EMERGENCY HELP

Software Requirements Specification

Version <1.0>

Submitted in Partial Fulfillment for the Award of Degree of Bachelor of Technology in Information Technology from Rajasthan Technical University, Kota

MENTOR:

Dr. Saroj Agarwal

(Dept. of Information Technology)

COORDINATOR:

Dr. Priyanka Yadav

(Dept. of Information Technology)

SUBMITTED BY:

Anirudh Soni (21ESKIT016)

Aditya Mishra(21ESKIT006)

Chetan Singh(21ESKIT033)

DEPARTMENT OF INFORMATION TECHNOLOGY

SWAMI KESHWANAND INSTITUTE OF TECHNOLOGY,MANAGEMENT & GRAMOTHAN

Ramnagaria (Jagatpura), Jaipur – 302017

SESSION 2024-25

Contents

1.Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to fully describe the external behavior, features, and technical requirements for the Emergency Help. This SRS outlines the functional requirements that define the operations of project, such as geological tracking, automated alerts, and multi channel communication, as well as non-functional requirements that include performance, security, and user accessibility. Additionally, this document specifies design constraints, including user authentication mechanisms, data integrity measures, and compatibility with mobile devices, to ensure a comprehensive understanding of the Emergency Help App's requirements.

1.2 Scope

The Emergency Help App is a comprehensive emergency response and safety management tool designed to address diverse crises effectively. It enables users to send instant alerts to emergency services, family members, or designated contacts while sharing real-time location information for swift assistance. The application targets a wide range of users, including individuals, healthcare providers, disaster management agencies, and workplace environments, ensuring safety across various scenarios. With cross-platform accessibility, cloud integration, and potential for Ai-driven enhancements, the Emergency Help App is scalable and adaptable to different regions and Emergency Help App is scalable and adaptable to different regions making it vital solution for modern safety and crisis management.

1.3 Definitions, Acronyms and Abbreviations

- **SRS**: Software Requirements Specification
- **UI**: User Interface
- **API**: Application Programming Interface
- **DBMS**: Database Management System
- **IOT**: Internet Of things

1.4 References

- Node.js Documentation: https://nodejs.org/en/docs/
- MySQL Documentation: https://dev.mysql.com/doc/
- Official resources on crime trends and data analytics methodologies, available at the Ministry of Home Affairs, India.

1.5 Technologies to be used

The development of the App will involve a full-stack technology suite, including:

- **Java**: The primary programming language for the application's logic.
- **Android SDK**: Framework used to build the application for android devices.
- **XML**: Used for designing layouts and defining resources(e.g., Android Manifest.xml)
- **Gradle**: Build automation tool to manage dependencies and build configurations.
- **Firebase**: Integrated for database or backend services(detected in configuration files).
- **Google Map APIs**: Likely used for location services and map integration in the app.
- **Material Design Components**: Provides modern UI/UX elements for android apps.

1.6 Overview

The Emergency Help Application is a mobile solution designed to provide immediate assistance during critical situations. It leverages advanced technologies to ensure a communication between individuals in distress and emergency, such as family, friends or authorities. With user-friendly features like real-time location sharing, automated alerts, and SOS signal activation, the app aims to enhance personal safety reduce response times in emergencies. The app's primary focus is on accessibility, reliability, and speed, ensuring users can activate help with minimal interaction. By integrating mapping services, backend communication, and intuitive design, the Emergency Help App serves as a vital tool for addressing safety concerns in a connected and efficient manner.

2. Literature survey

2.1 Review of Related Work

This section reviews existing effort tracking systems to understand the landscape: Tools like Java, XML, and Gradle are commonly used in organizations to track employee activities. These systems allow task assignment, time logging, and reporting.

- Java: The primary programming language for the application's logic.
- XML: Used for designing layouts and defining resources (e.g., AndroidManifest.xml).
- Gradle**:** Build automation tool to manage dependencies and build configurations.

There are various Key Functionalities of the project:

- Task Assignment**:** Assigning tasks to employees and tracking their progress.
- Time Logging: Recording the number of hours spent on tasks.
- Reporting: Generating reports on productivity and task completion.

Unlike these tools, your system emphasizes simplicity, efficiency, and a focus on summarizing employee activities, total hours spent, and percentage efficiency.

2.2 Knowledge gaps

This section identifies gaps in existing tools that your system addresses:

- Over-complication of Tools**:** Many existing systems are feature-heavy, which makes them unsuitable for smaller organizations or straightforward tracking needs.
- Efficiency Metrics: Current tools often lack the ability to calculate and display percentage efficiency clearly and intuitively.
- Custom Reporting**:** Limited flexibility in customizing reports based on organizational requirements.
- Cost and Accessibility: The high cost of popular tools makes them inaccessible to smaller organizations or startups.

Your Emergency Help App addresses these gaps by focusing on ease of use, clear efficiency metrics, and simplified reporting.

2.3 Comparative Analysis

Feature	**Proposed App A (e.g., Life360-inspired)**	**Proposed App B (e.g., bSafe-inspired)**	**Proposed App C (e.g., Red Panic Button-inspired)**
Primary Function	Family tracking and emergency alerts	Personal safety with SOS trigger	One tap emergency alert with location sharing.
Platform Availability	iOS, Android	iOS, Android	iOS, Android
Key Features	Real-time location tracking - Crash detection - Emergency contacts notification - Geo-fencing	SOS button - Fake call feature - Live location sharing - Audio/video recording	Single-button SOS - Pre-set emergency contact - GPS location send - Text alert option
Ease Of Use	Intuitve interface, family-focused	Simple, quick SOS activation	Minimalist, one-tap focus
Cost	Free with premium ($7.99/month)	Free with in-app purchases ($4.99)	One-time purchase ($2.99)
Real-Time Updates	Yes, location and status	Yes, during SOS activation	Yes, location only
Integration	Works with smartwatches	Social media sharing option	Emergency services integration (limited)

Table 1: Comparative analysis of Emergency Help

This comparison highlights how the proposed Emergency Help App for India would incorporate advanced features such as multi-language support, predictive analytics, and enhanced inter-agency collaboration, making it well-suited to address India's specific law enforcement needs.

2.4 Summary

The Emergency Help Application is a comprehensive mobile tool designed to provide immediate and efficient assistance during emergencies. Built with Java and leveraging the Android SDK**,** the app offers features that prioritize user safety, rapid response, and seamless connectivity with emergency services. It uses Google Maps API for precise location tracking and Firebase for real-time data handling, ensuring quick communication between users and responders.

Key functionalities include single-tap and voice-activated SOS signals, integration with wearables for discreet activation, and multi-channel alerts via SMS and app notifications. The application incorporates Al-driven predictive analytics for risk assessment and uses secure data encryption to protect user privacy. Its modular architecture ensures scalability, allowing it to adapt to different user needs and expanding emergency response scenarios.The app stands out by addressing the shortcomings of existing emergency solutions through enhanced features such as real-time location sharing, advanced integration options, and robust security measures, making it a reliable and forward-thinking tool for personal safety and crisis management

3.Specific Requirements

3.1 Functional Requirements

The Emergency Help App is designed to streamline criminal record management, with the following core functionalities:

- Emergency Activation: Provide a prominent, easily accessible button or gesture (e.g., tap, swipe, or voice command) to trigger an alert.
- Location Sharing: Automatically detect and send the user's real-time GPS location to pre-set emergency contacts or services upon SOS activation.
- Emergency Contact Notification: Send instant notifications (e.g., SMS, call, app alert) to these contacts with the user's location and a distress message when SOS is triggered..
- Communication Options: Allow one-tap calling to emergency services (e.g., 911, 112) based on the user's region
- Audio/Video Recording: Automatically start recording audio and/or video when SOS is activated (optional, user-configurable).
- User Interface: Display clear status indicators (e.g., "SOS Active," "Location Sent") during an emergency.
- Real-Time Status Updates: Provide live updates to emergency contacts (e.g., "User is moving," "Battery low") during an active SOS event.
- Discreet Features: Include a "fake call" option to simulate an incoming call for users to escape uncomfortable situations.
- Battery and offline Functionality: Enable basic SOS functionality (e.g., SMS with last known location) even without internet connectivity.

