

1.Functional Requirements:

➤ User Management Service:

- The system shall allow app user to view alerts, profile, accidents reports, and report a new incident.
- User with the role admin can review the reported new incident and approves the reported incident.

• Endpoints:

▪ User Endpoints:

- POST /api/users/register
- POST /api/users/login
- GET /api/users/viewDashboard
- GET /api/users/viewProfile
- GET /api/users/viewAlerts
- GET /api/users/viewAccidentReports
- POST /api/users/postAnIncident
- POST /api/users/sendAlertToResponders
- PUT /api/users/updateProfile

▪ Admin Endpoints

- POST /api/admin/login
- PUT /api/admin/reviewReportedIncident
- POST /api/admin/ForwardIncidentToResponders

➤ Alert Service:

- The system shall send real-time alerts to users regarding natural disasters and man- made accidents.

- Users shall receive alerts through the application interface whenever he wants to know weather conditions in a locality regarding current emergencies.

Endpoints:

- GET /api/alerts/getAlerts
- PUT /api/alerts/markAsRead

➤ **SOS Service:**

- The system shall provide an SOS button for users to request immediate assistance during emergencies.
- Users must be able to activate the SOS button with a single tap/click, triggering an immediate alert to local emergency services and displaying the user's location.
- The system shall provide feedback confirming that the SOS alert has been sent.

Endpoints:

- POST /api/sos/SendAlertsToResponders

➤ **Location Service:**

- The system shall provide location-based information for nearby emergency services during incidents.

Endpoints:

- GET /api/location/getLocation
- POST /api/location/setLocation

➤ **Helpline Service:**

- The system shall display the user's current location and nearby hospitals, police stations, and other emergency services.

Endpoints:

-GET /api/helpline/getHelplineByLocation/

-GET /api/helpline/getHelplineByService/

2.Non-Functional Requirements:

➤ Performance:

- **Criteria:** Fast loading times, especially critical during emergencies. - Efficient geo-tracing to minimize delays in displaying location-based information.
- **Purpose:** To ensure users can access necessary information and functionalities swiftly when needed most.

➤ Scalability:

- **Criteria:** Ability to handle many concurrent users, particularly during widespread emergencies.
- **Implementation:** Design the application architecture to support horizontal and vertical.

➤ Security:

- **Criteria:** Protect sensitive user data (e.g., names, mobile numbers, incident descriptions) through encryption and secure transmission protocols (e.g., HTTPS).
- Implement robust authentication and authorization mechanisms, especially for the admin module.
- Safeguard against common web vulnerabilities such as Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF).
- **Purpose:** To ensure data privacy, maintain user trust, and protect the application from malicious attacks.

➤ Reliability and Availability:

- **Criteria:** Ensure high uptime (e.g., 99.9%) to guarantee that the application is accessible during emergencies.
- Implement failover mechanisms and redundant systems to minimize downtime.
- **Purpose:** To provide dependable access to critical information when it's most needed.

➤ **Usability:**

- **Criteria:** Design an intuitive and user-friendly interface that can be easily navigated, even under stress. Ensure that forms and interactive elements are straightforward and accessible.
- **Purpose:** To facilitate quick and effective use of the application by individuals in emergency situations.
-

➤ **Compatibility:**

- **Criteria:** Ensure the application functions correctly across various web browsers (e.g., Chrome, Firefox, Safari, Edge) and devices (desktops, tablets, smartphones).
- Optimize responsive design to cater to different screen sizes and resolutions.
- **Purpose:** To provide a consistent experience to all users, regardless of their device or browser choice.

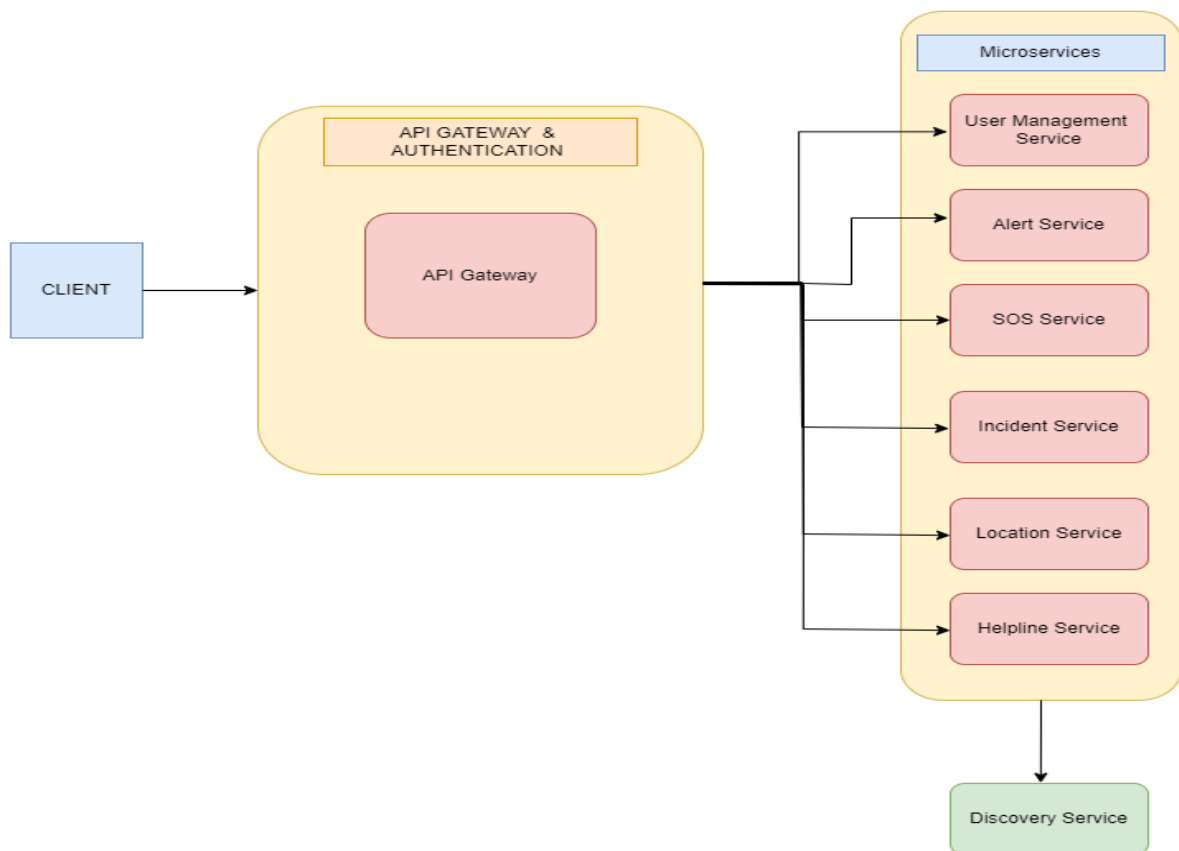
➤ **Accessibility:**

- **Criteria:** Adhere to accessibility standards (e.g., WCAG) to make the application usable by individuals with disabilities.
- Include features such as screen reader compatibility, keyboard navigation, and sufficient contrast ratios.
- **Purpose:** To ensure that the application is inclusive and usable by as many people as possible.

➤ **Technologies**

- Frontend - Angular
- Backend - Spring-boot
- Database - MySQL, MongoDB.

➤ Architecture Diagram:



➤ Database Schema Diagram:

