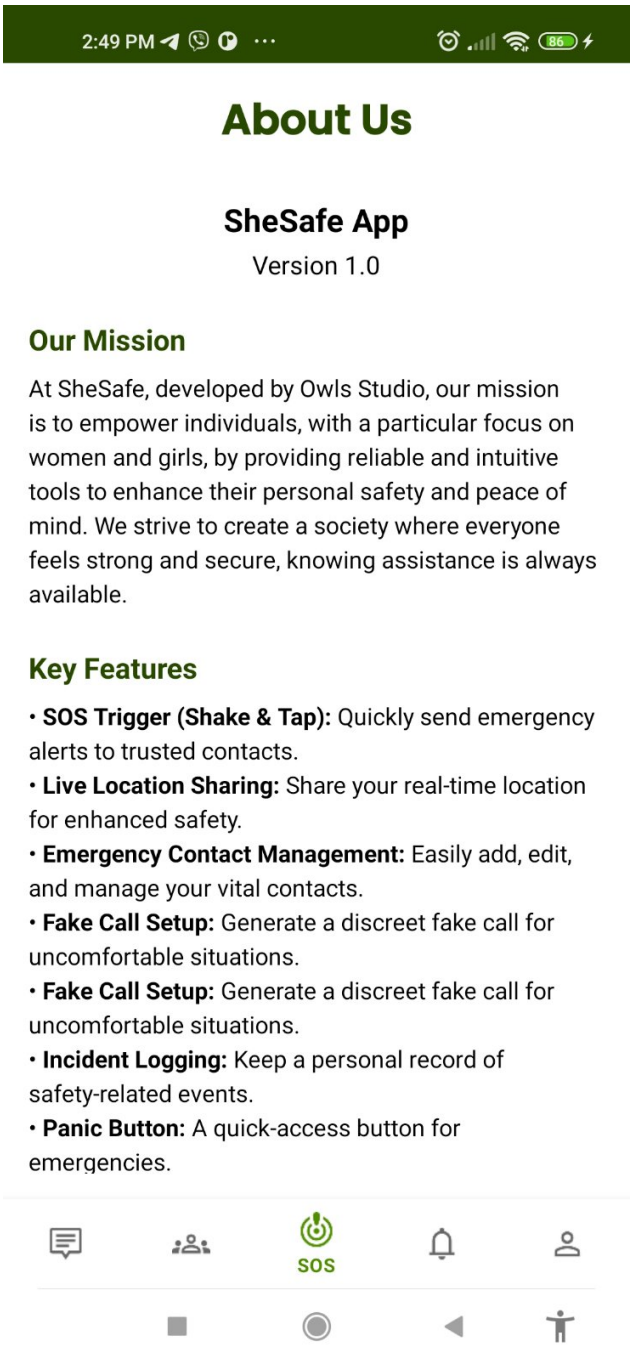

Figure 28 : About Us

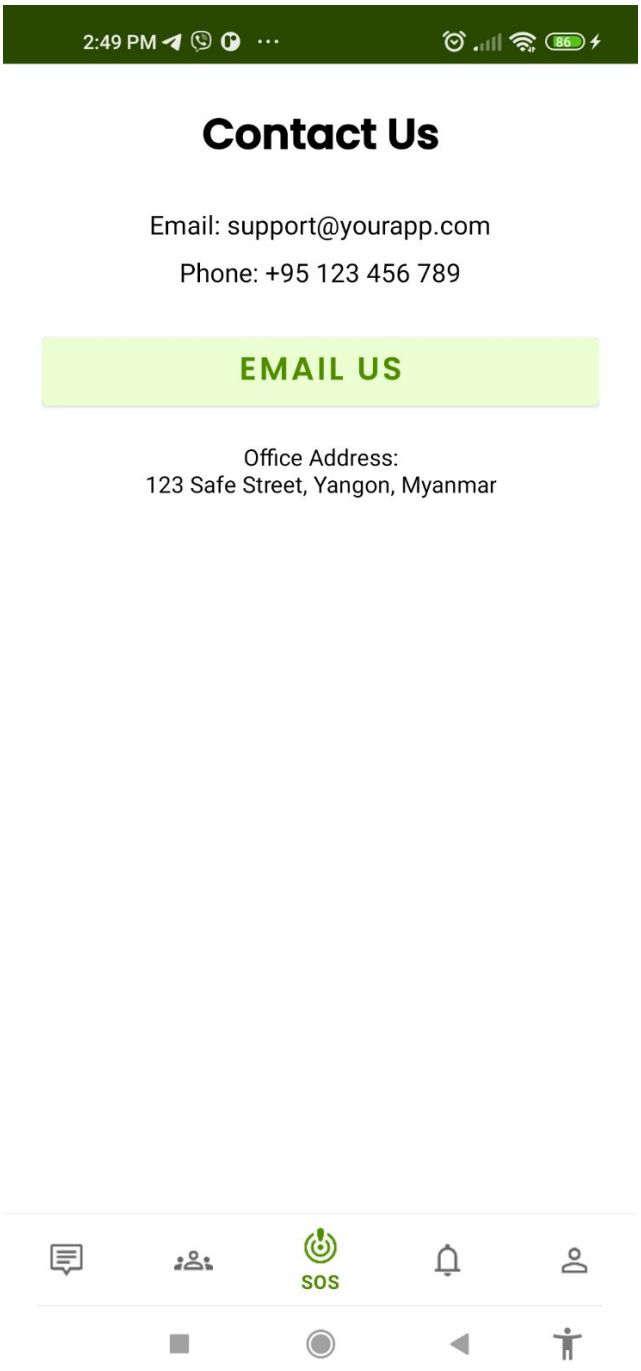

Figure 29 : Contact Us Screen

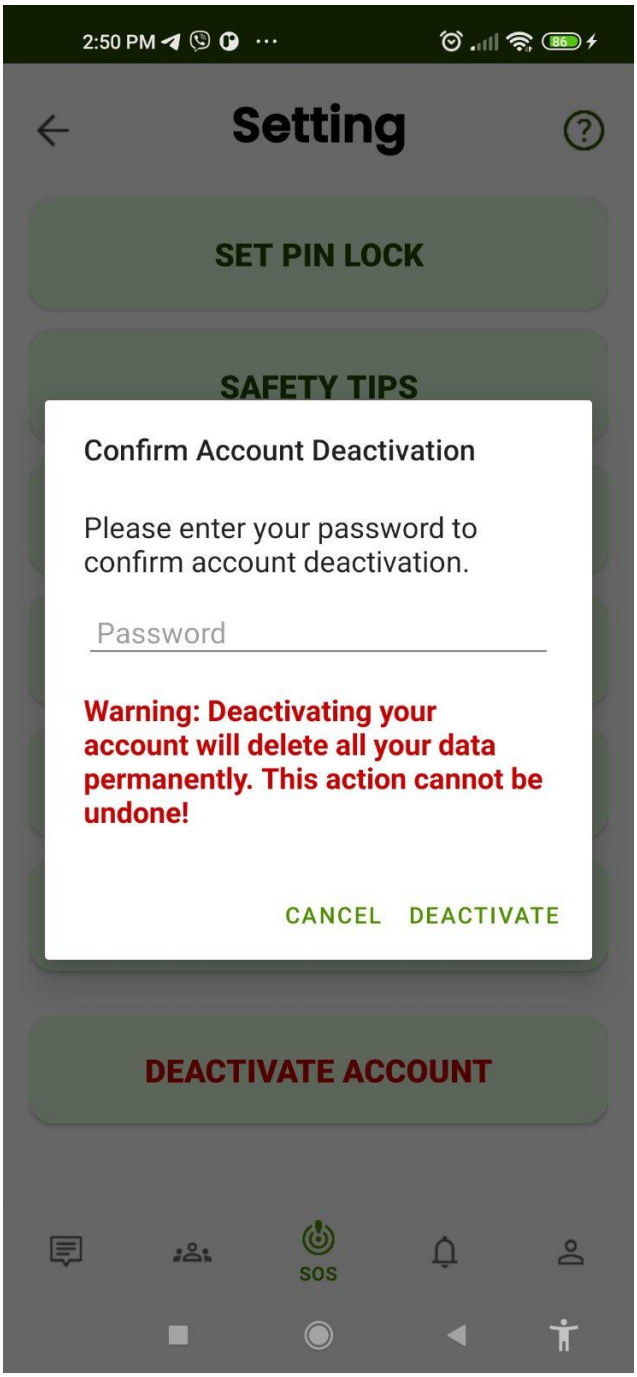

Figure 30 : Deactivate Account Screen

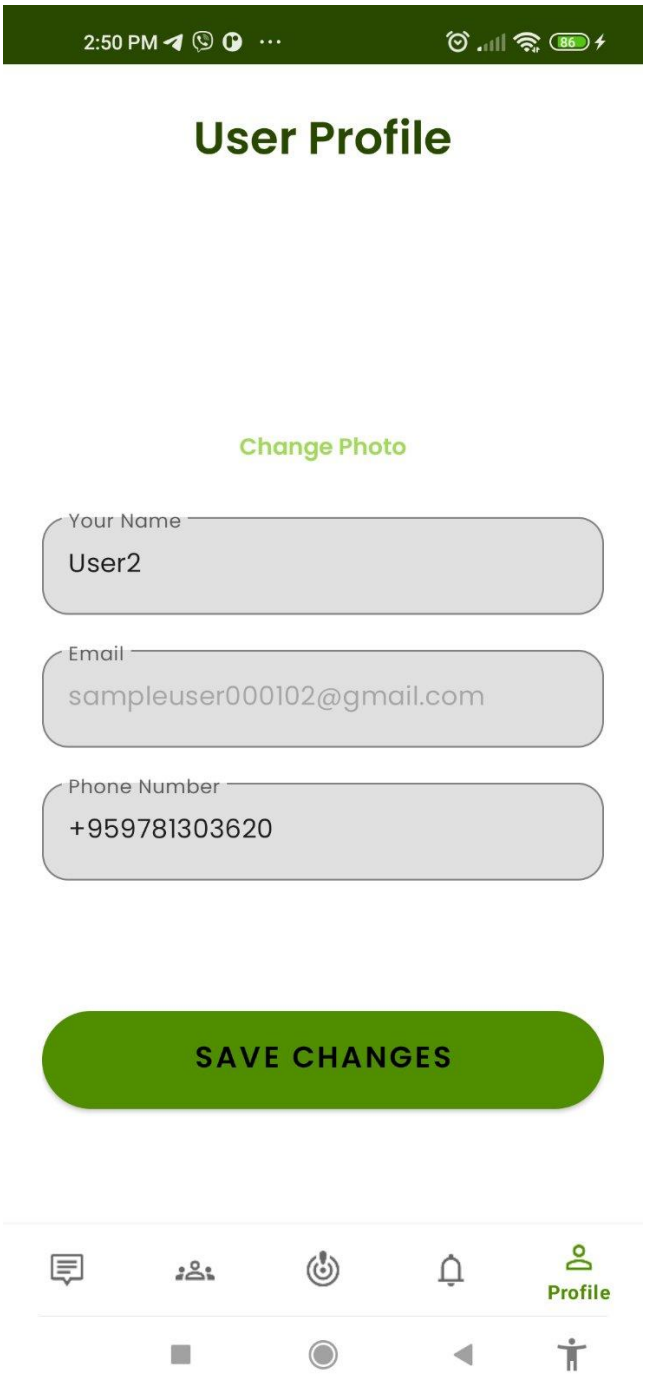

Figure 31 : User Profile Screen 1

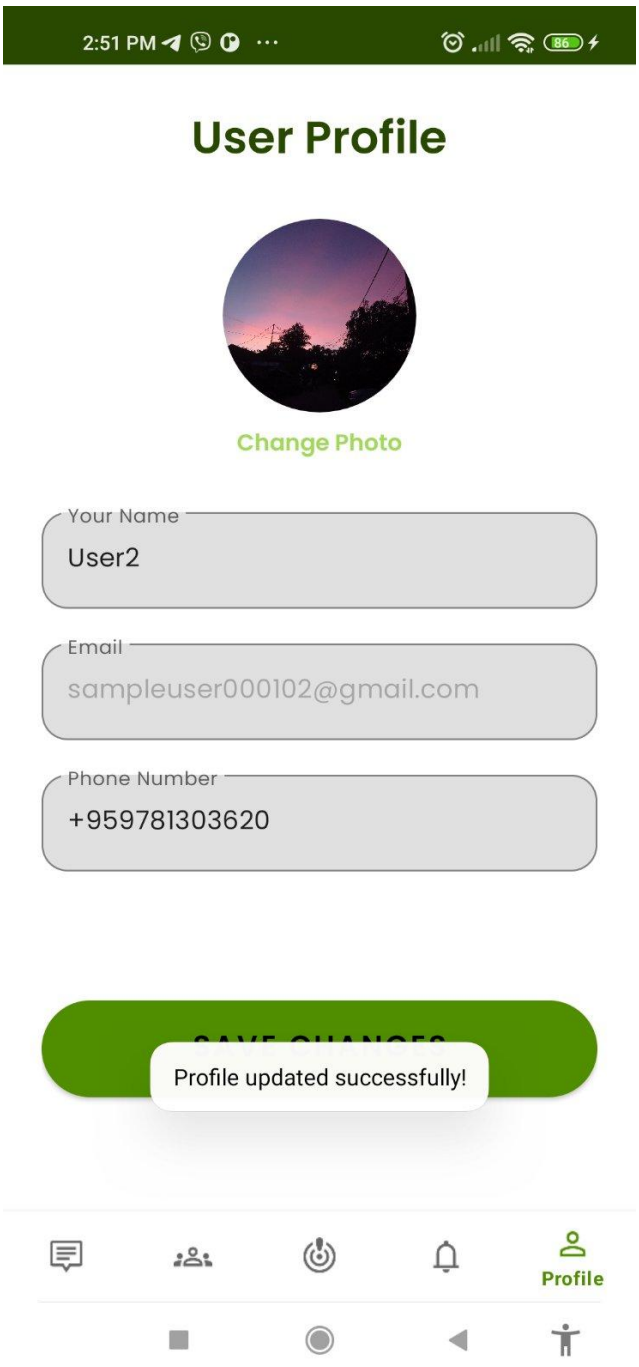

Figure 32 : User Profile Screen 2

1.5. Security Features of the system

1. User Authentication & Verification
    - Firebase Authentication for secure sign-up, login, and password reset.
    - Email verification enforcement before granting access.
    - Client-side SHA-256 password hashing stored securely in SQLite.
    - Account deactivation with re-authentication to prevent unauthorized actions.
2. Data Protection
    - Secure storage of sensitive data with hashed passwords (never plain text).
    - Controlled access to user and emergency contact data in encrypted/local SQLite databases.
    - Fine-grained security rules in Firebase Firestore for location sharing.
3. Encryption
    - SHA-256 for password confidentiality.
    - Secure data transmission over SSL/TLS with Firebase services.
    - Storage of audio recordings and personal data in app-private directories inaccessible to other apps.
4. Input Validation and Sanitization
    - Regex and checks to ensure proper email format, password strength, and valid phone numbers.
    - Prevention of duplicate contacts and invalid inputs (e.g., empty messages, excessive contacts).
5. Protection Against Common Attacks
    - Passwords never stored or transmitted in plain text.
    - Re-authentication for critical operations to prevent session hijacking.
    - Runtime permissions enforce user consent and prevent unauthorized access to sensitive device features.
    - Limited retry to prevent abuse in location fetch to avoid denial-of-service.
6. Logging and Auditing
    - Detailed logging of all emergency messages sent, including timestamps, recipients, message content, and location.
    - Local incident logs stored and accessible for user review and debugging.
7. Permission Management and Secure Communication
    - Explicit runtime permission requests and graceful fallback on denials for SMS, location, microphone, and contacts.
    - All communications with Firebase secured over HTTPS using user-specific paths and merge/update to protect privacy.

Section D

1. Testing

1.1. User Authentication

- **Test Name :** User Authentication
- **Test Data :**
  - Register a new user with valid details via SignupActivity.
  - Attempt to log in before email verification—should prompt verification.
  - Verify email via received mail and log in again.
  - Use "Forgot Password" to send reset email.
  - Deactivate the account via SettingFragment with correct password confirmation.

- **Expected Result:**
  - Registration completes with verification email sent.
  - Login blocked until email verified.
  - Password reset email received and usable.
  - Account deactivation signs user out and removes access.

- **Actual Result:** Figure D.1.1.1 – (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)

