Table of Contents

Chapter 1

Introduction

1.1 Abstract

The **Maha Link** Flood Response App is designed to assist rescue teams and volunteer organizations in reaching areas where aid is urgently needed, ensuring efficient coordination and minimizing redundancy. The app provides real-time alerts and predictions, helping users stay informed of potential flood risks before they occur. Key data sources include nearby weather stations, dam administrators, and the app's own weather prediction mechanisms. The app aims to streamline disaster response efforts, maximize resource deployment, and safeguard lives by ensuring timely and accurate information flow. This document outlines the system's objectives, requirements, and design principles, with a focus on reliability, scalability, and usability.

1.2 Objectives

The overall objectives of the project are as follows:

1. To develop an effective and intuitive flood response system that improves the coordination of rescue operations.
2. To gather and document both functional and nonfunctional requirements for the app.
3. To model real-time interaction, data collection, and user engagement requirements.
4. To incorporate data from various sources, such as weather stations and dam authorities, for timely alerts and disaster management.

1.3 Project Description

The Maha Link project is grounded in the principles of agile software development, following an iterative approach to refine the system's design and functionality. The app integrates weather data and dam information from various trusted sources to provide a comprehensive flood risk assessment and alert mechanism.

The project began by identifying shortcomings in existing disaster response systems and outlining the key functionalities needed for an effective solution. Collaborative discussions with rescue teams and disaster management organizations shaped the app's requirements, which were then detailed in a Software Requirements Specification (SRS) document. Using this SRS, the system requirements were modeled through scenario-based planning, with subsequent design focusing on real-time alerts, user interaction, and data synchronization.

Chapter 2
Statement of Problems

The development of the **Maha Link** Flood Response App was driven by the need to address challenges faced by communities during flood emergencies. This chapter outlines the key issues identified, focusing on the real-life impact on affected regions and how **Maha Link** aims to offer a solution. The app is built with a focus on improving disaster response efforts, with the goal of saving lives and reducing damage in flood-prone areas.

2.1 Social Needs

- **Lack of real-time dam and weather data**: Communities and rescue teams often do not have access to real-time information on the status of nearby dams or weather conditions, leaving them vulnerable to unexpected floods.
- **Inability to identify high-priority areas**: During floods, rescue teams face difficulties in identifying which areas require immediate attention due to insufficient data on flood severity and affected locations.
- **Poor coordination between rescue efforts**: Multiple organizations and rescue teams often overlap or fail to cover all affected areas, leading to inefficient use of resources and unbalanced aid distribution.
- **Delayed alerts and warnings**: Existing flood alert systems are not timely or specific enough to provide communities and rescue teams with actionable information to take preventive measures before the flood hits.

2.2 Social Requirements

- **Real-time data integration**: The app must provide rescue teams and communities with up-to-date information from local weather stations and dam authorities, showing real-time flood risks.
- **Precise location-based alerts**: The system should deliver alerts and notifications specific to the user's geographical location, informing both citizens and rescue teams of imminent flood risks.
- **Rescue team coordination tools**: The app should allow rescue teams to communicate, plan, and manage their operations in real time to avoid redundant efforts and maximize resource distribution.
- **User-friendly navigation**: The app should feature an intuitive and simple interface, making it easy for users to receive alerts, view maps, and understand flood risks regardless of technical expertise.
- **Offline availability for critical information**: The app should offer offline functionality for critical features like rescue maps and safety instructions, ensuring access even in areas with poor internet connectivity.
- **Volunteer and resource tracking**: Rescue teams should be able to log and track available volunteers, resources, and equipment in real time to improve the allocation of these resources to the most affected areas.

2.3 Social Values

- **Improved disaster response and safety**: By providing real-time flood data and effective coordination tools, the app will enable rescue teams to reach the most affected areas quickly and efficiently, potentially saving lives.
- **Timely community preparedness**: The app will give communities enough time to evacuate or prepare for a flood, reducing the risk of loss and damage.
- **Better resource management**: By allowing rescue teams to coordinate their efforts, the app ensures that resources are used where they are needed most, avoiding duplication and increasing operational effectiveness.
- **Enhanced trust in flood response efforts**: The app will foster community trust in local authorities and rescue organizations by consistently delivering timely, accurate, and actionable information during floods.

2.4 Expected Solution

- **Real-time Flood Monitoring**: The app must integrate with weather stations and dam authorities to provide users with up-to-the-minute data on flood conditions and water levels.
- **Location-Based Alerts**: Alerts and notifications should be tailored to the user's specific location, ensuring that only the relevant population receives alerts about impending floods.
- **Rescue Coordination Dashboard**: The app should include a dashboard where rescue teams can see ongoing operations, track which areas are covered, and communicate to ensure that all affected locations are properly attended to.
- **Offline Critical Information**: Maps and safety instructions should be available offline, ensuring that users can access important data even in areas where internet connectivity is disrupted by the flood.
- **Resource and Volunteer Management**: The system should allow rescue teams to track available volunteers, equipment, and resources, helping to allocate them efficiently during operations.
- **User-Friendly Interface**: The app must have a clear, simple interface that allows users to easily access alerts, maps, and safety information, regardless of their technical expertise.
- **Real-Time Updates**: Rescue teams and users must be able to see real-time updates on which areas have been affected, which areas are being helped, and what resources are available for rescue operations.

Chapter 3
Software Requirement Specifications

In this chapter, the detailed SRS for **Maha Link** will be discussed in subsequent sections. These SRS results will then be used to guide the requirement modeling efforts described in later chapters. The ultimate goal is to create a reliable, scalable, and user-friendly system that improves the coordination and effectiveness of flood response operations for rescue teams and volunteers.

3.1 Purpose of System

The purpose of the system is to provide an integrated platform that monitors real-time weather conditions and manages donation activities to support villages affected by water-related disasters. The system will enable users to view up-to-date weather data for specific regions and track ongoing donation efforts, ensuring efficient allocation of resources and preventing duplicate donations to the same village. By offering accurate weather forecasts and transparent donation tracking, the system aims to enhance relief efforts and optimize the distribution of aid to the communities in need.

3.2 System Scope

Maha Link aims to develop a reliable and efficient platform that allows rescue teams and volunteer organizations to coordinate flood response operations. The system will focus on providing real-time data from weather stations and dam authorities, sending location-based alerts, and ensuring that rescue efforts are well-coordinated to avoid overlapping and missed areas.

The primary objectives are to:

- Improve disaster response efficiency: By providing real-time information and coordination tools, the system will help rescue teams and volunteers reach areas where help is most needed.
- Enhance community preparedness: The system will give local communities early warnings and actionable insights about potential flood risks, allowing them to take preventive measures.
- Facilitate better resource allocation: Through rescue team coordination, the system will ensure that volunteers, resources, and equipment are directed to the areas of greatest need.

This SRS outlines the functional and non-functional requirements necessary for the successful implementation of Maha Link, ensuring scalability, usability, and security across multiple platforms.

3.3 Environmental Characteristics

Maha Link will be accessible on web and mobile platforms (iOS, Android), allowing real-time access for rescue teams and volunteers. It will integrate cloud-based services to store and retrieve data from weather stations and dam authorities, ensuring up-to-date information.

The server infrastructure must be scalable to handle surges in traffic during emergencies, with platforms like AWS or Google Cloud. Regular maintenance and updates will ensure smooth functionality and security, with reliable integration of external data sources critical for timely alerts.

3.4 User Classes

1. **Rescue Teams**:
   - Primary users who use the system to locate areas in need of immediate assistance, view real-time flood data, and coordinate rescue efforts.
2. **Volunteers**:
   - Individuals or organizations providing support during the flood. They use the app to find nearby areas that require help and avoid overlap with other teams.
3. **Administrators**:
   - System managers responsible for overseeing the app's functionality, managing user permissions, and ensuring data from weather stations and dam administrators is accurate and up-to-date.
4. **Local Authorities**:
   - Government officials or dam administrators who provide critical data on water levels, dam conditions, and other key information related to flood warnings.
5. **General Public**:
   - Citizens who can access alerts and information about flood risks in their areas and request assistance if needed.

3.5 Operating Environment

The system will be designed to operate across multiple platforms, ensuring accessibility and functionality on both Android and iOS devices. The core components of the system, including the mobile application, database, and external APIs, will function within the following environments:

- Mobile Platforms:
  - Android: The application will support Android versions 9.0 (Pie) and above to ensure compatibility with a wide range of devices.
  - iOS: The application will support iOS versions 12.0 and above, targeting both iPhones and iPads.
- External Services:
  - Weather API: The system will rely on external weather data services, such as OpenWeatherMap or WeatherAPI, to provide real-time weather information.
  - Database and Backend: The application's backend, developed with FastAPI, will be hosted on cloud infrastructure with a scalable database system to store donation and village data.
- Internet Connectivity: A stable internet connection is required for users to access real-time weather updates, synchronize donation records, and interact with the backend services. The app will leverage HTTPS for secure data transmission.
- Hardware Requirements:
  - Android and iOS devices with at least 2GB of RAM and 100MB of available storage.
  - GPS functionality for location tracking and identifying the villages in need of aid.

The system is designed to be highly responsive and functional in environments where reliable internet connectivity is available, and it will be optimized to run efficiently on mobile devices across various hardware configurations.

3.6 System Constraints

- **Data Accuracy**: The system must rely on accurate, real-time data from weather stations and dam administrators to ensure reliable flood alerts and rescue coordination.

- **Real-time Updates**: Any changes in flood conditions, weather data, or rescue team status must be updated instantly across the platform to ensure timely decision-making.

- **Connectivity**: The system requires internet connectivity to receive and display real-time data. Limited offline functionality may exist, but critical features depend on active network access.

- **Scalability**: During flood events, the system must scale to handle sudden increases in users, including rescue teams, volunteers, and affected citizens.

- **Data Security**: Sensitive data, such as user information and location data, must be protected through encryption and secure protocols to ensure user privacy.

- **Device Compatibility**: The app must function reliably across multiple platforms (iOS, Android, and Web) and a range of device types (smartphones, tablets, and desktops).

- **Legal Compliance**: The system must comply with local government regulations regarding data collection, privacy, and emergency response protocols.

3.7 User Documentation

- **User Manual**: A comprehensive guide detailing how to install and use the application on both Android and iOS platforms.
- **Quick Start Guide**: A simplified version of the user manual that provides essential steps to get started quickly.
- **FAQs (Frequently Asked Questions)**: A list of commonly asked questions with concise answers.
- **In-App Help:** Built-in help features within the app that provide context-sensitive assistance (e.g., tooltips or an in-app FAQ section).

3.8 Functional Requirements

The system will operate based on functional requirements that define the specific actions and outputs of the platform. Each functional requirement will include a description, input, and expected output. These requirements are categorized as "R1," "R2," and so on, each representing an essential function.

- **R1: User Registration**
  - **Description**: The system must offer a simple and intuitive registration process.
  - **R1.1 Email Registration**: allows users to login with email
  - **R1.2 Social Media Login** :allows users to login with social media accounts

- **R2: Login**
  - **Description**: Registered users must log in to access the app's features.

- **R3: Organization Register**
    - **Description**: The system must offer a simple and intuitive organization registration process.

- **R4: Communication Between Users**
    - **Description**: The system must offer a fast and reliable way for effective communication between users

- **R5: Real Time Data**
    - **Description**: Users can create and manage playlists.
    - **R5.1 Real Time Weather Information:** Display accurate weather information.
    - **R5.2 Real Time Donation Information:** Display accurate volunteers and donation information.

- **R6: Feedback**
    - **Description**: Users can submit feedback and suggestions on the app.

3.9 Non-functional Requirements

3.9.1 Performance

- **Real-time Data Updates**: Alerts and notifications must be delivered within 1-2 seconds after data from weather stations or dam administrators is received.
- **Server Uptime**: The system should maintain an uptime of **99.9%**, especially during flood events.
- **Scalability**: The system must scale to handle **sudden increases in traffic**, including spikes in usage by volunteers and rescue teams during emergencies.

3.9.2 Usability

- **Intuitive Interface**: The system must offer simple and clear navigation, ensuring rescue teams and volunteers can use the app efficiently, even under stress.
- **Mobile Compatibility**: The design should be fully responsive, offering optimal functionality and usability across mobile devices, tablets, and desktops.

3.9.3 Security

- **User Privacy**: The system must protect user and location data in compliance with regional privacy laws.
- **Data Encryption**: Sensitive information, including personal and location data, must be encrypted to ensure security.

3.9.4 Concurrency

- **Concurrent Users**: The system must support hundreds of concurrent users, particularly during disaster events, without degradation in performance.

3.9.5 Scalability

- **System Growth**: The platform must be able to accommodate increasing numbers of rescue teams, organizations, and geographical data as the app expands to cover more regions.

3.9.6 Accessibility

- **Inclusive Design**: The system must be accessible to users with disabilities, supporting features like screen readers and alternative input methods for ease of use during emergency situations.

3.9.7 Interface Requirements

- **Home Page**: Displays current flood alerts, risk areas, and real-time updates.
- **Map Page**: Allows rescue teams and volunteers to search for affected areas, shelters, and resources
- **Rescue Operation Dashboard**: Provides an overview of rescue teams, their assigned locations, and their real-time status.
- **Notification Panel**: Sends real-time alerts on weather changes, dam levels, or areas needing immediate attention.

Chapter 4
Scenario-Based Modeling

After analyzing the system requirements, the process of requirement modeling is set to begin. This involves various modeling techniques, including scenario-based modeling, logical data modeling, conceptual data modeling, class-based modeling, and behavioral modeling. Scenario-based modeling lays the groundwork for subsequent models by breaking down the system into subsystems and illustrating user interactions. This chapter discusses the outcomes of scenario-based modeling, which will inform logical data modeling and behavioral modeling. The use cases described here will also be referenced in class-based modeling.

In this scenario-based model, the **Maha Link** system will be represented from the perspectives of distinct user roles and their interactions with the system. Based on the Software Requirements Specification (SRS), we will identify the general functionalities of the system, significant categories of users (or 'actors'), the tasks they perform (or 'use cases'), and the relationships between them. This will be followed by schematic representations using UML use case diagrams, along with detailed textual descriptions for each use case.

4.1 Actors

The 'actors' who will interact with the **Maha Link** system include:

- **User (Rescue Team Member/Volunteer)**: Individuals who utilize the application for flood response activities, such as seeking assistance, reporting conditions, and managing tasks.

4.2 User Profiles

From the SRS, the various tasks or ways the actors will interact with the system are defined as follows:

Actor 1: User (Rescue Team Member/Volunteer)

- **Sign Up**: Register for the application to gain access.
- **Log In**: Authenticate their account to use the system features.
- **Browse Alerts**: View ongoing flood alerts and real-time conditions in specific areas.
- **Report Conditions**: Submit updates on the ground situation, such as water levels and rescue needs.
- **Manage Tasks**: Access and manage assigned rescue tasks and operations.
- **Provide Feedback**: Share experiences and insights to improve system functionality.

4.3 Use Case Diagram

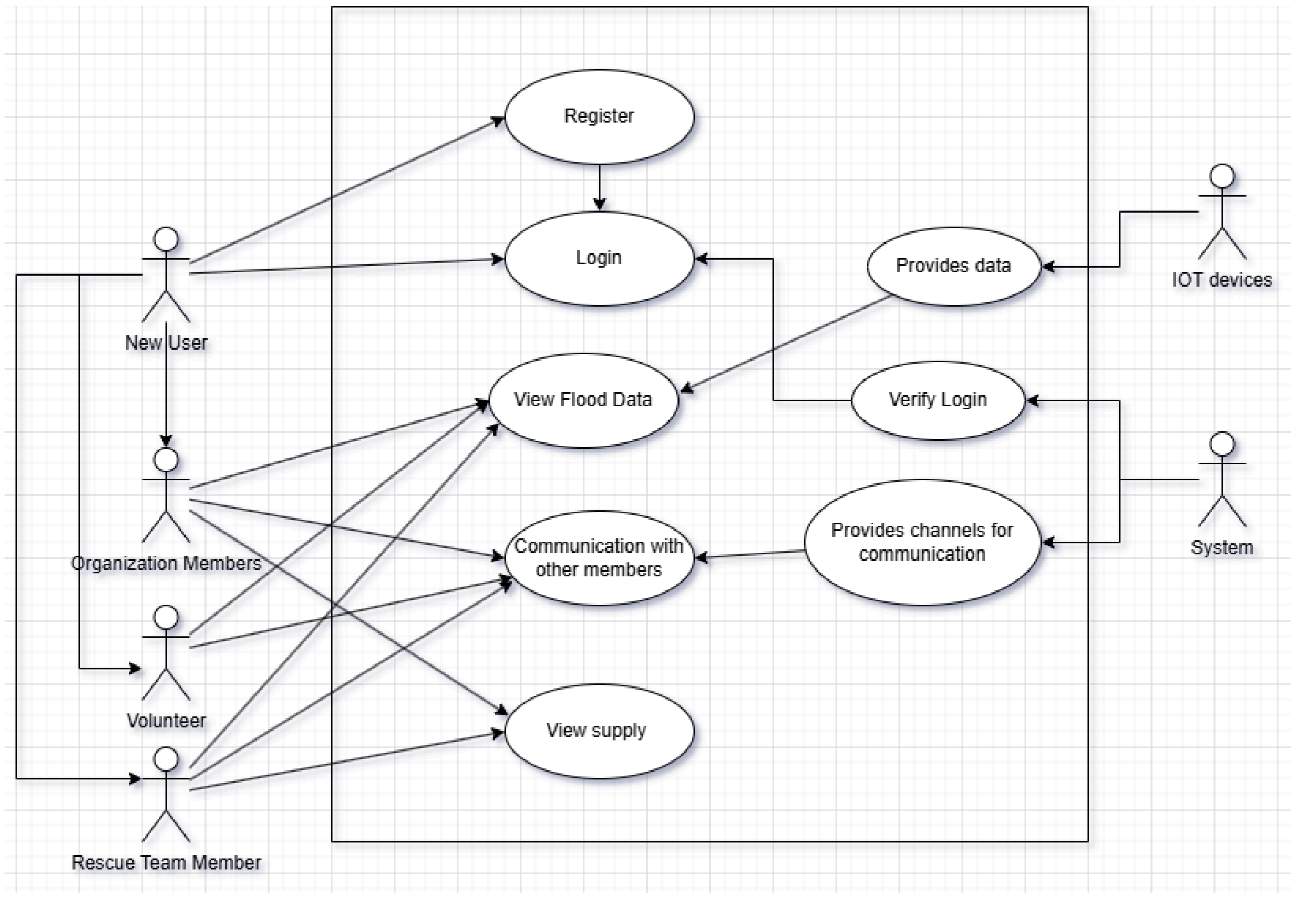


Figure 4.1: Use Case Diagram for Maha Link

4.4 Use Case Details

Use Case 1: Register to the System

Primary Actor: Rescue Team Member / Volunteer

Goal in Context: To allow rescue team members or volunteers to register for access to the flood response app and its features.

Preconditions: The user must have access to a device with internet connectivity.

Trigger: The user initiates the registration by selecting "Register" on the app's login page.

Scenario:

1. The user navigates to the app's login page.
2. The user selects the "Register" option.

3. The user enters required information (e.g., name, contact details, organization).
4. The user submits the registration form.
5. Upon successful registration, the user must enter the registered name and password to access the app.

Exceptions:

- If the user provides invalid contact information, the system will prompt for correction.

Priority: Essential
When Available: First increment
Channel to Actor: Via the app's login and registration interface.
Open Issues:

- Should users be allowed to register with duplicate phone numbers?
- Is there a security limit on the number of verification attempts?

Use Case 2: Access Real-Time Flood Data

Primary Actor: Rescue Team Member / Volunteer
Goal in Context: To enable users to access up-to-date flood information, including water levels and weather alerts.
Preconditions: The user must be logged in and have an internet connection.
Trigger: The user opens the flood data section of the app.
Scenario:

1. The user navigates to the flood data section.
2. The app retrieves and displays real-time flood data from weather stations and local authorities.
3. The user views key information like water levels, expected rainfall, and areas at high risk.
4. The user can choose to receive alerts when certain thresholds (e.g., rising water levels) are reached.

