

Application Requirements Specification
For
<Rescue Track System>

Prepared by

Saloni

Nishant

Surya

Index

No	Title	Page no.s
1	<u>Purpose and Scope Statement</u>	3
2	<u>Requirements Overview</u>	4-7
3	<u>Objectives</u>	8
4	<u>Functional Specification</u>	9-12
5	<u>Logic Specification</u>	13

1. Purpose and Scope Statement

The Rescue Track System aims to enhance emergency response time and improve coordination between various emergency departments. By streamlining workflows, it minimizes delays, allowing quicker interventions during critical incidents. The system supports rapid emergency response deployment, optimizing rescue efforts and improving survival chances in life-threatening situations.

In emergency situations, particularly those involving life-threatening conditions like cardiac arrest, existing response times of around 10 minutes are often too slow to ensure survival. Immediate medical and rescue interventions within the first few minutes are critical. Delays can lead to severe complications or loss of life. There is a need for a comprehensive emergency response system that minimizes response times by improving communication, enhancing coordination among emergency services, deploying rescue resources efficiently, and utilizing innovative technologies like drones. This project seeks to bridge this critical time gap and ensure timely interventions during emergencies.

2. Requirements Overview

2.1 User Authentication (Login Page) Functionality: The system requires a secure login page for operators to access the system. Users can only make requests or manage operations once they authenticate successfully.

Modules: User Authentication Module: Handles login, logout, registration, and role based access control.

- Classes and Objects: User
- Attributes: username, password, role, lastLoginTime
- Methods: login(), logout(), register(), changePassword() AuthManager
- Methods: authenticate(user), authorize(user, role), generateToken()
- Libraries: **javax.security.auth** for authentication. **Spring Security** for role-based access control.

2.2 Emergency Request Assistance (Call System) Functionality: Users (victims) can request immediate assistance. Depending on the type of emergency, the system automatically connects to the relevant department for rapid deployment of services.

Categories for Emergency Requests:

- Heart Attack / Disease: Directed to Ambulance
- Safety (e.g., crimes or accidents): Directed to Police
- Civic Issues (e.g., potholes, drainage): Directed to PWD Department
- Fire: Directed to Fire Station

For each emergency, the following details are recorded:

- Name
- Phone
- Type of Incident
- Brief Description
- Department Transferred to
- Average Handling Time
- TAT of Response
- Officer Involved

Modules:

- Call Handling Module: Connects emergency requests to the relevant departments based on the incident type.
- Classes and Objects: EmergencyRequest
- Attributes: requestId, callerInfo, location, status, departmentAssigned, handlingTime, tat, officer
- Methods: initiateCall(), transferCall(), recordDetails(), completeCall() , CallManager
- Methods: connectCall(request), monitorCall(), assignDepartment(request)

2.3 Nearby Emergency services Integration Functionality: The system displays nearby emergency services departments based on the user's location to optimize response times.

- Modules: Location Services Module: Calculates the user's location and identifies nearby emergency departments.
- Classes and Objects: Department
 - Attributes: departmentId, name, location, contactInfo
 - Methods: getNearbyDepartments(location) LocationService
 - Methods: calculateDistance(), findClosestDepartments()
- Libraries: **java.util.Geolocation** for location services. **Spring Data JPA** for data access

2.4. Emergency Response Coordination Functionality: Ensures efficient coordination between various responders and emergency departments to minimize response times.

- Modules: Coordination Module: Manages communication between different responders (ambulance, police, fire stations, etc.).
- Classes and Objects: Responder
 - Attributes: responderId, type, availability
 - Methods: dispatchResponder(), updateStatus() ResponseManager
 - Methods: coordinateResponders(), trackProgress()
- Libraries: **java.util.concurrent** for multithreaded management.

2.5 System Administration and Management Functionality: Administrators can monitor the system, manage resources (e.g., drones, responders), and ensure smooth operation of the Rescue Track System.

- Modules: Admin Dashboard: Provides system monitoring, resource management, and performance tracking.
- Classes and Objects: Admin
 - Attributes: adminId, permissions
 - Methods: manageUsers(), manageResources() SystemMonitor
 - Methods: viewLogs(), monitorPerformance()
- Libraries: **Spring Boot Actuator** for health and performance monitoring.

2.6. Reporting System Functionality: The system generates detailed reports for administrators to assess the performance of the emergency response operations.

- Modules: Reporting Module: Handles incident reporting and generates summaries of the operations.
- Classes and Objects: Report
 - Attributes: reportId, content, dateGenerated
 - Methods: generateReport(), exportReport() ReportManager
 - Methods: createSummaryReport(), viewDetailedLogs()
- Libraries: **JasperReports** for report generation. **java.time** for date and time management.

Summary of Technology used:

User Authentication Module:

- **javax.security.auth**: For authentication.
- **Spring Security**: For role-based access control.

Nearby Emergency Services Integration:

- **java.util.Geolocation**: For location services (you may need a specific library for geolocation).
- **Spring Data JPA**: For data access to the database.

Emergency Response Coordination:

- **java.util.concurrent**: For managing multithreaded operations.

System Administration and Management:

- **Spring Boot Actuator**: For health and performance monitoring.

Reporting System:

- **JasperReports**: For generating reports.
- **java.time**: For date and time management.
- **Spring websocket**, and **spring messaging** for WebSocket Communication and real time messaging.

Database Design Summary

The RescueTrack System will utilize a relational database with four primary tables to manage emergency response operations:

- **EmergencyRequest**: Central table storing all emergency incidents with details like caller information, incident type, location, and status.
- **Department**: Maintains information about emergency response departments including medical, police, fire, and civic authorities.
- **Officer**: Tracks all emergency responders, their availability, specialization, and department affiliation.
- **RequestLog**: Maintains a comprehensive audit trail of all status changes and updates to emergency requests.

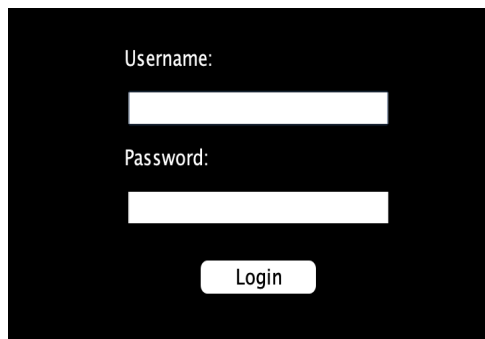
Key features :

- Enumerated types for standardized categorization of incidents, departments, and officer specializations
- Built-in tracking of response times and handling durations
- Complete audit trail capabilities for compliance and performance analysis
- Optimized indexing for high-performance emergency response coordination

3. Objectives

The primary objective of this project is to enhance emergency response efficiency and reduce critical response times to save lives during emergencies. By creating a comprehensive emergency assistance system, this project aims to ensure rapid coordination and deployment of emergency services. The application facilitates immediate help requests, coordinates between nearby emergency departments, manages critical drone activities, and oversees system operations. By providing real-time assistance and optimized emergency workflows, this solution addresses life-threatening situations faster, ensuring that victims receive timely medical and rescue assistance. This approach focuses on minimizing delays, particularly in critical cases where every second counts, thereby significantly improving survival rates and outcomes.

Below is the basic structure and expected login pages from the application



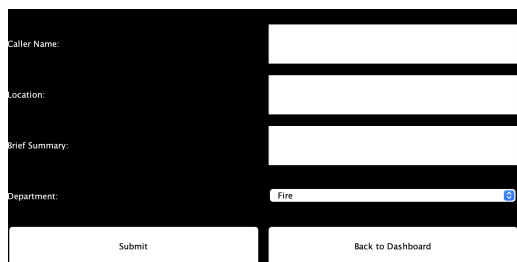
A login form with a black background. It features two white input fields for 'Username:' and 'Password:'. Below the password field is a white 'Login' button.

Login Page



A dashboard screen with a light gray background. It contains two large white rectangular buttons. The top button is labeled 'Manage Emergency Requests' and the bottom button is labeled 'Generate Reports'.