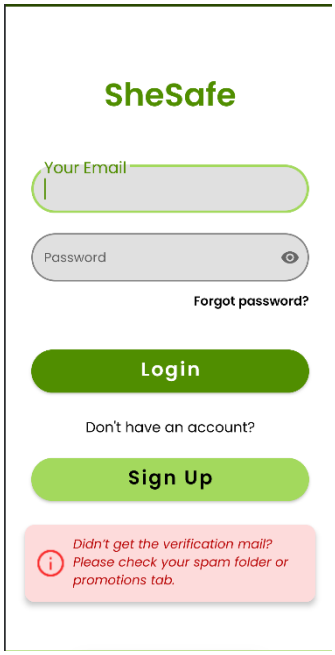

Figure D.1.1.1 – 1

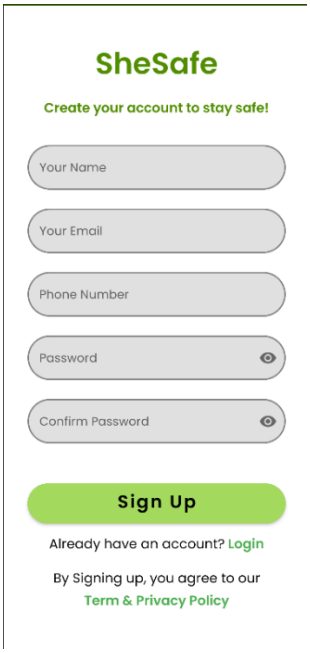

Figure D.1.1.1 – 2

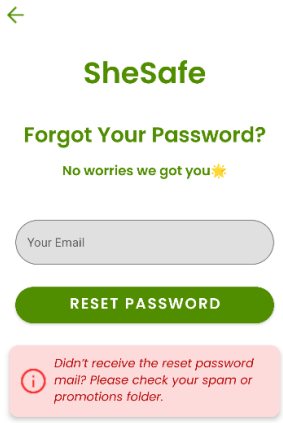

Figure D.1.1.1 – 3

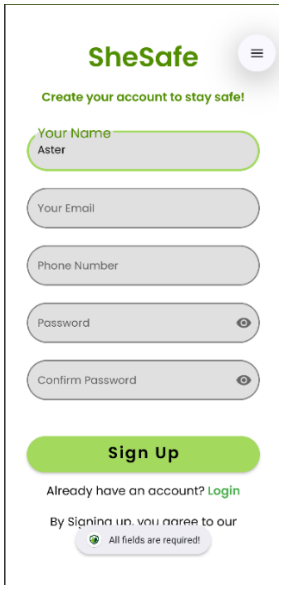

Figure D.1.1.1 – 4

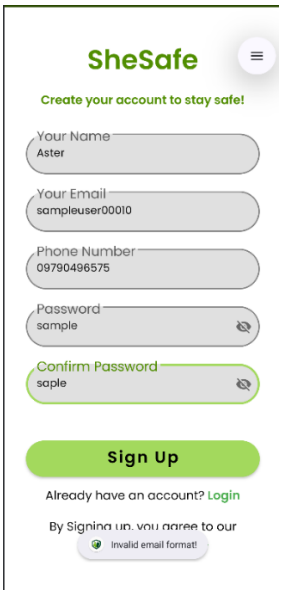

Figure D.1.1.1 – 5

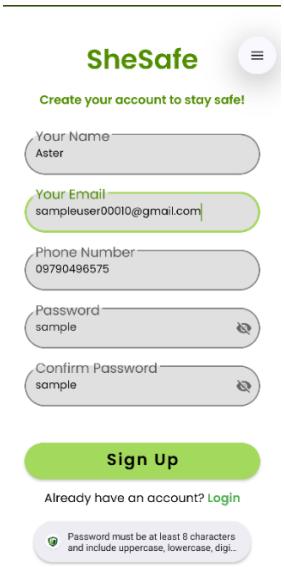

Figure D.1.1.1 – 6

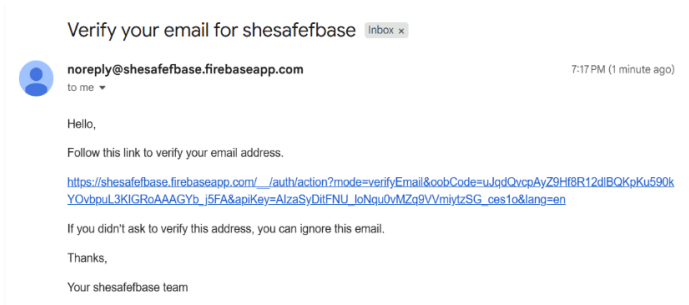

Figure D.1.1.1 – 7

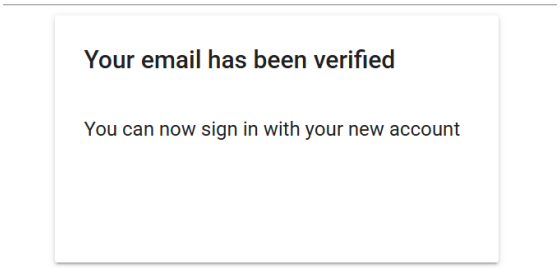

Figure D.1.1.1 – 8

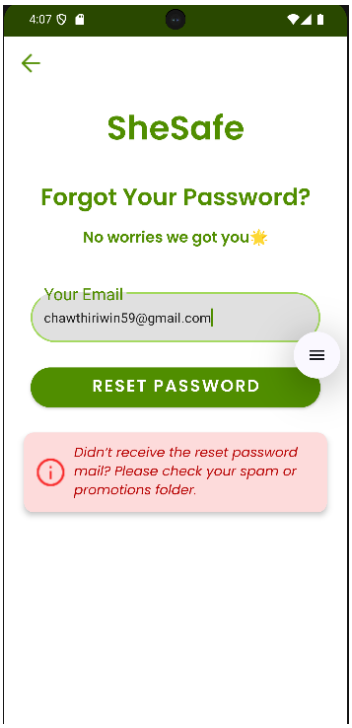

Figure D.1.1.1 – 9

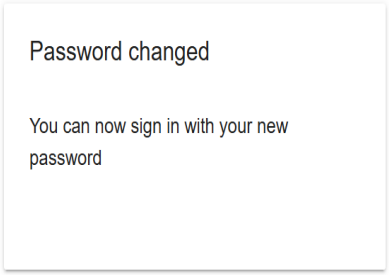

Figure D.1.1.1 – 10

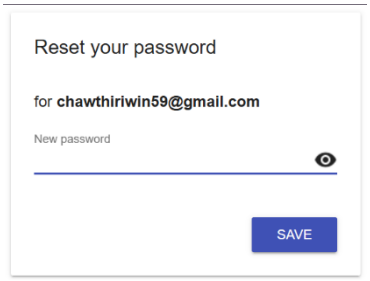

Figure D.1.1.1 - 11

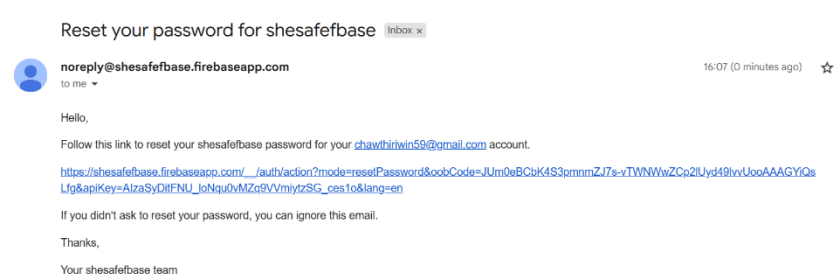

Figure D.1.1.1 – 12

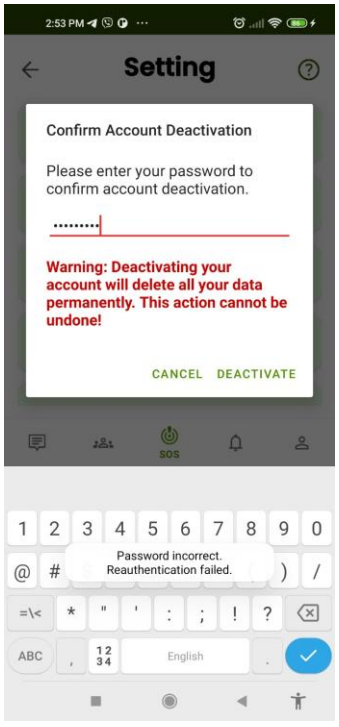

Figure D.1.1.1 – 13

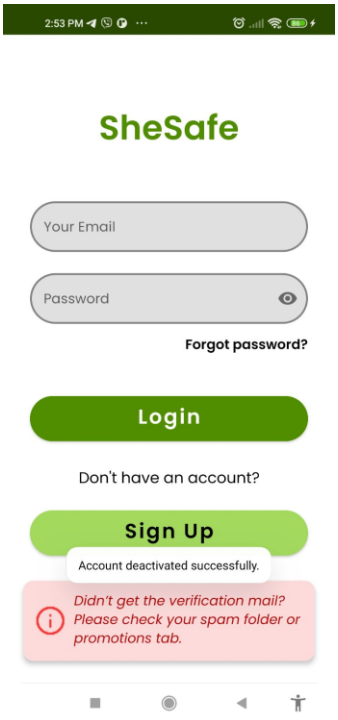

Figure D.1.1.1 – 14

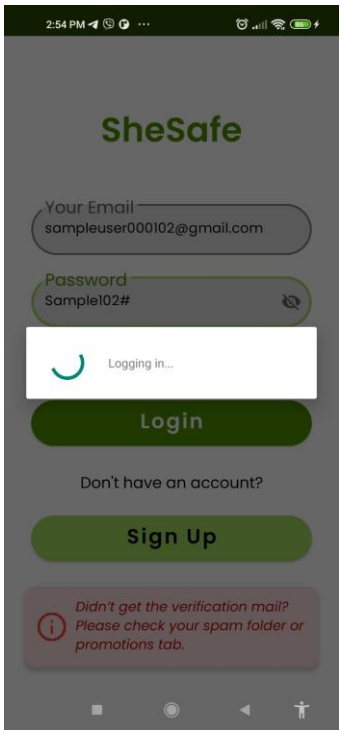

Figure D.1.1.1 – 15

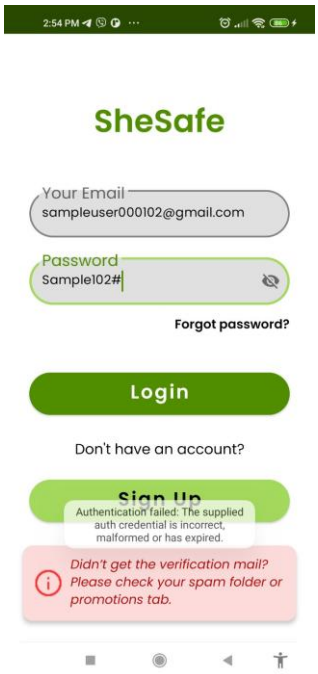

Figure D.1.1.1 – 16

1.2. PIN Lock Feature

- **Test Name :** PIN Lock Feature
- **Test Data :**
    - Set a 4-digit PIN in SetPinLockFragment.
    - Close and reopen the app to trigger PIN unlock screen.
    - Unlock with correct and incorrect PIN codes.
- **Expected Result:**
    - App prompts for PIN on startup/resume.
    - Correct PIN grants access; incorrect PIN shows error and blocks access.
- **Actual Result:** Figure D.1.1.2 – (1,2,3,4,5,6,7,8)

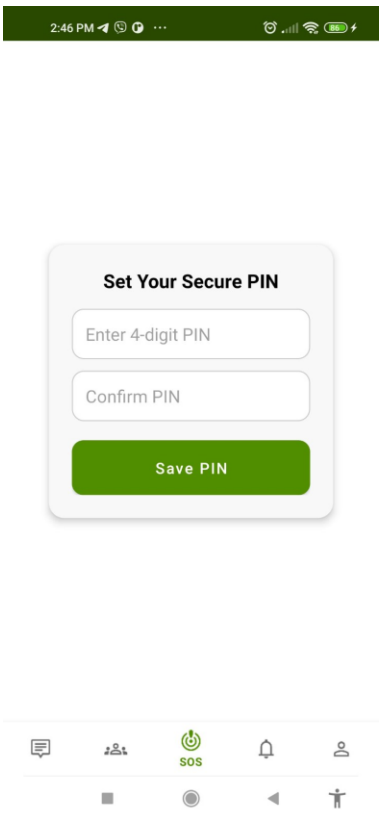

Figure D.1.1.2 – 1

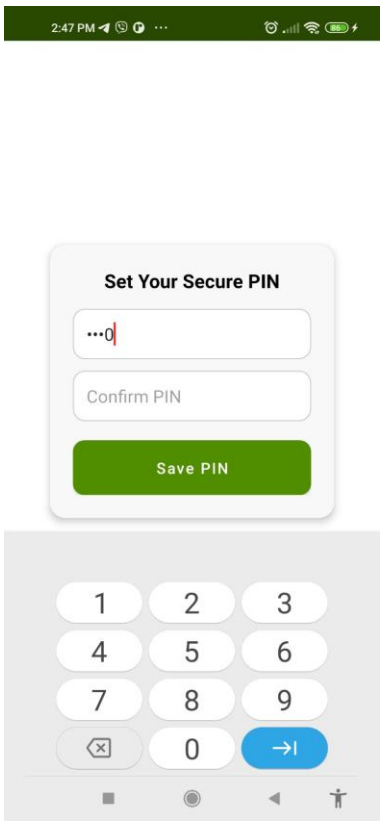

Figure D.1.1.2 – 2

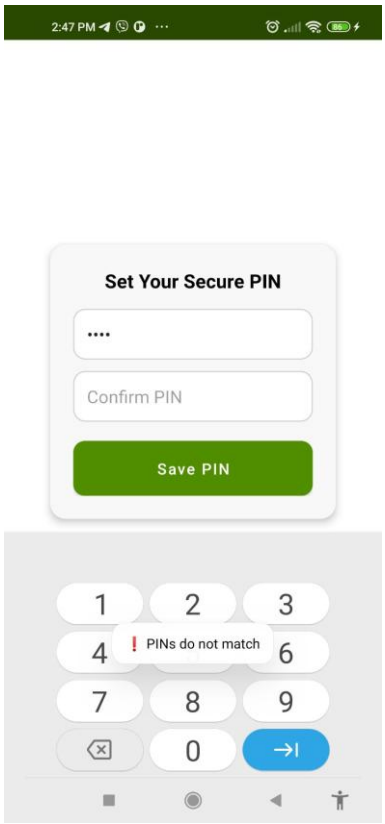

Figure D.1.1.2 – 3

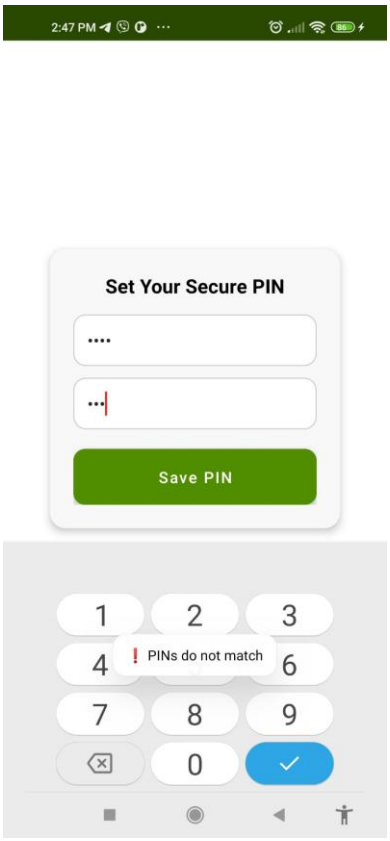

Figure D.1.1.2 – 4

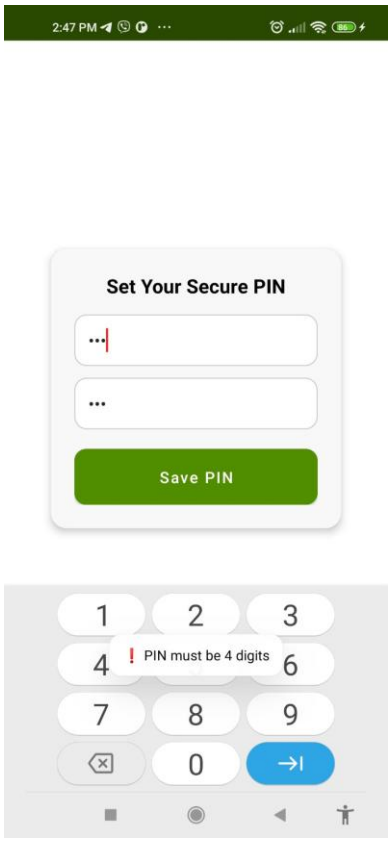

Figure D.1.1.2 – 5

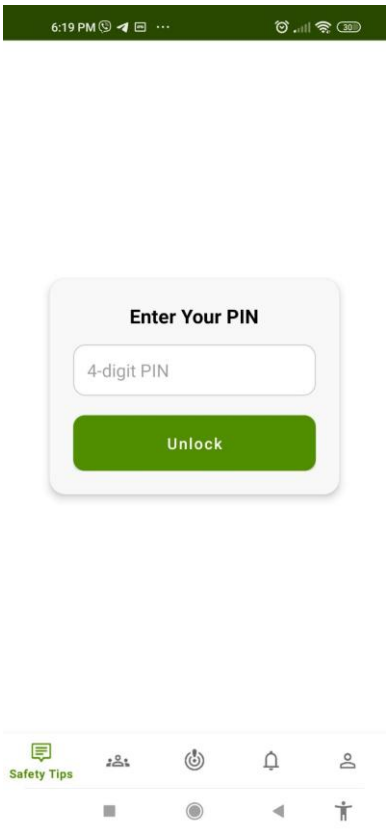

Figure D.1.1.2 – 6

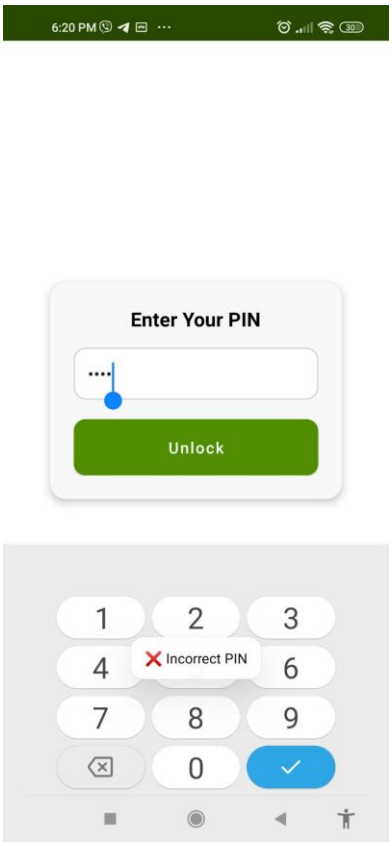

Figure D.1.1.2 – 7

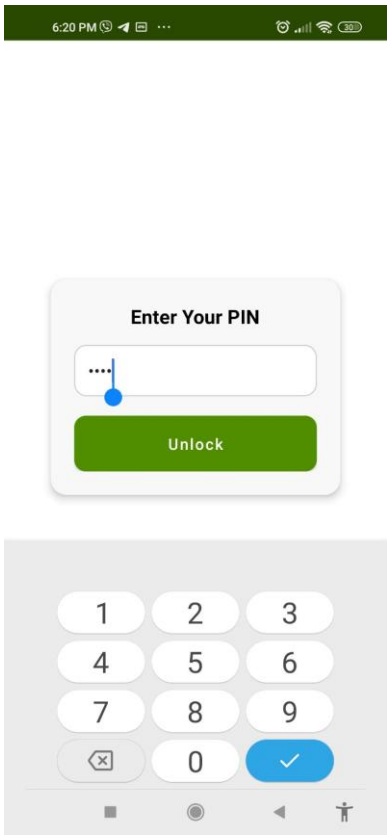

Figure D.1.1.2 – 8

1.3. Emergency Messaging (SOS & Alert)

- **Test Name :** Emergency Messaging
- **Test Data :**
    - Send SOS via one-tap button and shake gesture.
    - Compose and send Alert messages to selected contacts in EmergencyMessageFragment.
    - Confirm SMS permissions requested and SMS fallback to SMS app works if direct send fails.
- **Expected Result:**
    - SMS sent to correct contacts with message and location.
    - Fallback SMS app opens if needed.
    - User receives confirmation notifications.
- **Actual Result:** Figure D.1.1.3 – (1,2,3,4,5,6)