Exceptions:

- If data retrieval fails, the system notifies the user and allows them to retry.

- If the app cannot connect to data sources, an error message is shown.

Priority: Essential
When Available: First increment
Channel to Actor: Via the flood data section of the app.
Open Issues:

- How frequently should data be updated to maintain accuracy?
- Should historical flood data be accessible as well?

Use Case 3: Communicate with Organizations

Primary Actor: Rescue Coordinator
Goal in Context: Enable coordinators to send and receive updates from organizations actively involved in flood response.

Preconditions:

- The user must be logged in and assigned as a coordinator.

Trigger:

- The coordinator opens the communication feature to send or receive updates.

Scenario:

1. The coordinator navigates to the communication section.
2. The coordinator selects a specific organization to communicate with.
3. The coordinator sends real-time updates or requests (e.g., additional supplies, evacuation orders).
4. The organization receives the update instantly and responds as needed.
5. The system logs all communications for record-keeping.

Exceptions:

- If the organization is out of signal range, the system queues messages to be sent once connectivity is restored.
- If a message fails to be sent, the coordinator receives a notification to retry.

Priority: Essential

When Available: First increment

Channel to Actor: Through the app's communication section

Open Issues:

- Should communication logs be accessible to all team members?
- Is there a limit on message size to ensure faster delivery?

Use Case 4: Track and Manage Supplies

Primary Actor: Supply Coordinator

Goal in Context: To help coordinators keep track of available supplies and avoid duplication in distribution.

Preconditions: The user must be logged in and have the role of supply coordinator.

Trigger: The user accesses the supply management section.

Scenario:

1. The coordinator navigates to the supply management section.
2. The coordinator updates the list of supplies (e.g., adding new items or marking items as distributed).
3. The coordinator assigns supplies to specific rescue teams based on demand.
4. The system updates the overall supply count and alerts other coordinators to prevent overlapping distribution.

Exceptions:

- If two coordinators attempt to allocate the same supplies, the system alerts them to avoid duplication.
- If the app loses connection, changes are queued and synchronized once reconnected.

Priority: Essential

When Available: First increment

Channel to Actor: Via the supply management section of the app.

Open Issues:

- Should there be a notification system to alert teams of low supplies?
- How often should supply data be refreshed?

Use Case 5: Send Emergency Alerts to All Users

Primary Actor: System
Goal in Context: Automatically notify all users of critical flood conditions or evacuation notices, regardless of their location.
Preconditions: User must have enabled alert permissions.
Trigger: The system detects critical flood coordination (e.g., flood level exceeds threshold.)
Scenario:

1. The system continuously monitors real-time flood data.
2. When a critical threshold is met, the system prepares an alert for all users.
3. The system sends an alert to all users' devices.
4. The alert includes critical safety instructions and information on the nearest shelters in each user’s area (if available).
5. The system logs each alert and monitors user responses..

Exceptions:

- If users have disabled notifications, they won’t receive alerts.
- If a user is in an area with poor connectivity, the alert may be delayed until connectivity improves.

Priority: High
When Available: Second increment
Channel to Actor: Automatic push notifications via app and SMS.
Open Issues:

- Should users have the option to share their real-time location with coordinators?
- Is there a maximum number of alerts per hour to prevent overwhelming users?

Chapter 5
Conceptual Data Modeling

The ER diagram for our *Maha Link* flood response application illustrates the core entities and relationships necessary to streamline disaster management and rescue operations. Key entities include Administrators, Organizations, Rescue Teams, Affected Areas, Weather Stations, and Alert Systems. The diagram shows how administrators manage organizations and rescue teams, while each organization supports resources and missions within affected areas. Real-time data from weather stations helps the alert system to generate timely notifications for rescue teams, who then coordinate resources and volunteers for effective response. This structured data model ensures that *Maha Link* operates efficiently, supporting collaborative efforts in flood response and minimizing redundancy across rescue initiatives.

Here's the ER Diagram of our Maha Link App.