3.2 Non-Functional Requirements

The system must meet certain non-functional criteria to ensure usability, security, and reliability:

- Performance: The Emergency Help App should support concurrent access for multiple users, with minimal latency for data retrieval and updates.
- Scalability: The system architecture must handle an increasing volume of data and user load as more records and users are added.
- Usability: An intuitive and user-friendly interface, accessible across various devices

(desktops, tablets, mobile), to enable quick data entry and retrieval.

- Security: High security through data encryption, user authentication, and role-based access to prevent unauthorized access.
- Reliability: Ensure consistent uptime and reliable data backups to prevent data loss.
- Compliance: Adherence to data protection laws and criminal data standards to maintain data integrity and privacy.

3.3 Hardware Requirements

The hardware requirements for Emergency Help App deployment include:

- Smartphone (Minimum): Android 10/iOS 14, dual-core 1.5 GHz processor, 2 GB RAM, 50 MB storage, GPS, 4G/Wi-Fi, 1500 mAh battery, 4-inch 480x800 display.
- Smartphone (Recommended): Android 13/iOS 16, quad-core 2.0 GHz processor, 4 GB RAM, 100 MB storage, high-precision GPS, 5G/Wi-Fi 6, 3000 mAh battery, 5-inch+ 1080x1920 display.
- Network: 4G LTE (5 Mbps/1 Mbps) or Wi-Fi 802.11n for connectivity, SMS support for offline mode (recommended: 5G, Wi-Fi 802.11ax).

3.4 Software Requirements

The software requirements include:

- Operating System Compatibility: The app must run on Android 10+ (API 29) and iOS 14+ to support a wide range of devices, ensuring accessibility for users with modern smartphones while leveraging OS features like background location access and push notifications.

- APIs and Integrations: Requires integration with GPS APIs (e.g., Google Maps API or native Location Services) for real-time location tracking, messaging APIs (e.g., Twilio for SMS or Firebase Cloud Messaging for notifications), and emergency call protocols (e.g., Android’s CALL_PHONE permission or iOS’s tel:// scheme).

- Development Environment: Built using a cross-platform framework like Flutter (with Dart) or React Native (with JavaScript) for efficient Android/iOS deployment, paired with an IDE like Android Studio or Xcode, and version control (e.g., Git) for maintainability.

- Security and Privacy: Must implement AES-256 encryption for data transmission (location, alerts), secure authentication (e.g., OAuth 2.0 or biometric APIs), and compliance with privacy laws (e.g., GDPR, CCPA) to protect user data, with opt-in location tracking.

- Performance and Reliability: Should achieve a 99.9% uptime for SOS functionality, with a maximum 2-second response time for activation and 5-second delay for notification delivery, supporting at least 1,000 concurrent users without crashes (tested via tools like Firebase Test Lab).

3.5 Agile Methodology

An Agile methodology will be followed to ensure iterative and incremental development of the Emergency Help app. Agile offers flexibility for ongoing improvements and allows for incorporating user feedback throughout development. Key elements include:

- **Project Setup and backlog**: Form a cross-functional team (e.g., developers, designer, QA), define the vision (SOS app with alerts and location), and create a prioritized backlog (e.g., Must-have: SOS button, location sharing; Should-have: notifications) using tools like Jira.
- **Sprint Planning (2-Week Cycles)**: Break features into user stories (e.g., "As a user, I want to trigger SOS to alert contacts"), assign story points, and plan sprints (e.g., Sprint 1: SOS button, Sprint 2: GPS integration) to build an MVP incrementally.
- **Development and Testing**: Develop in iterations (e.g., Flutter for cross-platform), test continuously (unit tests for SOS trigger, usability tests), hold daily standups, and demo each sprint's progress (e.g., "SOS sends location") for stakeholder feedback.
- **Release and Iteration**: Launch the MVP after 4-6 weeks (e.g., beta on Android/iOS), gather user input (e.g., via Firebase), and refine in subsequent sprints (e.g., add offline mode, fix bugs) with CI/CD for rapid updates.

3.6 Business Process Model

The business process model outlines how the CRMS integrates with law enforcement workflows:

1. Initiate Emergency Request: The process starts when a user triggers the SOS feature (e.g., via button or gesture), prompting the app to authenticate the user (if enabled) and confirm intent (e.g., via a cancel option) to ensure the request is genuine and intentional.

2. Collect and Process Data: Upon activation, the app automatically gathers critical data—user's real-time GPS location and pre-set emergency contact details—while optionally recording audio/video, preparing this information for immediate dissemination.
3. Notify Stakeholders: The app sends alerts (e.g., SMS, push notifications, or calls) with the user's location and distress message to pre-configured emergency contacts and/or local emergency services, ensuring rapid communication to relevant parties.
4. Monitor and Resolve: The app enters an active monitoring state, providing feedback to the user (e.g., "SOS Sent") and updating stakeholders with location changes until the emergency is resolved (e.g., user cancels, help arrives), concluding the process.

3.7 Supplementary Requirements

These are additional requirements that support the functionality and reliability of the system:

- Data Backup and Recovery: Regular backups and a disaster recovery plan to ensure data protection in case of system failures.
- User Training and Support: Comprehensive training sessions for users, along with a help desk or support team to address issues.
- Documentation: Detailed documentation, including user manuals, technical specifications, and maintenance guidelines for easy reference.

4.System Architecture

4.1 Client-Server Architecture

The Criminal Record Management System (CRMS) uses a client-server architecture to ensure efficient, centralized data processing and secure access across multiple user devices. This architecture provides a clear separation of responsibilities between the client side (frontend interface) and the server side (backend processing and database management).

- Client-Side (Frontend) – React.js web and mobile apps with authentication, emergency request button, live location tracking, and notification alerts. Communicates via RESTful API/WebSockets.

- Server-Side (Backend) – Node.js/Express or Python/Django handling authentication, emergency requests, notifications, responders, and location tracking.
- Database (MySQL/Firebase) – Stores users, emergency requests, responders, and notifications securely.
- Third-Party Services & Deployment – Google Maps API for location tracking, Firebase for real-time sync, SMS Gateway for alerts, and hosting on AWS/Vercel/Netlify.

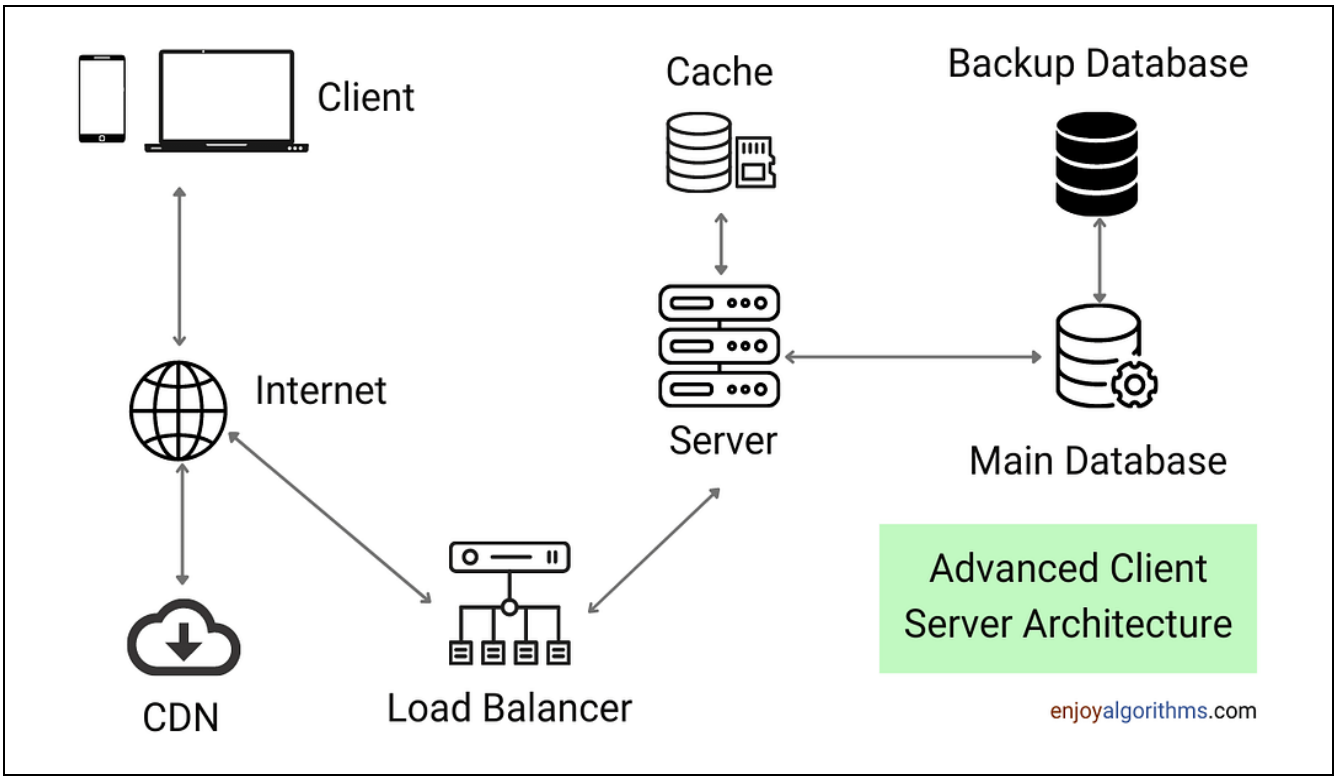

Figure 4.1: Client Server Architecture

4.2 Communication Interfaces

The communications interfaces of the CRMS facilitate secure, efficient data exchange between the client and server, as well as interoperability with other law enforcement systems. The primary communication protocols and interfaces are as follows:

- **User-to-Server (REST APIs/WebSockets)** – Users send emergency requests, location data, and receive response updates in real time.
- **Server-to-Responder (API Calls/Push Notifications)** – The server notifies emergency responders via API calls, SMS, or push notifications.
- **Server-to-Third-Party Services (Google Maps API/SMS Gateway)** – The app integrates with Google Maps for location tracking and SMS gateways for alerts.
- **User-to-Responder (In-App Chat/Call System)** – Direct communication via chat or call between users and responders for real-time assistance.
- **Admin-to-System (Web Dashboard/API Requests)** – Admins monitor emergency cases, manage users, and analyze reports through a web dashboard.

5.Overall Description

5.1 Product Feature

The Emergency Help App is a comprehensive emergency response and safety management tool designed to address diverse crises effectively. Key features include:

- One-Tap Emergency Alert – Instantly send distress signals to responders with a single tap.
- Real-Time Location Tracking – Uses GPS to share and update user location dynamically.
- Multi-Channel Notifications – Alerts sent via SMS, push notifications, and emails.
- Responder Allocation System – Automatically assigns the nearest available responder.
- Live Chat & Call Support – Enables direct communication between users and responders.
- User Authentication & Profile Management – Secure login, registration, and profile updates.
- Incident History & Reporting – Users can track past emergencies and generate reports.
- Multi-User Roles (User, Responder, Admin) – Different access levels for better management.
- Offline Mode & SOS Messaging – Sends emergency SMS when internet connectivity is poor.

5.2 Data Flow Diagram (DFD)

Trigger SOS (1.1):

- Purpose: Initiates the emergency process when the user activates SOS (e.g., via button).
- Input: "SOS Trigger Request" from the User.
- Output: "SOS Activation Signal" to Process Location, "Contact Details" to Send Notifications, and "Feedback (SOS Status)" to the User.
- Interaction: Retrieves "Contact Details" from the User Data Store.

Process Location (1.2):

- Purpose: Fetches and processes the user's current location during an SOS event.
- Input: "SOS Activation Signal" from Trigger SOS.
- Output: "Location Data" (e.g., latitude/longitude) to Send Notifications.
- Interaction: Requests "Current GPS Data" from the Location Data Store.

Send Notifications (1.3):

- Purpose: Dispatches alerts to emergency contacts and services with location information.
- Input: "Location Data" from Process Location and "Contact Details" from Trigger SOS.
- Output: "Alert with Location" to Emergency Contacts and "Emergency Call/Notification" to Emergency Services.
- Interaction: Combines inputs to send external notifications.