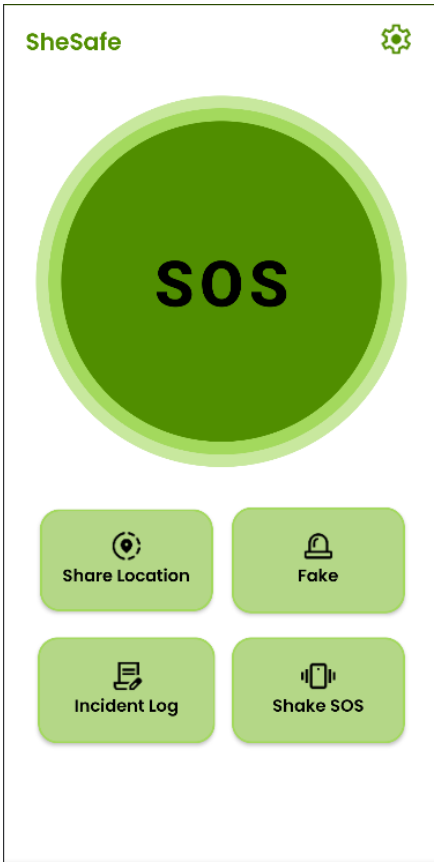

Figure D.1.1.3 – 1

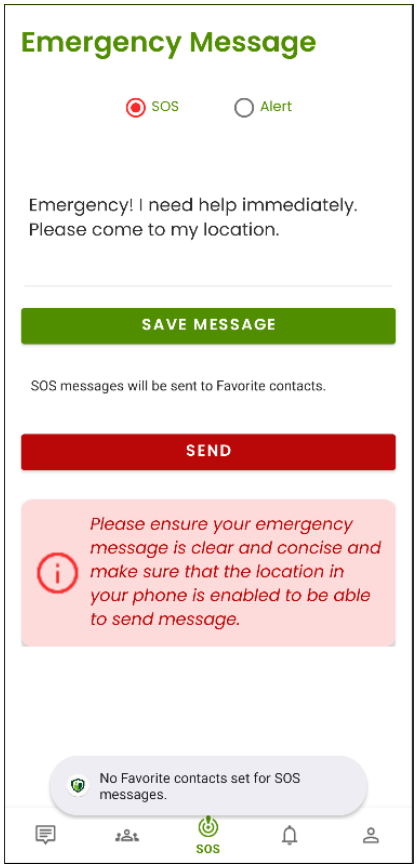

Figure D.1.1.3 – 2

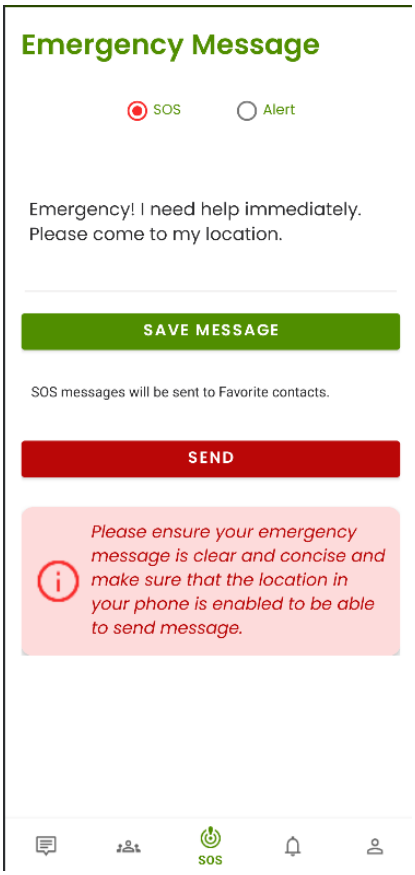

Figure D.1.1.3 – 3

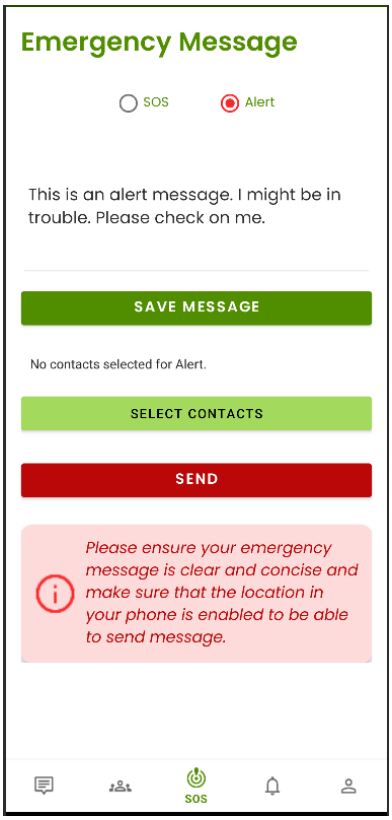

Figure D.1.1.3 – 1

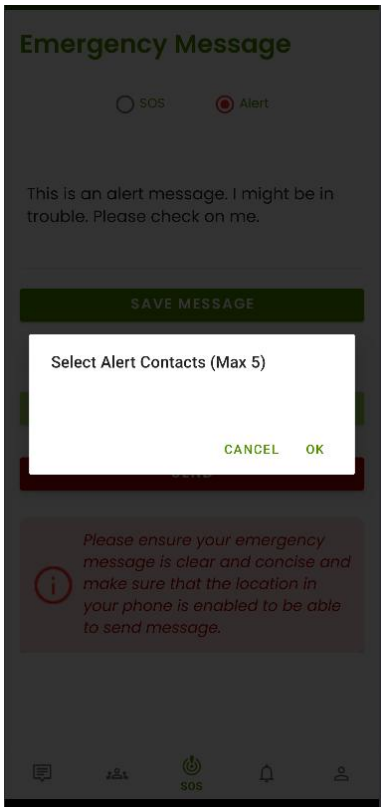

Figure D.1.1.3 – 2

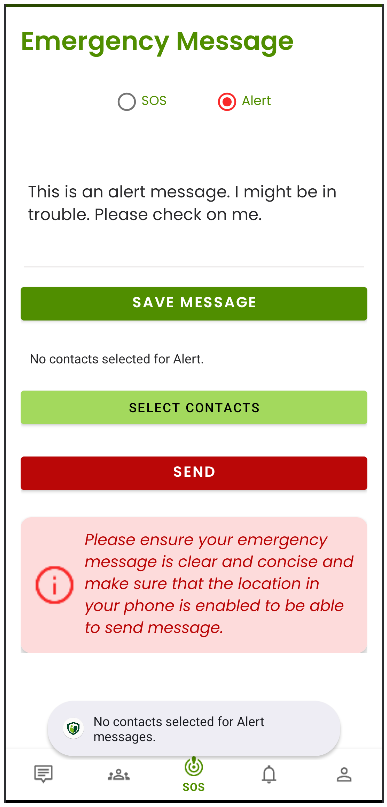

Figure D.1.1.3 – 3

1.4. Contact Management

- **Test Name :** Contact Management
- **Test Data :**
    - Add emergency contacts by choosing from device contacts.
    - Attempt adding duplicates to verify prevention.
    - Edit contacts to mark as favorites or toggle alert flags.
    - Delete contacts and confirm UI updates.
- **Expected Result:**
    - Contacts display correctly with proper flags.
    - Duplicates prevented with error message.
    - Contact list updates immediately after edits/deletions.
- **Actual Result:** Figure D.1.1.4 – (1,2)

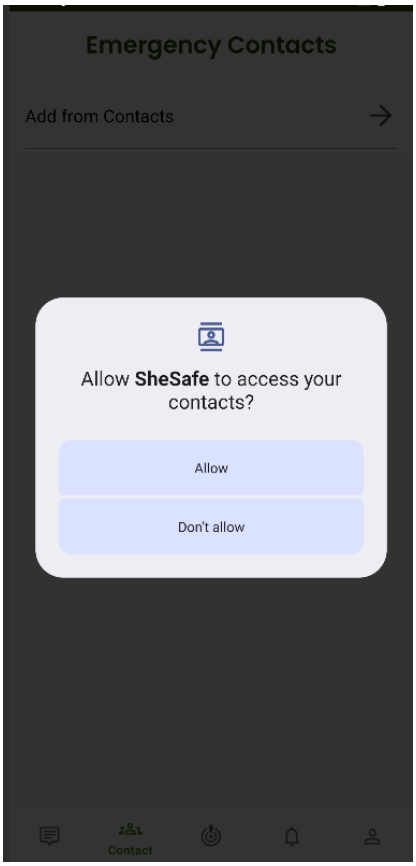

Figure D.1.1.4 – 1

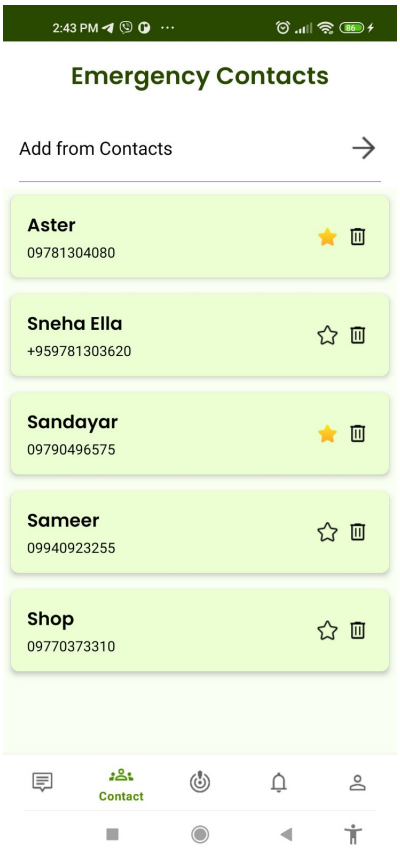

Figure D.1.1.4 – 2

1.5. Panic Mode with Audio Recording

- **Test Name : Panic Mode with Audio Recording**
- **Test Data :**
    - Start Panic Mode and observe countdown timer.
    - Grant microphone permission if requested.
    - Confirm audio is recording after countdown and stops on button press.
    - Playback recorded audios and delete from RecordingHistoryFragment.
    - Confirm SOS message sent after countdown.
- **Expected Result:**
    - Countdown visible and accurate.
    - Audio recording saved properly and playable.
    - SOS message sent with correct info and location.
    - Audio deletions reflected immediately.
- **Actual Result:** Figure D.1.1.5 – (1,2,3,4,5,6)

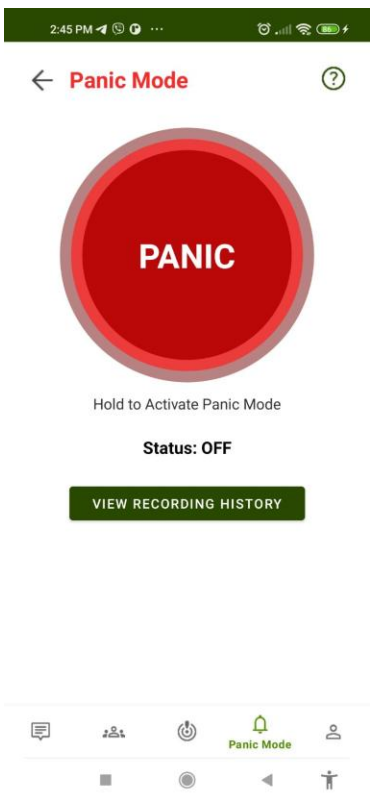

Figure D.1.1.5 – 1

Figure D.1.1.5 – 2

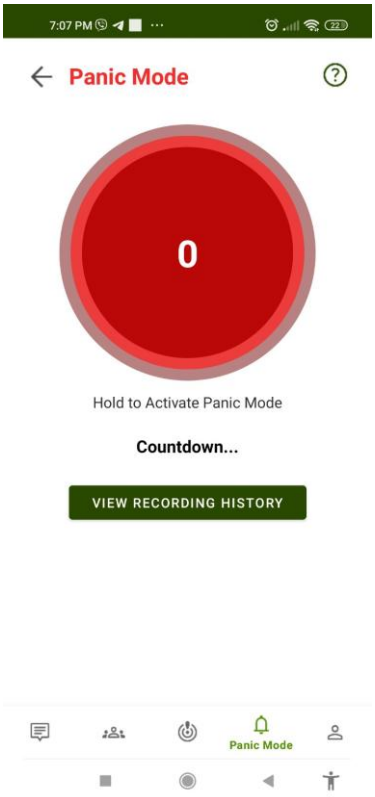

Figure D.1.1.5 – 3

Figure D.1.1.5 – 4

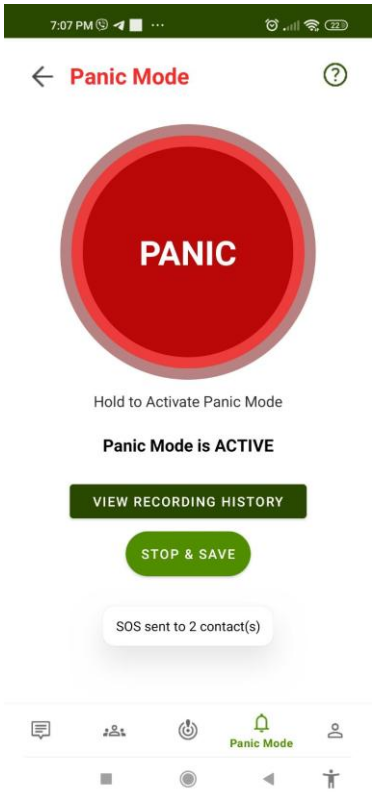

Figure D.1.1.5 – 5

Figure D.1.1.5 – 6

1.6. Fake Call Simulation

- **Test Name : Fake Call Simulation**
- **Test Data :**
    - Launch fake call from FakeHomeFragment.
    - Select mode: ringtone only, silent screen, or voice playback.
    - Receive fake call UI and hear ringtone/voice as applicable.
    - Answer or decline fake call.
- **Expected Result:**
    - Fake call appears realistically with audio playing.
    - Answering or declining behaves as expected.
    - Audio stops and UI closes appropriately.
- **Actual Result:** Figure D.1.1.6 – (1,2,3,4,5,6)

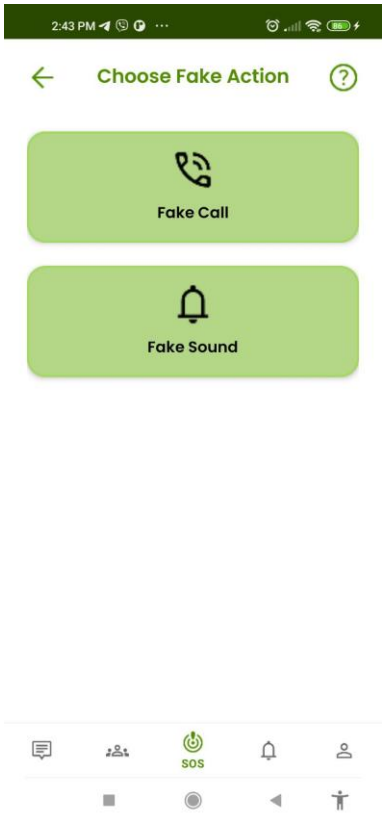

Figure D.1.1.6 – 1

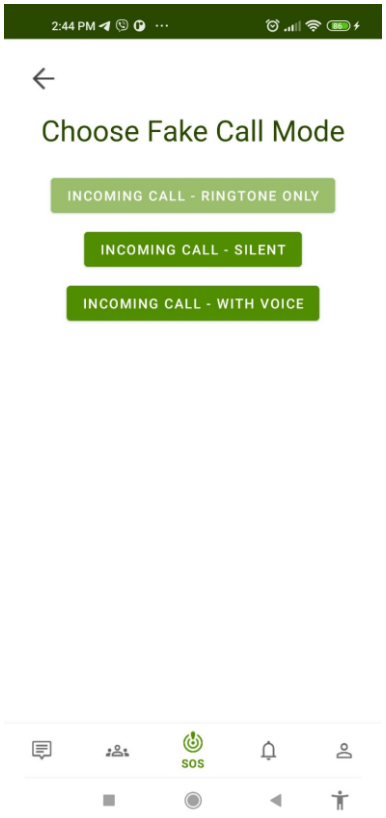

Figure D.1.1.6 – 2

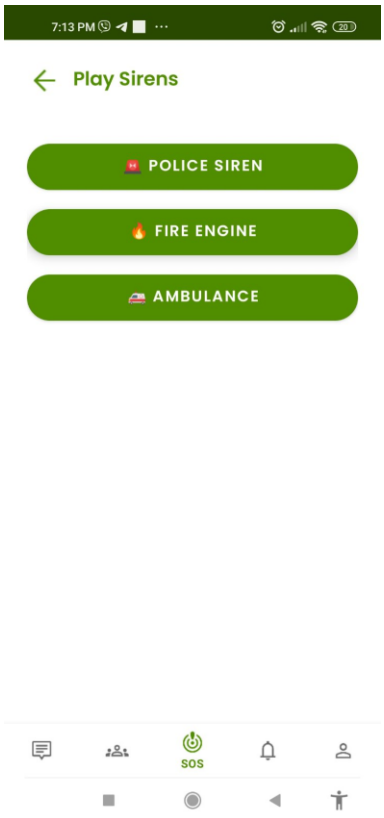

Figure D.1.1.6 – 3

1.7. Incident Log Viewing

- **Test Name : Incident Log Viewing**
- **Test Data :**
  - Open IncidentLogFragment.
  - View emergency message history with timestamps, message types, and recipients.
  - Use filters to switch between All, SOS only, and Alert only views.
- **Expected Result:**
  - Incident logs show accurate filtered data instantly.
  - This is the Logs of SOS and Alert from Emergency Message.
- **Actual Result:** Figure D.1.1.7 – (1,2,3)

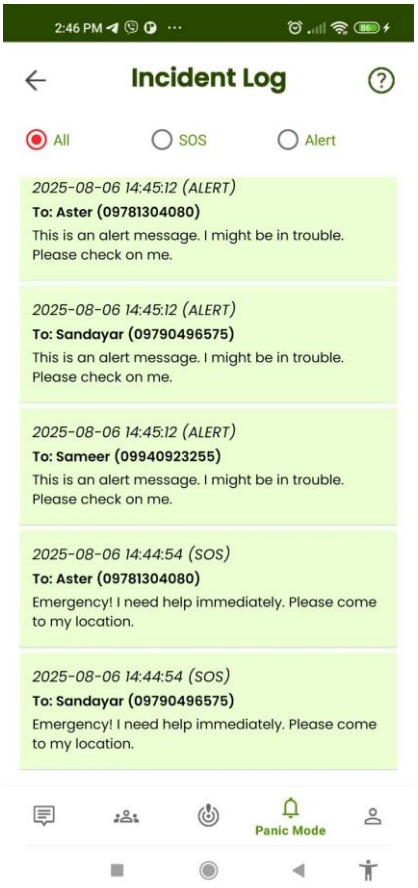

Figure D.1.1.7 – 1

Figure D.1.1.7 – 2

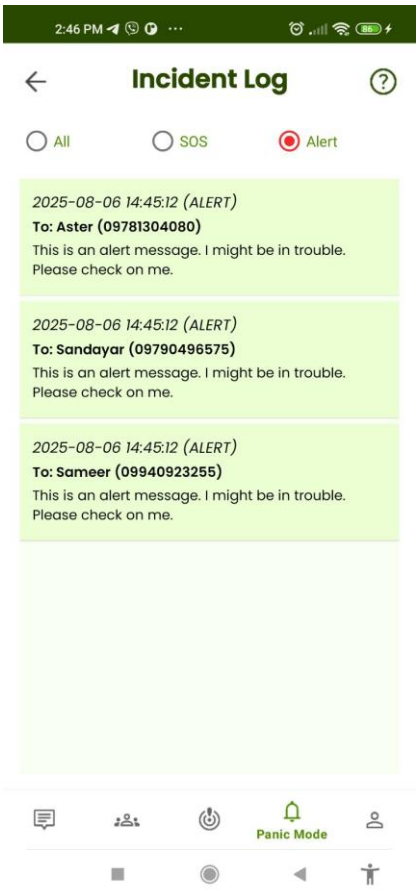

Figure D.1.1.7 – 3

1.8. Shake SOS Functionality

- **Test Name : Shake SOS Functionality**
- **Test Data :**
    - Shake the device to trigger SOS alert.
    - Observe countdown and permission request if applicable.
- **Expected Result:**
    - SOS message sent to favorited contacts with location link.
    - User need to wait during countdown and it will automatically send message.
- **Actual Result:** Figure D.1.1.8 – (1,2)

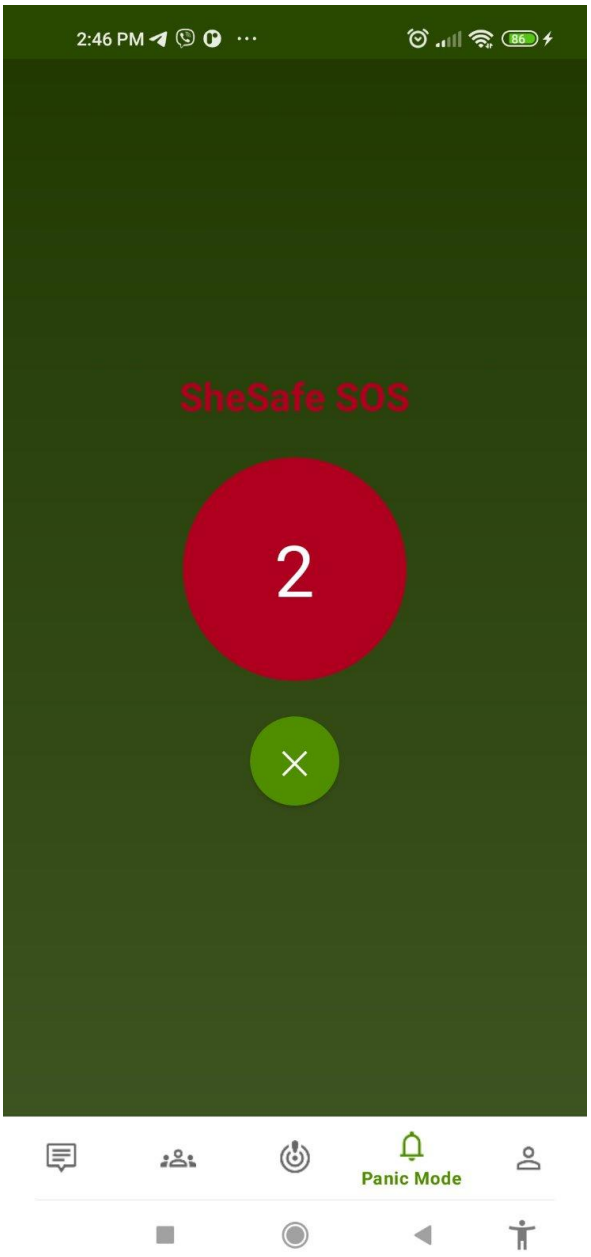

Figure D.1.1.8 – 1

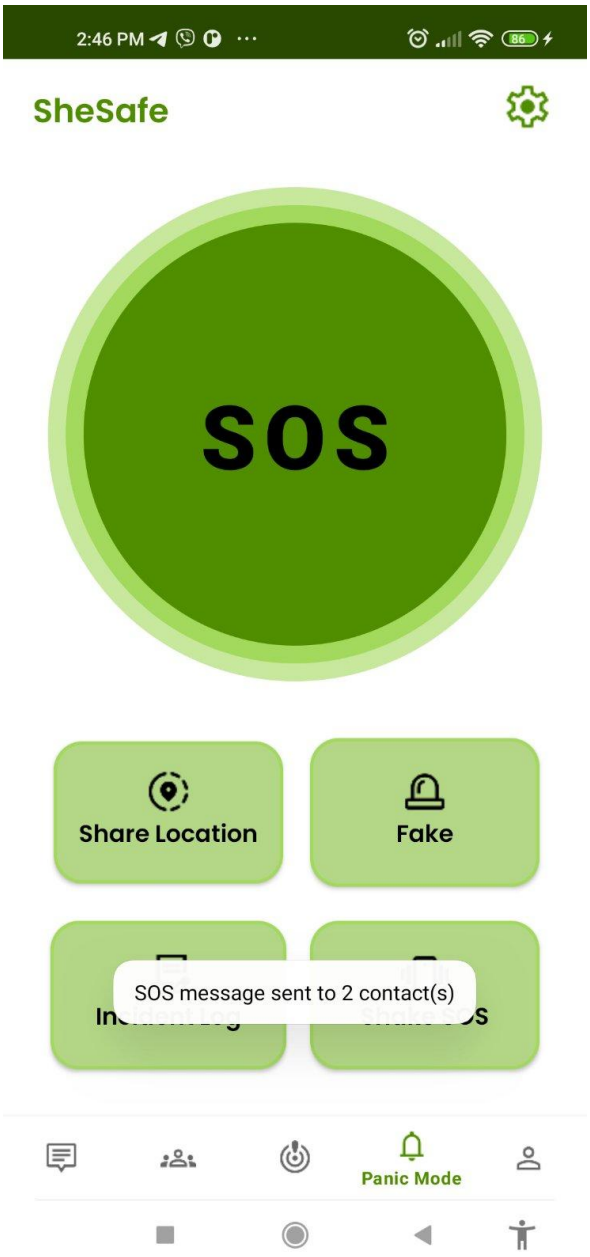

Figure D.1.1.8 – 2

1.9. Runtime Permissions Handling

- **Test Name : Runtime Permissions Handling**
- **Test Data :**
    - Deny SMS, Contacts, Audio, or Storage permissions when requested.
    - Observe app behavior (error message or fallback).
    - Later grant permissions and verify app regains full functionality.
- **Expected Result:**

- App handles permissions denied gracefully without crashing.
- After granting, features dependent on permissions work correctly.