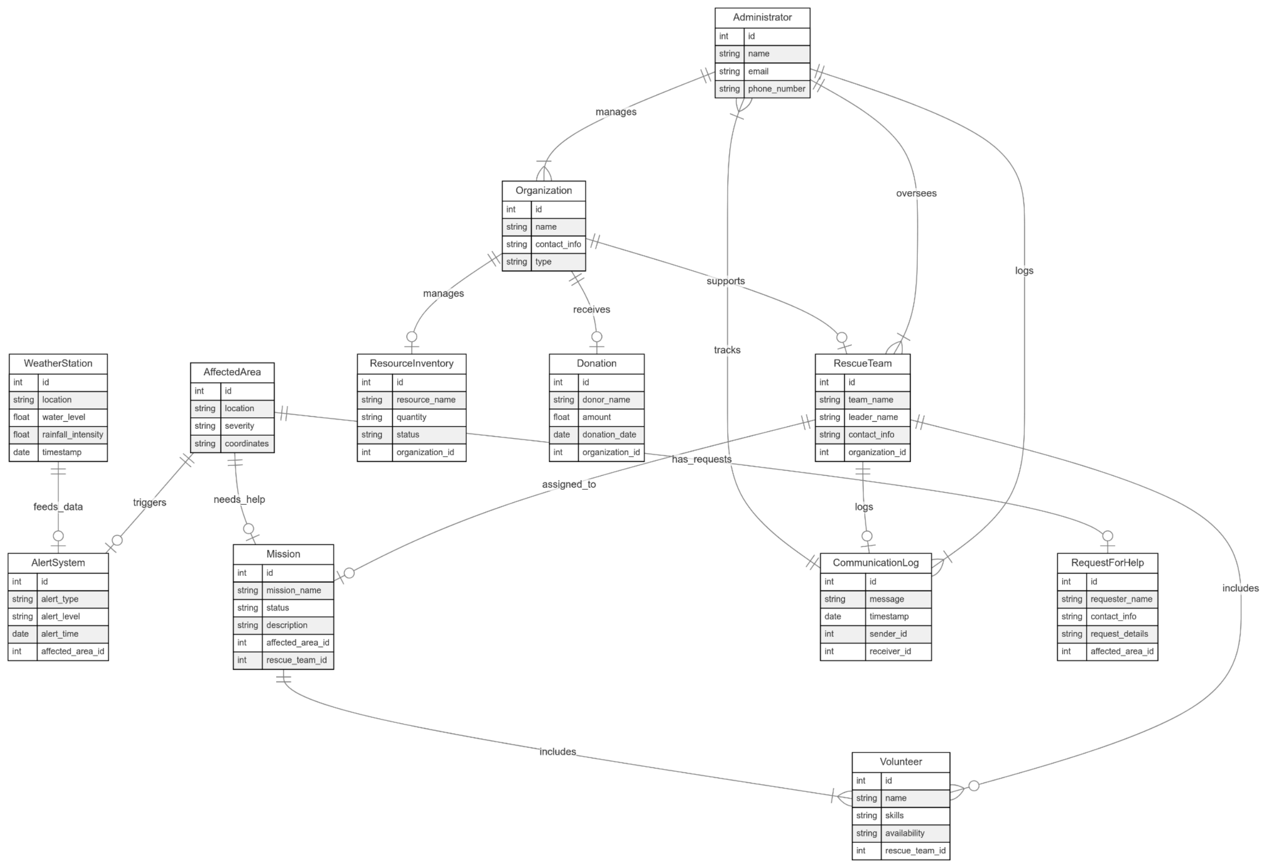
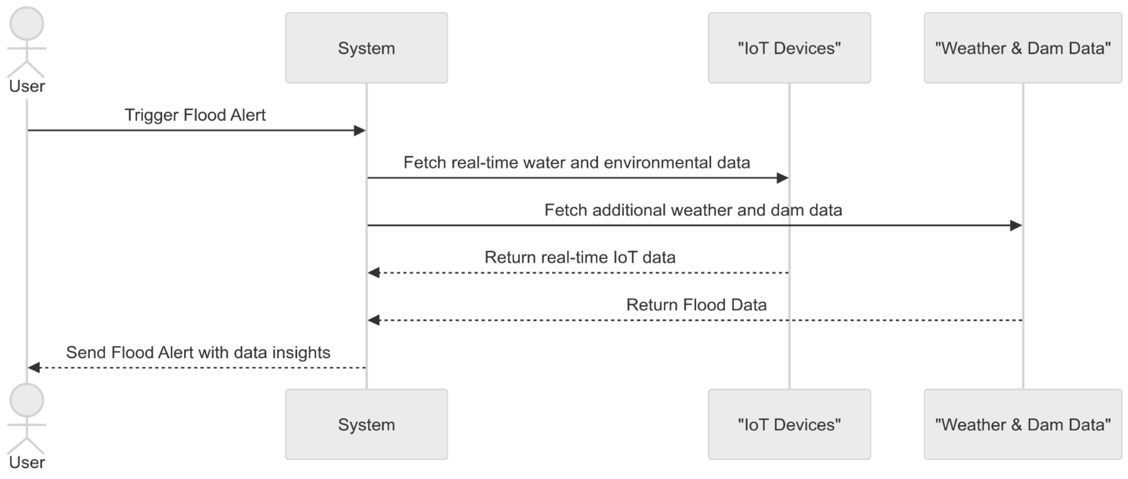