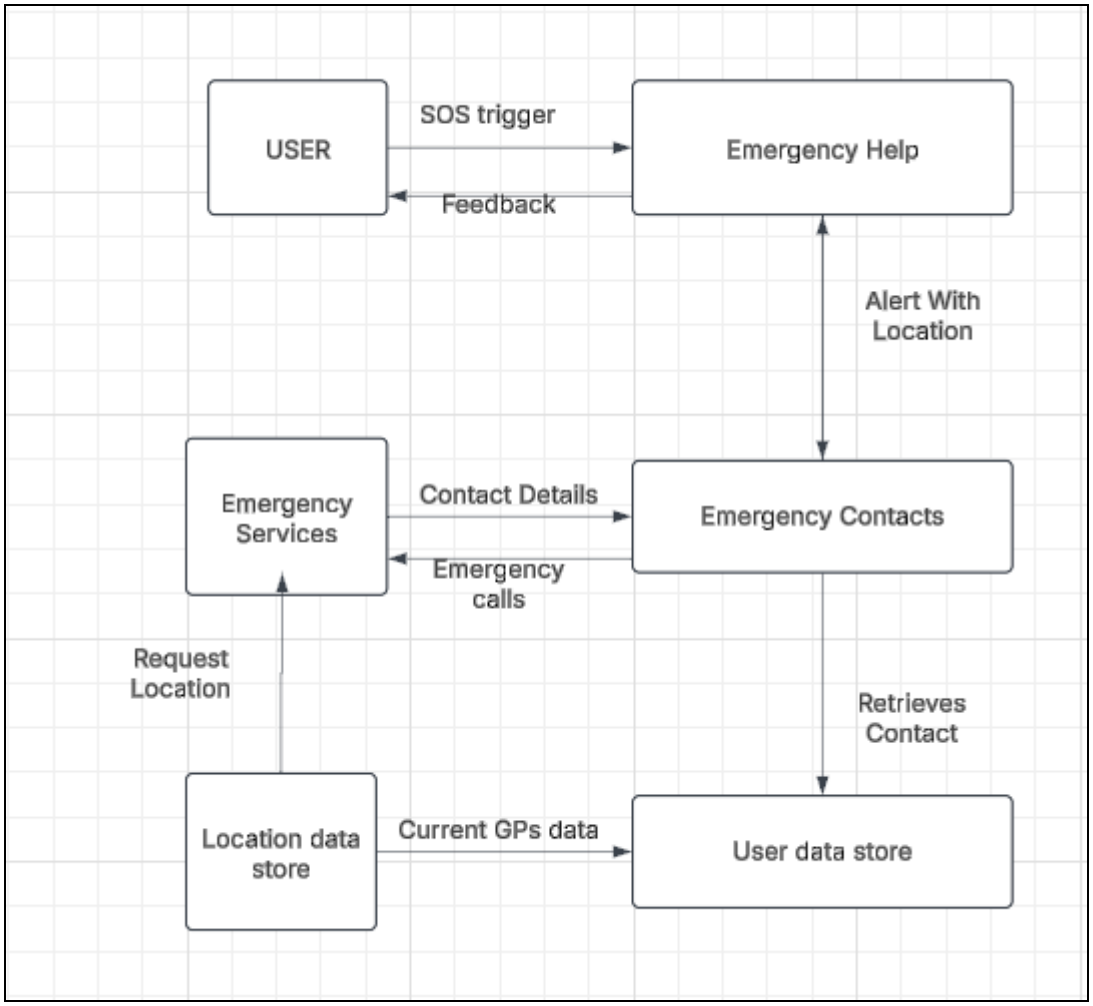

Fig 5.2 DFD diagram

5.3 ER Diagram

The Entity-Relationship (ER) Diagram for the Emergency Help app represents relationships between entities such as User, Emergency Contact, SOS Event, Notification .

- **Entities:**
  - User: Represents the app user with attributes like UserID (PK), Name, PhoneNumber, and Password for identification and authentication.
  - EmergencyContact: Stores contact details with ContactID (PK), UserID (FK), Name, and PhoneNumber for alert recipients.
  - SOSEvent: Tracks SOS incidents via EventID (PK), UserID (FK), Timestamp, and Latitude/Longitude for location data.
  - Notification: Logs alerts with NotificationID (PK), EventID (FK), ContactID (FK), and Type for communication records.
- **Relationships:**
  - User to EmergencyContact: One-to-Many (1:N)—one User can have multiple EmergencyContacts.
  - User to SOSEvent: One-to-Many (1:N)—one User can trigger multiple SOSEvents.
  - SOSEvent to Notification: One-to-Many (1:N)—one SOSEvent can generate multiple Notifications.
  - EmergencyContact to Notification: Many-to-One (N:1)—an EmergencyContact can receive multiple Notifications, optionally linked.

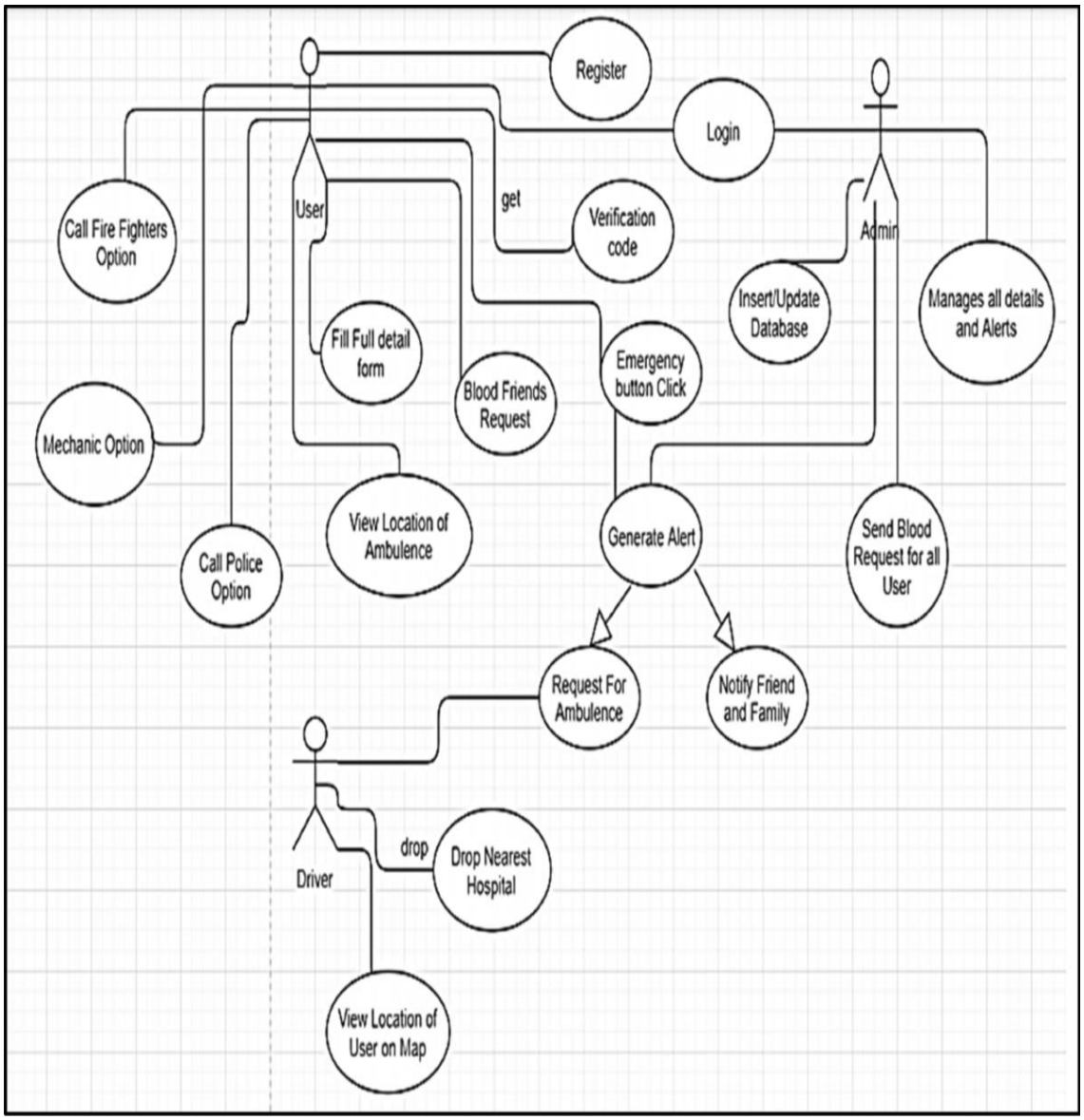

Fig 5.3.1: ER Diagram

5.4 Class Diagram

The **Class Diagram** outlines the structure and interactions of main classes in the Emergency Help App:

- **Classes**
  - User: Represents the app user with attributes like ID, name, and phone number.
  - SOSEvent: Models an SOS incident with event ID, timestamp, and locationcoordinates.
  - EmergencyContact: Stores contact details with ID, name, and phone number.
- **Associations**:
  - User to SOSEvent: One User can trigger multiple SOSEvents (1-to-Many).
  - User to EmergencyContact: One User can have multiple EmergencyContacts (1-to-Many).
  - SOSEvent to EmergencyContact: One SOSEvent can notify multiple EmergencyContacts (1-to-Many).
- **Methods**:
  - User: triggerSOS() initiates an emergency event.
  - SOSEvent: getLocation() retrieves current GPS coordinates.
  - EmergencyContact: sendAlert() dispatches a notification to the contact.