- **Actual Result:** Figure D.1.1.9 – (1,2,3)

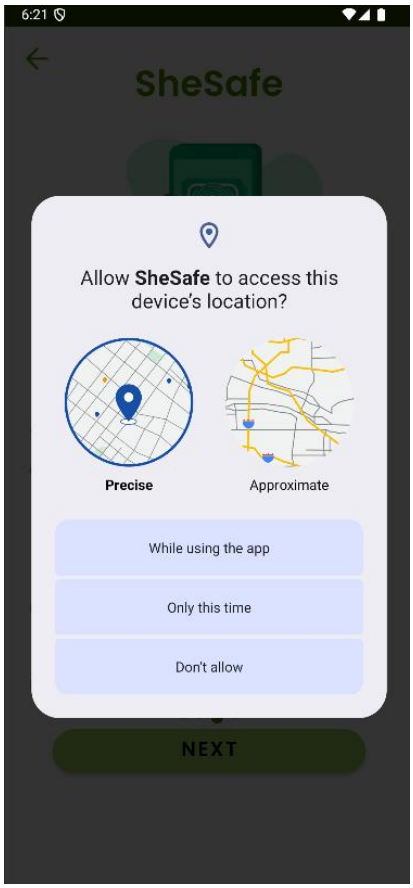

Figure D.1.1.9 – 1

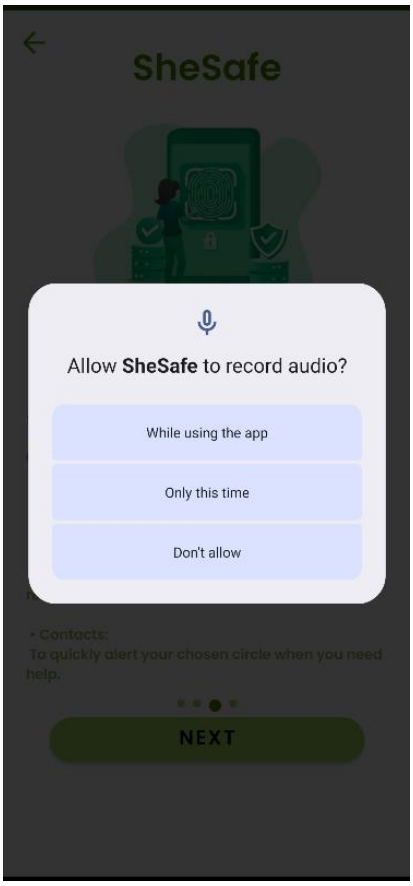

Figure D.1.1.9– 2

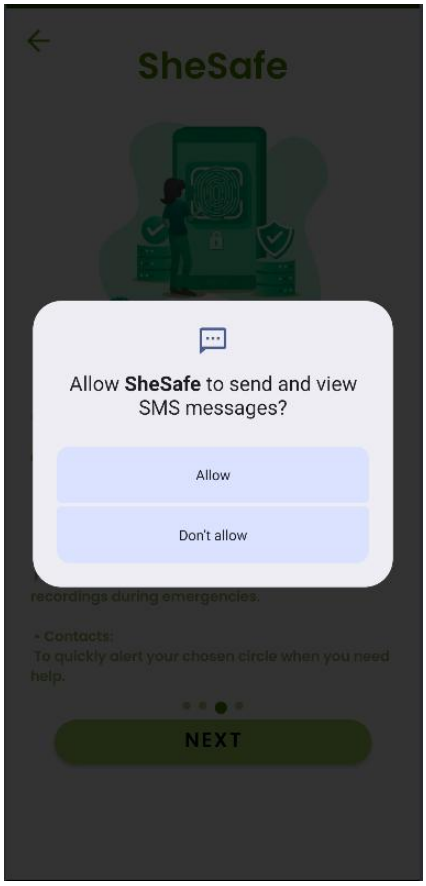

Figure D.1.1.9 – 3

1.10. UI Navigation and Confirmation Dialogs

- **Test Name : UI Navigation and Confirmation Dialogs**
- **Test Data :**
  - Use bottom navigation bar to switch between main fragments.
  - Perform actions requiring confirmation (e.g., sending messages, deleting contacts, account deactivation).
  - Accept and cancel confirmations.
- **Expected Result:**
  - Navigation is smooth and without errors.
  - Confirmation dialogs appear where appropriate.
  - Canceling stops action; confirming proceeds correctly.
- **Actual Result:** Figure D.1.1.10 – (1,2,3,4,5,6)

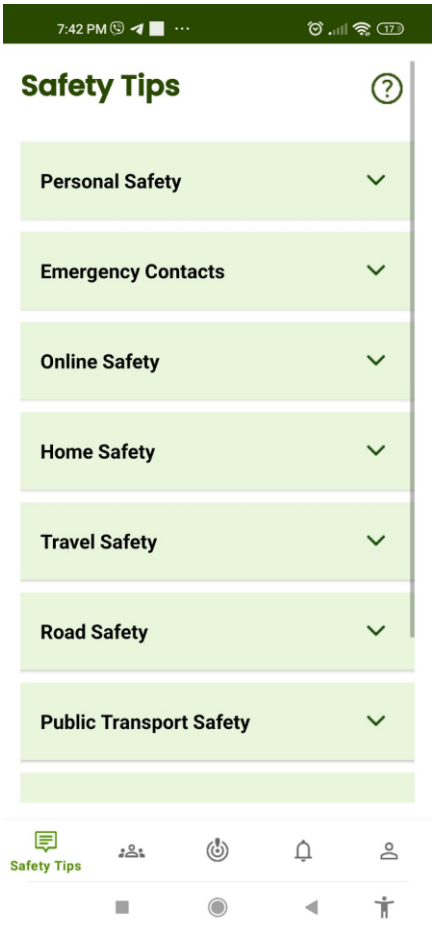

Figure D.1.1.10 – 1

Figure D.1.1.10 – 2

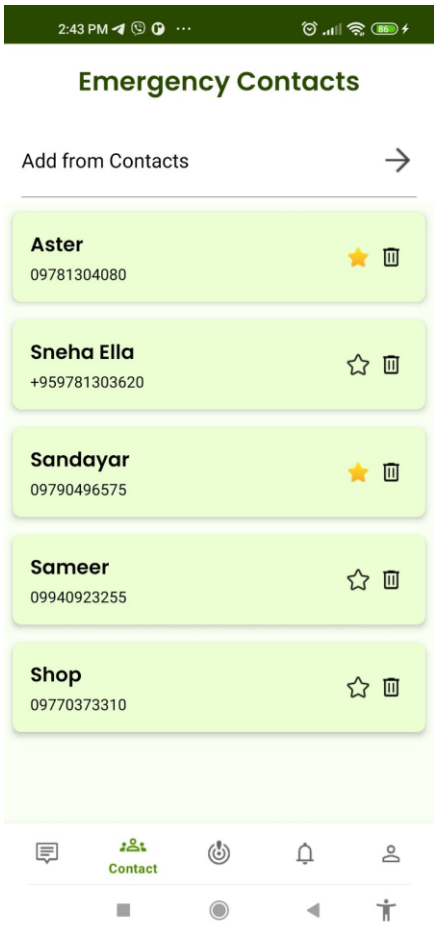

Figure D.1.1.10 – 3

Figure D.1.1.10 – 4

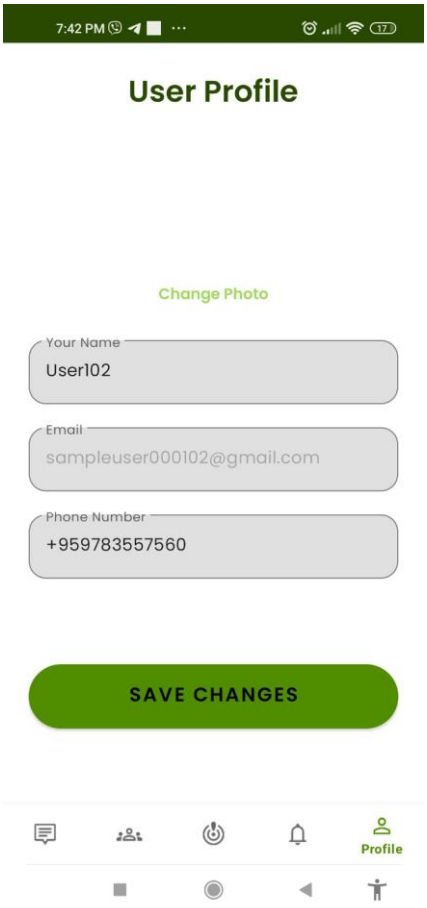

Figure D.1.1.10 – 5

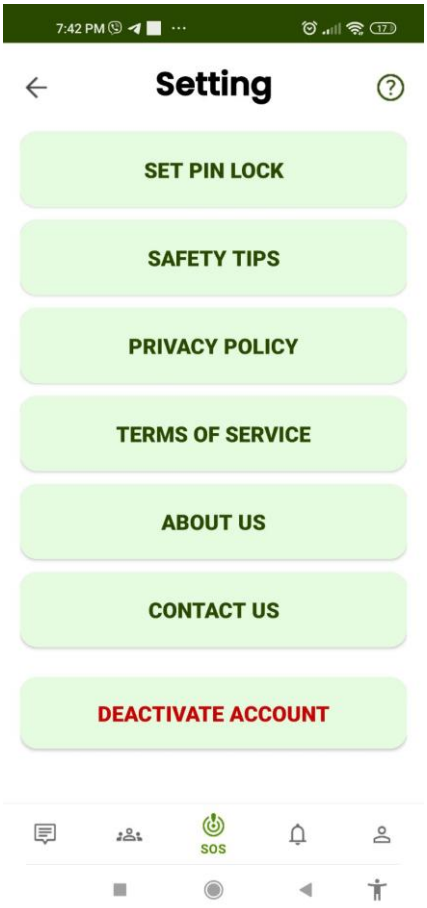

Figure D.1.1.10 – 6

2. System Implementation

2.1. Implementation Plan

Overview

This SheSafe app is developed from scratch, targeting Android devices. The project have used modern Android development practices, a modular architecture, and a combination of local and cloud-based data storage to create a reliable and scalable safety application.

1. Conversion Procedures

- **New System Build**: The app is developed using Android Studio with Java, leveraging the Android SDK. The architecture includes modular components like fragments and activities for different features, such as authentication, messaging, and contacts.
- **Data Input**: User profile information, emergency contacts, and messages will be collected through the app's UI and stored using local databases and Firebase.
- **Initialization**: On the first run, the app will create necessary local databases (DatabaseHelper1 and DatabaseHelper2) and populate them with default data, such as messages and user roles, as defined by the database schema.

2. Data Migration

For this initial version, the primary focus is setting up new data storage rather than migrating from an old system.

- **SQLite Database Setup**: Local storage will be implemented to manage user data, including user accounts, contacts, and incident logs. The database helper classes will handle table creation and data insertion upon the app's first launch.
- **Firebase Synchronization**: The app use Firebase for features requiring real-time updates and cloud storage. This includes:
    - **User Authentication**: Firebase Authentication manage user sign-up, login, and password reset functionalities. Email verification is a required step for a user to proceed to the main app.
    - **Live Location Sharing**: Firestore is used to handle and listen for real-time location updates between users. This feature is critical for the app's safety functionalities, such as allowing users to share their live location with their contacts.
    - **User Registration**: The user's phone number and unique Firebase ID (UID) will be registered and linked in the Firestore database during the sign-up process.