Dashboard Screen



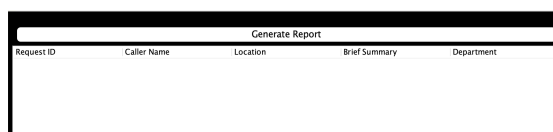
An emergency request form with a black background. It has four white input fields for 'Caller Name:', 'Location:', 'Brief Summary:', and 'Department:'. The 'Department:' field has a dropdown menu showing 'Fire'. At the bottom are two white buttons: 'Submit' and 'Back to Dashboard'.

Emergency Request Screen



A confirmation screen with a light gray background. It features a blue icon of a flame and a blue button labeled 'OK'. The text reads: 'Emergency Request Submitted with ID: 4c745acc-92a6-48c4-b028-cc4bc7a1c269'.

Emergency Request Created



A reporting screen with a black background. It has a white header bar labeled 'Generate Report'. Below the header is a table with columns: 'Request ID', 'Caller Name', 'Location', 'Brief Summary', and 'Department'.

Reporting Screen

4. Functional Specification

1. User Authentication and Access Control

Functionality:

- Provide secure user authentication for accessing the system.
- Support user registration and role-based access control.
- Supporting Programs / Modules:
- User Authentication Module (handles login, logout, and role management).

Classes and Objects:

- User
- Attributes: username, password, role, lastLoginTime
- Methods: *login()*, *logout()*, *register()*, *changePassword()*
- AuthManager

Methods:

- *authenticate(user)*, *authorize(user, role)*, *generateToken()*

Java Libraries:

- `javax.security.auth` for authentication.

Spring Security for role-based access control.

2. Emergency Request Assistance (Call System)

Functionality:

- Allow users to make or receive calls for help.
- Initiate communication between victims and emergency responders.
- Supporting Programs / Modules:

Call Handling Module (connects requests to emergency services).

Classes and Objects:

EmergencyRequest

- Attributes: *requestId*, *callerInfo*, *location*, *status*
- Methods: *initiateCall()*, *terminateCall()*

CallManager

- Methods: *connectCall(request)*, *monitorCall()*

Java Libraries:

- java.net for network communications.
- WebSocket APIs for real-time communication.

3. Nearby 911 Departments Integration

Functionality:

- Locate and display nearby 911 departments based on user location.
- Facilitate coordination with appropriate emergency responders.

Supporting Programs / Modules:

- Location Services Module (determines user location and proximity to services).

Classes and Objects:

- Department
- Attributes: *departmentId*, *name*, *location*, *contactInfo*
- Methods: *getNearbyDepartments(location)*

LocationService

- Methods: *calculateDistance()*, *findClosestDepartments()*

Java Libraries:

- java.util.Geolocation for handling location-based services.
- Spring Data JPA for data access.

4. Emergency Response Coordination

Functionality:

- Ensure coordinated responses among various emergency services.
- Supporting Programs / Modules:
- Coordination Module (manages interactions between responders).

Classes and Objects:

- Responder
- Attributes: *responderId*, *type*, *availability*

Methods:

- *dispatchResponder()*, *updateStatus()*

ResponseManager

Methods: *coordinateResponders()*, *trackProgress()*

Java Libraries:

- `java.util.concurrent` for multithreaded response management.

5. System Administration and Management

Functionality:

- Oversee system functions, manage resources, and monitor operations.

Supporting Programs / Modules:

- Admin Dashboard (provides system oversight).
- Classes and Objects:

Admin

- Attributes: *adminId*, *permissions*
- Methods: *manageUsers()*, *manageResources()*

SystemMonitor

- Methods: *viewLogs()*, *monitorPerformance()*

Java Libraries:

- Spring Boot Actuator for system health monitoring.

6. Reporting System

Functionality:

- Generate detailed reports of emergency requests, responses, and system metrics.

Supporting Programs / Modules:

- Reporting Module (handles report generation).

Classes and Objects:

Report

- Attributes: *reportId*, *content*, *dateGenerated*
- Methods: *generateReport()*, *exportReport()*

ReportManager

- Methods: *createSummaryReport()*, *viewDetailedLogs()*

Java Libraries:

- JasperReports for report generation.
- java.time for date and time management.

5. Logic Specification