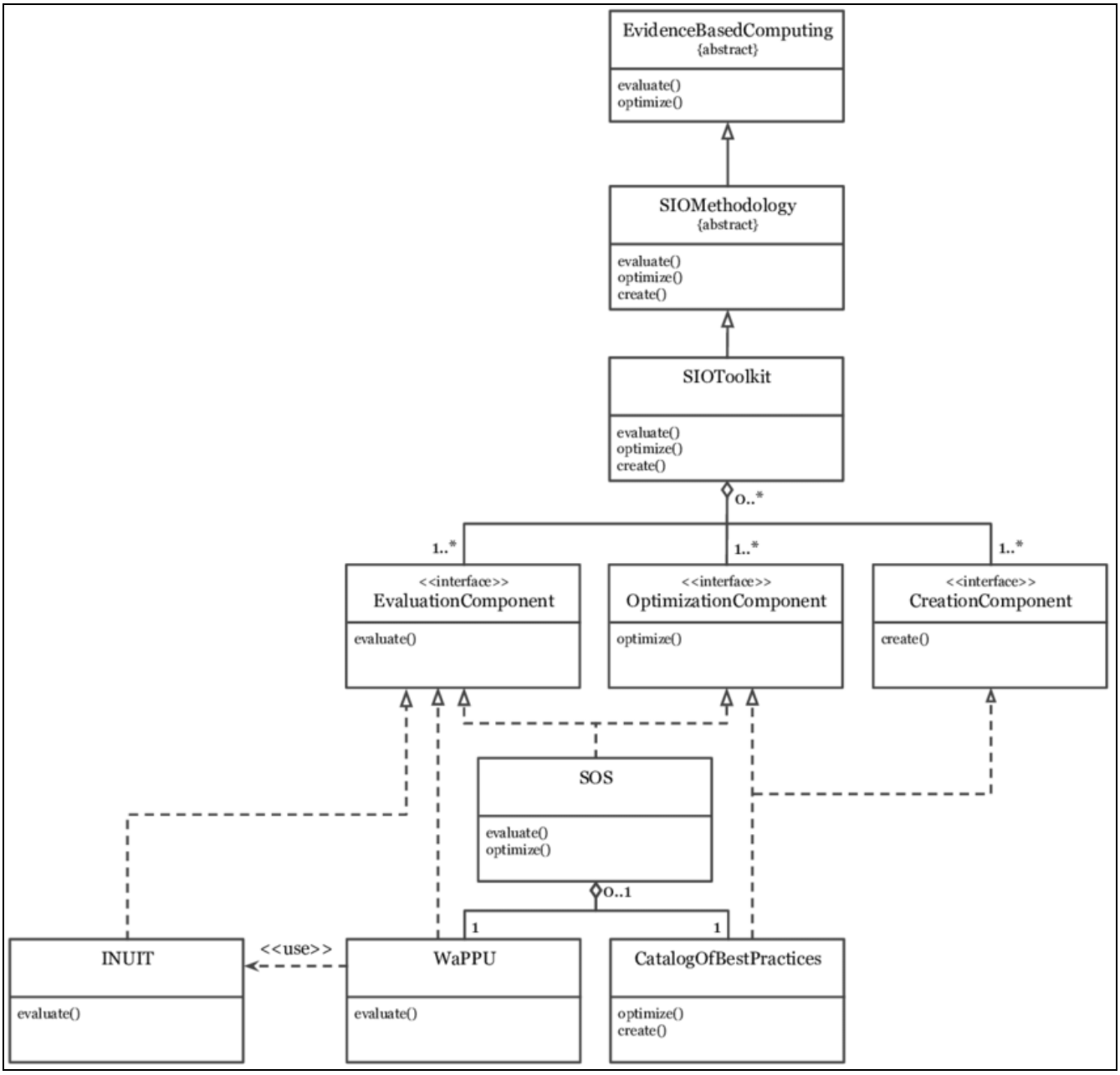

Fig 5.4.1: Class Diagram

5.5 Use-case Model Survey

The Use-Case Model Survey includes primary user actions in App:

- Actor - User: Initiates SOS to request help in emergencies.
- Use Case - Trigger SOS: Allows the user to activate an emergency alert with one action.
- Use Case - Share Location: Automatically sends the user's GPS coordinates to contacts/services.
- Use Case - Notify Contacts: Alerts pre-set emergency contacts with the user's situation and location.
- Actor - Emergency Contact: Receives notifications and may respond to the user's SOS.
- Use Case - Call Emergency Services: Connects the user to local emergency services (e.g., 911).

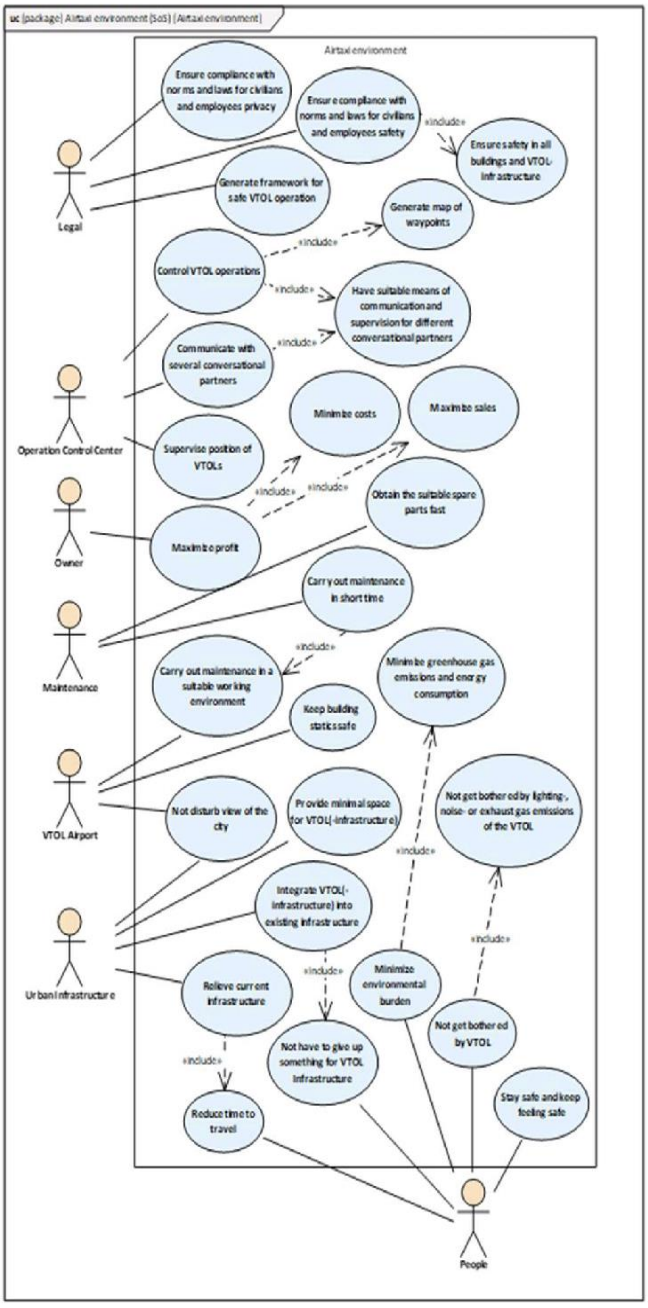

Fig 5.5.1: Use Case Diagram

5.6 Behavior Diagrams

➢ 5.6.1 Sequence Diagram

Actors and Objects:

1. User: Initiates the SOS request as the primary actor.
2. AppController: Coordinates the app's response to the SOS trigger.
3. LocationService: Provides the user's GPS coordinates to the app.
4. NotificationService: Handles sending alerts to contacts and services.