3. Development & Integration

The development phase is modular, focusing on individual features that are then integrated to form the complete application.

- **Modular Implementation**: The app is structured with multiple activities for authentication and onboarding (SignupActivity, LoginActivity, WelcomeActivity) and fragments for core functionalities (HomeFragment, IncidentLogFragment, LiveLocationMapFragment, EmergencyMessageFragment).
- **Third-Party Libraries**: The app will integrate the following libraries and services:
    - **Firebase**: For authentication, real-time location sharing via Firestore, and potentially analytics and crash reporting.
    - **OSMdroid**: An open-source mapping library for displaying live location on a map.
    - **Android APIs**: Native Android APIs will be used for managing permissions, recording audio, and handling media playback for panic mode features.
- **Permission Management**: The app handle runtime permissions for critical features like location, SMS, contacts, and audio recording. Permissions will be requested proactively to ensure a smooth user experience.
- **UI Components**: The app use RecyclerView and custom adapters to display dynamic lists of data, such as a user's contacts and incident logs.

4. Deployment Setup

Deployment to the Google Play Store will follow standard procedures to ensure a successful launch.

- **Build Configuration**: The app is built using Gradle in Android Studio, targeting a minimum API level of 23 for proper runtime permission handling.
- **App Signing**: A release keystore will be generated to sign the app for official distribution.
- **Play Store Deployment**: Final APK/AAB files will be prepared with necessary assets like icons and screenshots for submission to the Google Play Console.
- **Firebase Configuration**: The google-services.json file is configured and linked to the app for proper communication with Firebase services.

5. Testing & Validation

Thorough testing will be conducted to ensure the app's stability, security, and functionality.

- **Unit Testing**: Individual components like the data helper classes and utility functions will be tested to ensure they work as expected.
- **Integration Testing**: The interactions between different parts of the app, such as the UI and the Firebase backend for live location, will be tested to confirm seamless data flow.
- **User Acceptance Testing**: The app will be tested by a group of users to validate its usability and ensure all features meet the project's requirements.

6. Post-Deployment Maintenance

After deployment, the focus will shift to monitoring, bug fixes, and continuous improvement.

- **Monitoring**: Crashlytics and Firebase Analytics will be used to monitor app performance and identify any errors or crashes in production.
- **Updates**: Regular updates will be planned to address bugs, enhance security, and add new features based on user feedback.

2.2. User Manual

"The comprehensive user manual for the SheSafe app is provided in a separate document titled 'SheSafe_User_Manual.docx' to ensure focused and detailed user guidance."

Section E

Critical Appraisal

1. Limitations of the System

The development of SheSafe exposed several technical and architectural challenges that affected feature completeness, reliability, and user experience:

1.1 Inaccurate Location Handling

- The emergency messaging module initially struggled to embed precise latitude and longitude coordinates into SMS messages.
- Messages sometimes contained incomplete or wrong location data, making the Google Maps links unreliable for assisting responders in real time.

1.2 Complexity in Shared Location Feature

- Implementing real-time location sharing between two users was the most challenging module.
- Synchronizing location data and permission requests across Firebase caused asynchronous update failures.
- Location share requests often failed to appear on the recipient's device, requiring multiple rewrites and backend restructuring.

1.3 Firebase Integration Issues

- Default Firebase server regions (Europe) didn't align with the target user region (Southeast Asia), causing location inaccuracy and latency.

- Extensive reconfiguration of Firebase Firestore security rules was required to permit real-time location data sharing.
- Managing authentication and secure data storage across local and cloud components proved complex.

1.4 Mapping and Visualization Problems

- The application initially used Google Maps SDK but switched to OSMDroid due to SDK issues.
- OSMDroid caused new problems, with the map sometimes failing to load and displaying a blue screen on certain devices, limiting live location visuals.

1.5 Limited Offline Capabilities

- Key features like emergency messaging with location and live sharing require an active internet connection.
- Lack of offline support limits app usability in rural or conflict-affected areas with poor or no network connectivity.

1.6 Runtime Permission Handling Gaps

- The app depends on sensitive permissions (SMS, Location, Audio).
- Current permission workflows lack sufficient fallback strategies, breaking critical functionality if permissions are denied or revoked.

2. Recommended Changes and Future Improvements

To address the above limitations and enhance user value, the following improvements are planned:

2.1 Upgrade Map SDK

- Replace OSMDroid with Google Maps SDK for better map rendering, stable location APIs, and reliable region-specific service integration.

2.2 Enhance Permission Workflows

- Introduce educational prompts explaining the necessity of permissions before requests.
- Implement graceful fallbacks or manual alternatives if permissions are denied.

2.3 Add Advanced Safety Features

- Implement automatic silent video recording triggered during panic mode.
- Integrate direct contact options for emergency services (ambulance, police, hospital).
- Provide self-defense tutorials as value-added content for premium users.

2.4 Improve Multi-user Location Sharing Sync

- Refactor backend synchronization to ensure reliable cross-device realtime updates.
- Handle network instability gracefully via retry and reconnection logic.

2.5 Strengthen Security & Privacy

- Enforce end-to-end encryption for sensitive messages.
- Harden Firebase security rules to protect user data.
- Mask personal data explicitly in logs and shared locations.

2.6 Optimize Onboarding and User Interface

- Add interactive onboarding and detailed help guides for new users, especially targeting beginners.
- Simplify UI to make the app accessible for all digital literacy levels.

2.7 Extend Offline Support

- Queue outgoing messages and location updates when offline for automatic sending upon reconnection.
- Cache essential data like contacts and recent location info locally for offline access.

3. Critical Reflection of the Project Experience

This project was a major technical and personal milestone, developed from scratch with no structured tutorials, resulting in deep learning and growth:

3.1 Technical Skill Growth

- Transitioned from a simple Java + SQLite app to a complex Firebase-integrated safety app with real-time sync.
- Gained hands-on experience with Firebase Authentication, Firestore, location APIs, sensor integration, and media handling.

3.2 Problem-Solving Under Pressure

- Debugged and overcame complex issues in live location synchronization, Firebase security rules, and permission management.
- The shared location feature presented the steepest learning curve, requiring multiple design iterations.

3.3 Process-Oriented Development

- Followed a clear iterative approach: drafting features → UI prototyping → coding → testing → refining.
- Ensured each feature was fully functional and user-friendly before proceeding, improving stability and maintainability.

3.4 Socially Driven Motivation

- The app's mission to serve vulnerable populations in crisis-prone areas (e.g., Myanmar) influenced design choices emphasizing free, accessible safety tools.
- Planned premium enhancements focus on meaningful user empowerment like self-defense education.

3.5 Key Lessons Learned

- Real-world app development requires not just coding, but usability, robustness, data privacy, and empathy.
- Time management, patience, and adaptability are critical for overcoming roadblocks.
- Creating a full-fledged app independently fosters confidence and a solid foundation for future projects.

Conclusion

The SheSafe application integrates location sharing, emergency communication, and user-centred design into a single, seamless mobile application, thereby meeting the critical demand for an accessible safety platform. The project exhibits great problem-solving and iterative development processes in spite of technical difficulties including handling permissions gracefully, integrating precise location data, and syncing shared locations across numerous users. With a secure Firebase backend and user-friendly procedures, the final app provides dependable essential safety functionalities. The project emphasised the value of privacy, security, and offline capabilities while improving skills in Android development, real-time data synchronisation, cloud services, and user experience design. With planned future enhancements including advanced safety features and improved UI, SheSafe holds promise as a valuable tool to aid individuals in urgent situations and contribute positively to public safety efforts.

Reference list

Android Studio (2024). *Android Studio and SDK tools*. [online] Android Developers. Available at: https://developer.android.com/studio [Accessed 18 Jun. 2025].

Bechtold, S. (2016). *JUnit 5 User Guide*. [online] Junit.org. Available at: https://docs.junit.org/current/user-guide/ [Accessed 20 Jun. 2025].

Blog.back4app. (2020). *Google Firebase Pricing for Dummies*. [online] Available at: https://blog.back4app.com/firebase-pricing/ [Accessed 15 Jun. 2025].

draw.io (2025). *Flowchart Maker & Online Diagram Software*. [online] app.diagrams.net. Available at: https://app.diagrams.net/ [Accessed 22 Jul. 2025].

Firebase. (n.d.). *Firebase Products*. [online] Available at: https://firebase.google.com/products-build [Accessed 18 Jun. 2025].

Jobicy. (2025). *Software Developer*. [online] Available at: https://jobicy.com/salaries/mm/software-developer#salary-section [Accessed 15 Jun. 2025].

Linkedin.com. (2025). *Teslio*. [online] Available at: https://www.linkedin.com/jobs/view/freelance-software-tester-burmese-at-testlio-4087442964/?originalSubdomain=mm [Accessed 15 Jun. 2025].

Mobilekingmyanmar.com. (2025). *Price List – Mobile King | Mobile, IT & Electronic*. [online] Available at: https://mobilekingmyanmar.com/price-list/ [Accessed 15 Jun. 2025].

Openstreetmap.org. (2024). *osmdroid - OpenStreetMap Wiki*. [online] Available at: https://wiki.openstreetmap.org/wiki/Osmdroid [Accessed 20 Jun. 2025].

PayLab. (2025). *Salary Info*. [online] Available at: https://www.paylab.com/mm/salaryinfo/information-technology [Accessed 15 Jun. 2025].

SQLite (2019). *SQLite Home Page*. [online] Sqlite.org. Available at: https://sqlite.org/index.html [Accessed 18 Jun. 2025].

staruml.io. (2025). *StarUML*. [online] Available at: https://staruml.io/ [Accessed 20 Jun. 2025].

Win, C.T. (2025a). *COS209-Diagrams*. [online] Available at: https://drive.google.com/drive/folders/1WzgVOpR-nFCxoDZDK-xR60uvF192cGUJ?usp=sharing [Accessed 29 Aug. 2025].

Win, C.T. (2025b). *Hierarchy Chart*. [online] Available at: https://drive.google.com/file/d/1W30vGyvIWQaOfoosooJHTR_y0JH8qKVo/view?usp=sharing [Accessed 31 Jul. 2025].

Win, C.T. (2025c). *Sample UI Draft*. [online] Available at: https://www.figma.com/design/sY9bOxF7MebPyL62SSYBY2/SheSafe?node-id=0-1&p=f&t=gnjSgCuLZRNtKEUG-0 [Accessed 18 Jun. 2025].

World Salaries. (2025). *Average UX Designer Salary in Myanmar for 2025*. [online] Available at: https://worldsalaries.com/average-ux-designer-salary-in-myanmar/ [Accessed 15 Jun. 2025].