Figure 5.1: ER Diagram of Maha Link App

Chapter 6
Behavioral Modeling

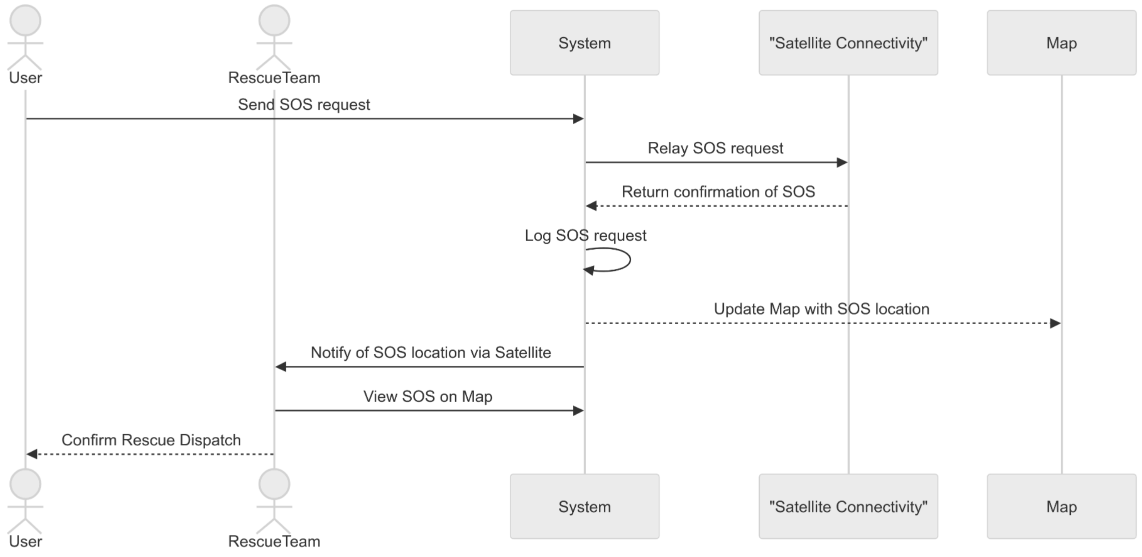
The sequence diagrams for **Maha Link** provide a comprehensive look at the flow of interactions between users, rescue and donation teams, and the system's core components. Each diagram highlights how the app operates across specific functions, such as flood alerts, SOS dispatch, donation coordination, and data logging. With real-time data from IoT devices and satellite connectivity for seamless communication, Maha Link offers reliable and up-to-date information for emergency response. These diagrams also underscore the system's efficient design, where each feature—whether mapping, alerting, or tracking—works collaboratively to support flood response efforts and ensure prompt, well-coordinated rescue and donation actions in critical situations.

1. Flood Alert Sequence Diagram with IoT Integration


User
System
"IoT Devices"
"Weather & Dam Data"
Trigger Flood Alert
Fetch real-time water and environmental data
Fetch additional weather and dam data
Return real-time IoT data
Return Flood Data
Send Flood Alert with data insights
User
System
"IoT Devices"
"Weather & Dam Data"

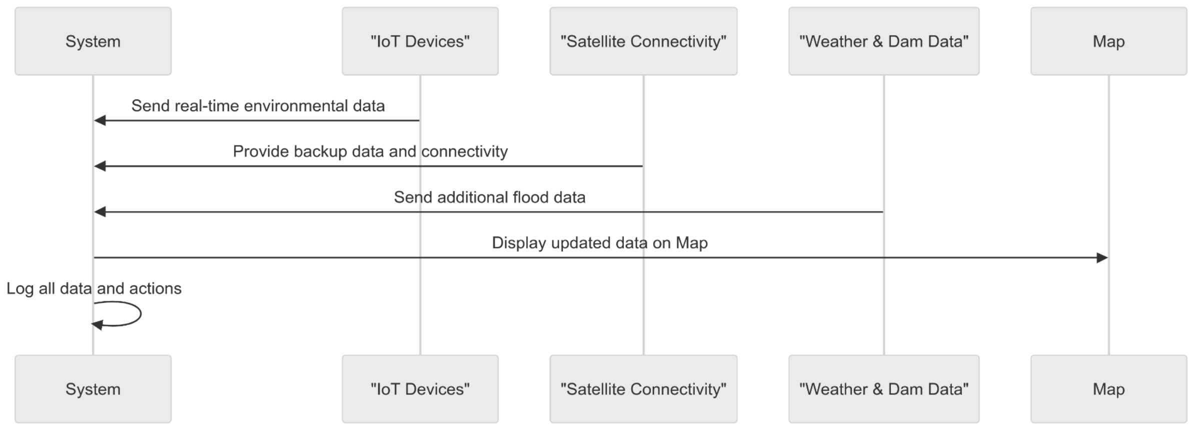
The User initiates the flood alert process. The System fetches data from IoT devices and DataSources, consolidating this information to send a detailed flood alert to the User. This feature ensures more accurate and up-to-date flood information.