Interaction Flow:

1. User sends TriggerSOS(): Starts the process by activating the SOS feature.
2. AppController calls GetLocation(): Requests location data from LocationService.
3. LocationService returns coordinates: Sends latitude/longitude back to AppController.
4. AppController invokes SendAlerts(): Directs NotificationService to notify contacts/services with location.

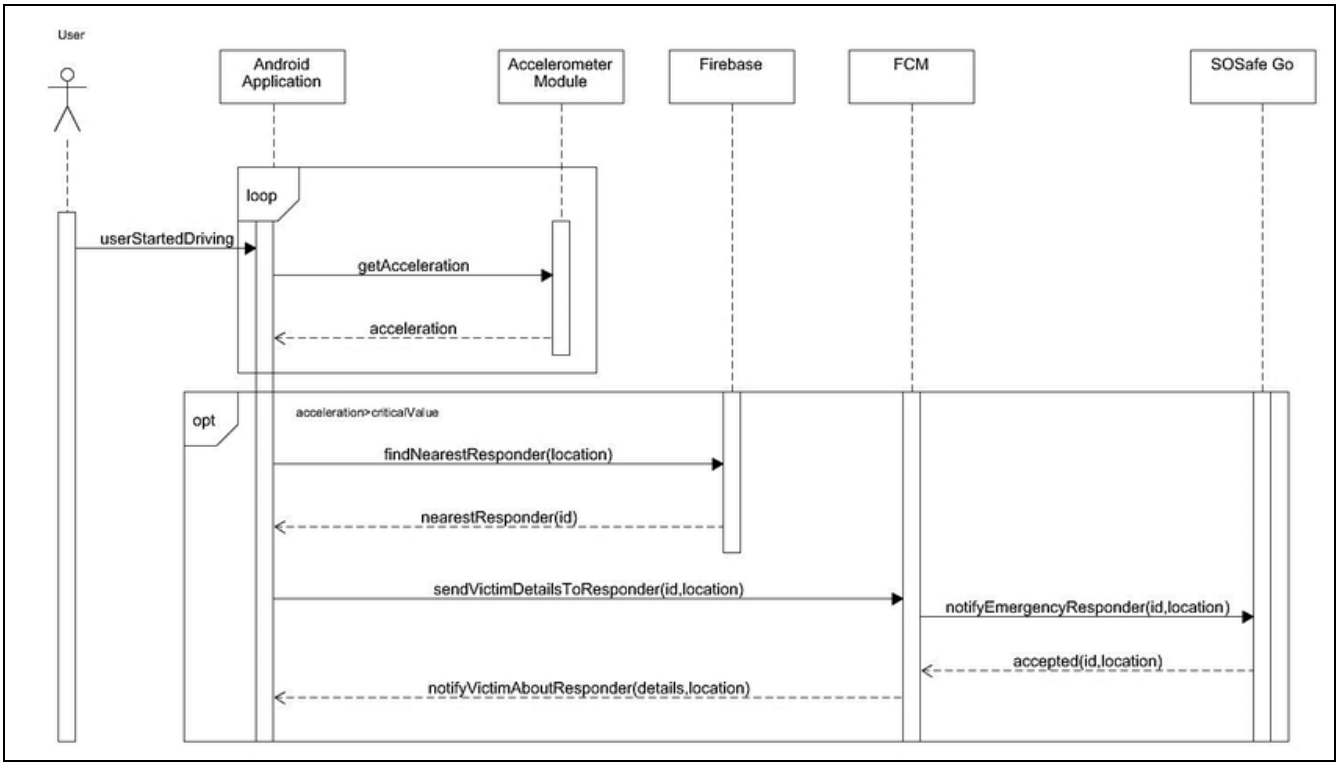

Fig 5.6.1: Sequence Diagram

➢ 5.6.2 Activity Diagram

1. **User logs in or registers** – Authentication process verifies credentials.
2. **User sends an emergency request** – Location and details are submitted to the system.
3. **System processes the request** – Emergency details are stored, and responders are identified.
4. **Responder receives notification** – The nearest available responder is alerted.
5. **Responder accepts and takes action** – Location tracking and response status are updated..