2. SOS Request Sequence Diagram with Satellite Connectivity



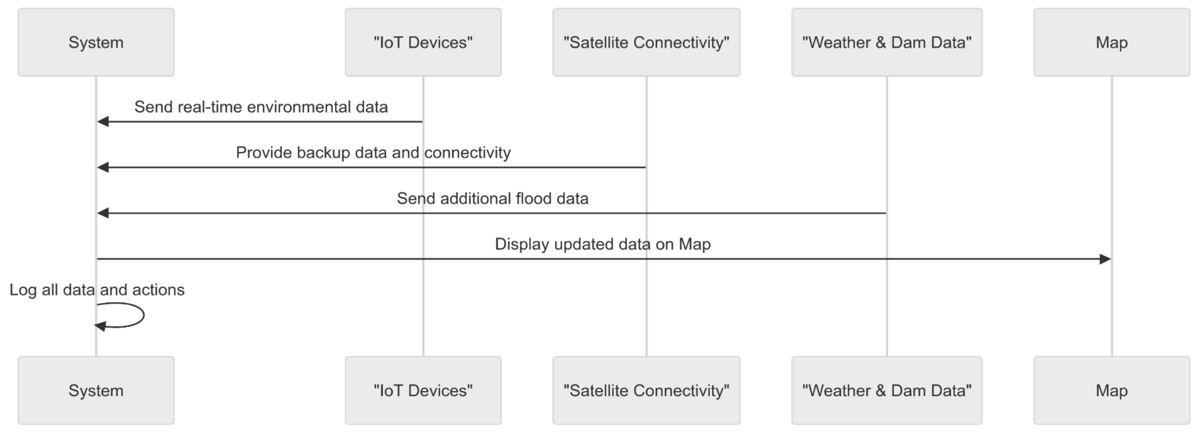
The User sends an SOS request, and if network issues arise, the System uses Satellite connectivity to ensure the message reaches the Rescue Team. The SOS location is logged and updated on the Map, enabling the Rescue Team to locate and assist the User reliably.

3. Donation Coordination with Map-Based Resource Allocation


System
"IoT Devices"
"Satellite Connectivity"
"Weather & Dam Data"
Map
Send real-time environmental data
Provide backup data and connectivity
Send additional flood data
Display updated data on Map
Log all data and actions

The System updates the Map with real-time flood and IoT data, notifying the Donation Team of critical areas. The Donation Team selects a target area, minimizing overlap and optimizing resource delivery. This sequence supports efficient and strategic donation response.

4. Full Data Logging with IoT and Satellite Data Sources


System
"IoT Devices"
"Satellite Connectivity"
"Weather & Dam Data"
Map
Send real-time environmental data
Provide backup data and connectivity
Send additional flood data
Display updated data on Map
Log all data and actions

The System gathers data from IoT Devices, Satellite, and DataSources to maintain a real-time log. The information is displayed on the Map and stored for analysis, ensuring accurate tracking and enhancing future response strategies.

Chapter 7
Future Plan

The *Maha Link* application is built to address current needs in flood response coordination, but we envision it evolving into a more comprehensive and proactive disaster management tool. Our future plans are focused on integrating advanced technologies and expanding the app's capabilities to better support rescue teams and communities affected by natural disasters. Below are our key focus areas for each year:

7.1 Timeline and Key Milestones

- **2025 - Expand User Base and Improve Features**
  Goal: Strengthen our foundational features and grow our user community. By August 2025, our focus will be on enhancing app usability and promoting the app to more organizations and communities.
- **2026 - IoT Integration**
  Goal: Implement IoT devices for real-time environmental monitoring. In September 2026, we plan to integrate IoT sensors to track water levels, rainfall, and other critical flood indicators. These devices will send live data to the app, allowing us to provide timely alerts and improve flood prediction accuracy.
- **2027 - Data-Driven Feature Enhancement**
  Goal: Improve features using insights from collected data. By October 2027, data gathered from IoT devices and user interactions will be analyzed to refine the app's functionalities, making the alerts and recommendations more reliable and location-specific.
- **2028 - Weather Prediction and Diversification**
  Goal: Develop an in-house weather prediction model. Starting in February 2028, we aim to incorporate AI-driven weather forecasting to provide accurate predictions of flood-prone areas. This step will diversify the app's capabilities, making it not only reactive but also predictive.
- **2029 - Market Expansion and Localization**
  Goal: Expand to new markets and adapt the app for diverse regions. In March 2029, we will begin adapting *Maha Link* to other geographic regions facing similar flooding challenges, tailoring the app for local conditions, languages, and infrastructure needs.

7.2 Implementing IoT and Weather Prediction

Although the current prototype does not include IoT functionality or weather prediction models, these features remain central to our future development. The integration of IoT devices and our own weather prediction models will transform *Maha Link* into a robust, data-driven platform. By 2028, our app will not only alert users about existing risks but will also predict potential flood scenarios, allowing for more proactive and effective flood management.

This roadmap will ensure that *Maha Link* continues to innovate, staying ahead of disaster management needs and making a meaningful impact in communities vulnerable to flooding.

Chapter 8
Conclusion

In conclusion, the design and development of *Maha Link* represent a significant step toward creating an effective flood response application for communities and rescue organizations. By utilizing data models and diagrams, such as the ER diagram, use case diagram, and context diagram, we've established a structured foundation for understanding the core components, relationships, and interactions within the system. These models have been instrumental in defining the primary entities—Administrators, Rescue Teams, Organizations, and Users—as well as their roles in maintaining and utilizing the platform.

The systematic design approach ensures that *Maha Link* will be both scalable and adaptable, capable of handling real-time updates and interactions critical in emergency scenarios. Although the current prototype is limited to basic functionality, our roadmap for future enhancements, including IoT integration and in-house weather predictions, demonstrates our commitment to continuous improvement and innovation. As we transition from design to implementation, the insights from these models will guide development, ensuring a user-friendly, reliable, and high-impact platform for flood management and response.