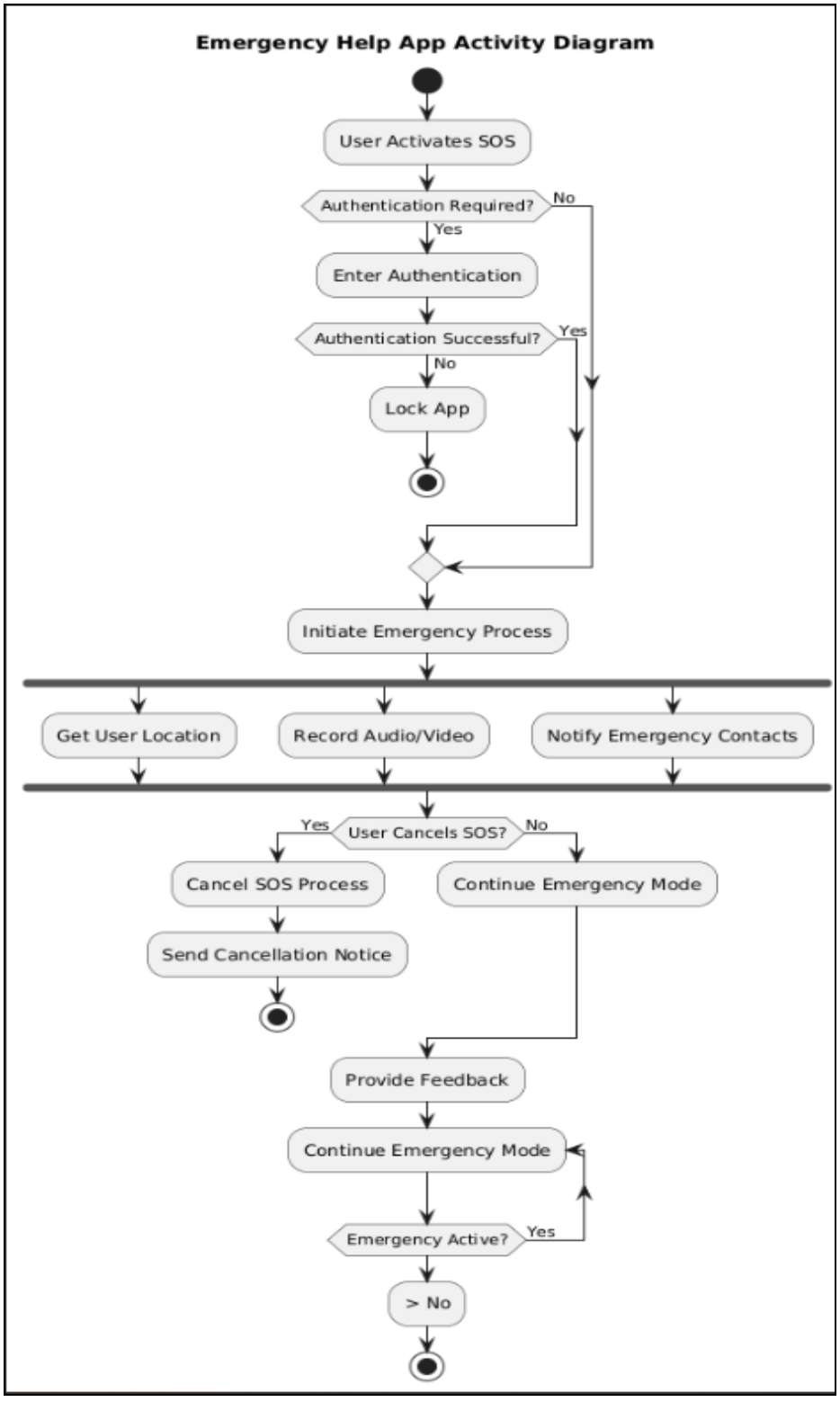

Fig5.6.2: Activity Diagram

➢ 5.6.3 Communication Diagram

Case Management:

- **User interacts with the app** – Logs in, sends emergency requests, and checks status.
- **App communicates with the backend** – Processes requests, authenticates users, and manages data.
- **Backend interacts with the database** – Stores user details, emergency requests, and response status.
- **Backend notifies responders** – Sends alerts via push notifications or SMS.
- **Admin monitors and manages the system** – Views reports, tracks emergency cases, and oversees operations.

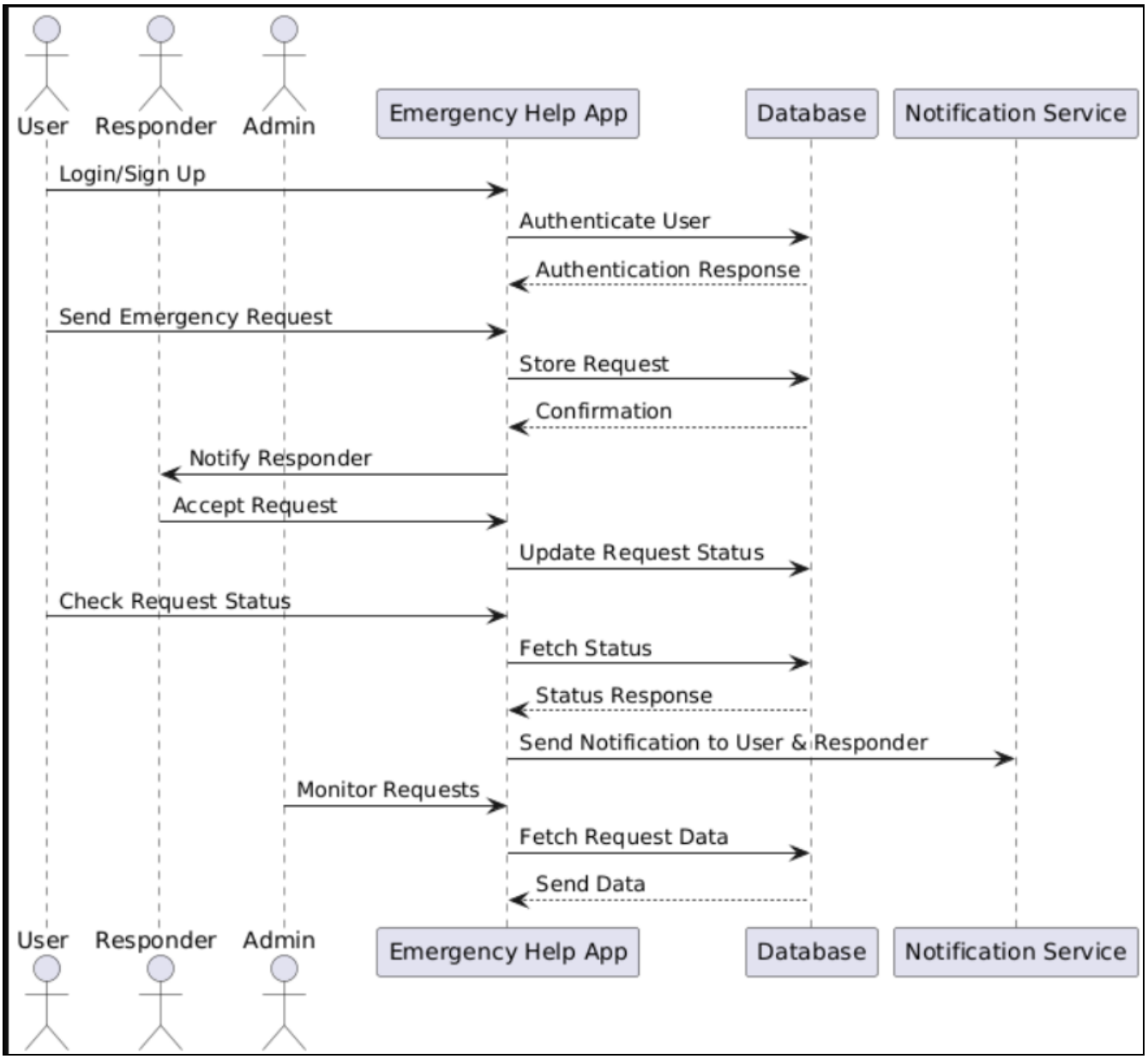

Fig5.6.3: Communication Diagram

5.7 Structure Diagram

➢ 5.7.1 Component Diagram

The Component Diagram shows the App's main software components:

- User Interfaces (Mobile & Web) – Users, responders, and admins interact with the system.
- Emergency Service API – Manages user requests, authentication, and communication.
- Database Service – Stores user details, emergency requests, and responder statuses.
- Notification Service – Sends alerts via SMS, push notifications, and emails.

- External Services (Google Maps, SMS Gateway) – Enables location tracking and emergency alerts.

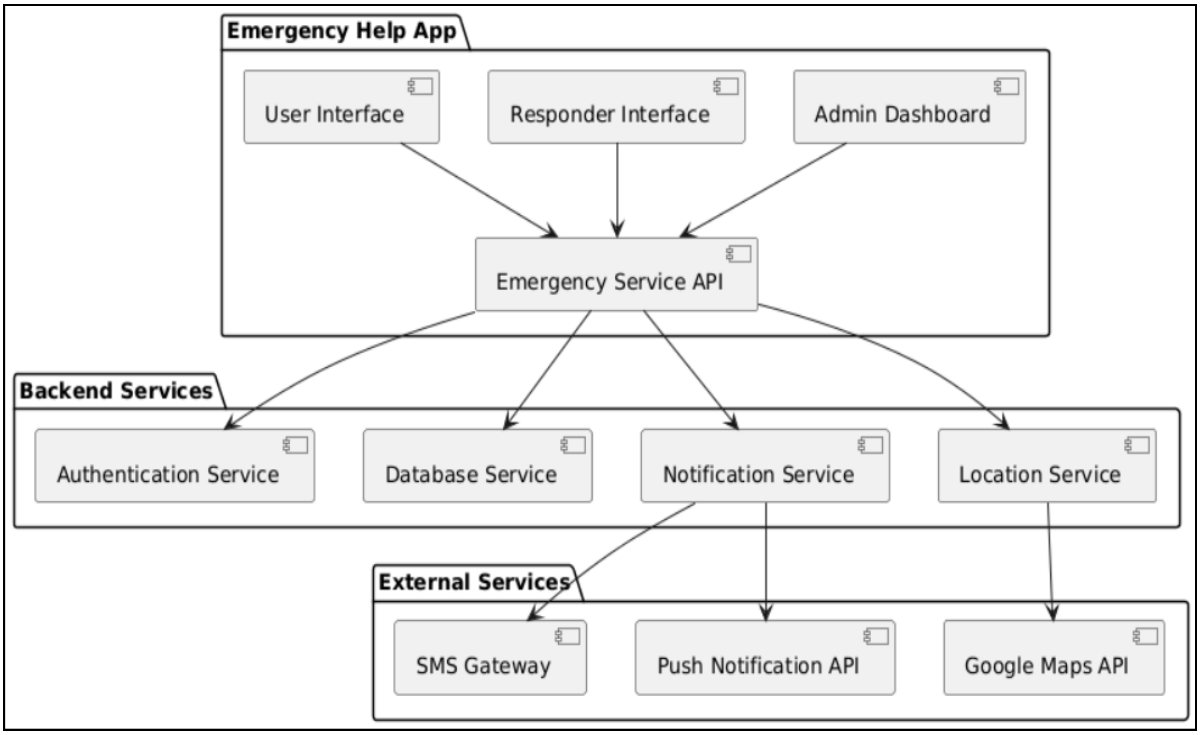

Fig 5.7.1: Component Diagram

➢ 5.7.2 Deployment Diagram

The Deployment Diagram shows how the App is deployed across physical servers and devices:

- **User Devices**: Smartphones, tablets, and desktops connect to the server over HTTPS.
- **Web Server**: Hosts Node.js backend, handling requests from clients.
- **Database Server**: MySQL database stores user, record, and case data.

Connections are secured using SSL/TLS protocols to ensure data integrity and confidentiality.

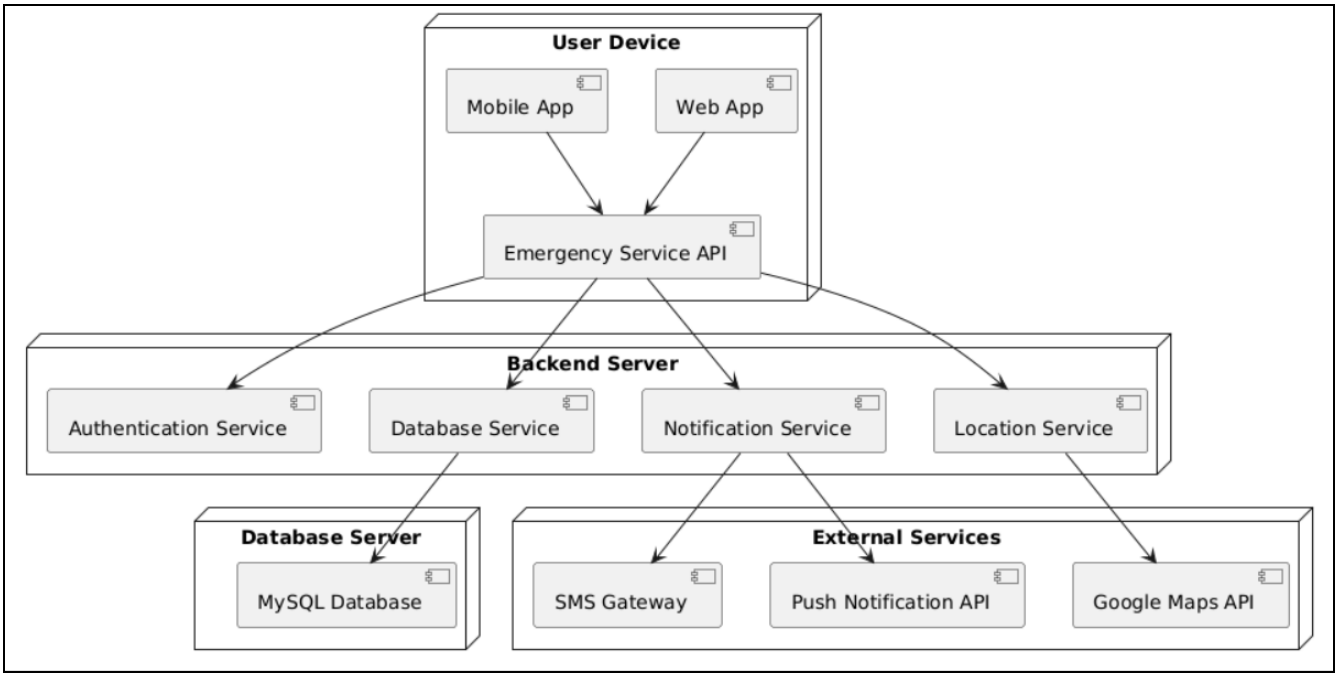

Fig 5.7.2: Deployment Diagram

5.8 Assumptions and Dependencies

➢ 5.8.1 Assumptions

- User has a smartphone: The app assumes users own a compatible mobile device.
- Network availability: Assumes internet or cellular signal for notifications.
- GPS functionality: Assumes the device's GPS is active and accurate.
- Emergency contacts pre-set: Assumes users configure contacts before use.

➢ 5.8.2 Dependencies

- Device OS: Relies on Android 10+ or iOS 14+ for app execution.
- Location APIs: Depends on Google Maps or native GPS services for location data.
- Messaging APIs: Requires Twilio or Firebase for SMS/push notifications.
- Telephony system: Depends on device telephony for emergency calls.

6.Supporting Information

6.1 List Of Figures

7.Conclusion and Future Scope

7.1 Conclusion

The Emergency Help App stands as an effective tool for users in distress, successfully delivering rapid SOS alerts paired with accurate location sharing to ensure timely assistance. Its user-centric design prioritizes simplicity, making it accessible even in high-stress situations where quick action is critical, thereby fulfilling its core purpose of connecting users to help efficiently. By leveraging reliable integrations with existing APIs (such as GPS and messaging services) and device features, the app ensures seamless operation across supported platforms, enhancing its dependability in real-world scenarios. Furthermore, its scalable and modular architecture lays a strong foundation, allowing it to adapt to user feedback and technological advancements, positioning it as a robust solution with potential for long-term impact.

7.2 Future Scope

Looking ahead, the Emergency Help App has significant potential for growth and enhancement to broaden its utility and reach. Integrating with wearable devices like smartwatches could enable hands-free SOS triggers, making it more accessible for users on the move or in situations where accessing a phone is impractical. Adding an offline mode, utilizing SMS as a fallback for alerts when internet connectivity is unavailable, would enhance reliability in remote or low-signal areas, ensuring uninterrupted functionality. Incorporating artificial intelligence to automatically detect emergencies—such as falls or car crashes—could transform the app into a proactive safety tool, reducing reliance on manual activation. Finally, expanding globally by introducing multi-language support and adapting to region-specific emergency numbers would make the app a versatile solution for users worldwide, amplifying its life-saving potential.

8. Concerns/Queries/Doubts if any