REPUBLIC OF CAMEROON

PEACE - WORK – FATHERLAND

MINISTRY OF HIGHER EDUCATION

REPUBLIQUE DU CAMEROUN

PAIX – TRAVAIL – PATRIE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

CEF440

INTERNET PROGRAMMING AND MOBILE PROGRAMMING

BY

GROUP 25

NAME	**MATRICULE**
FOFIE FOPA ELISABETH	FE21A189
MBISHU FABRICE YENVEN	FE21A232
EDI EDISON FORNANG	FE21A178
TAKOH CLOVERT NFUA	FE21A311
KIMBI CYRIL BONGNYU NFOR	FE21A216

COURSE INSTRUCTOR: DR NKEMINI Valery **MAY 2024**

TABLE OF CONTENT

SYSTEM MODELING AND DESIGN

1. INTRODUCTION

Disaster management systems are critical for minimizing the impact of natural and man-made disasters on human lives and property. These systems integrate various technologies and processes to effectively manage and respond to emergencies. Our disaster management systems are vital for preparing for, responding to, and recovering from emergencies such as natural disasters, industrial accidents, and terrorist attacks.

System modeling is a methodological approach used to create abstract representations of complex systems. These models help in understanding, analyzing, and designing systems by providing a clear and structured visualization of their components and interactions. In the context of disaster management, system modeling can be employed to simulate different scenarios, predict outcomes, and optimize response strategies.

Effective system design involves creating detailed models to visualize and specify the system's structure and behavior. Unified Modeling Language (UML) provides a standardized way to represent these models, ensuring clear communication and comprehensive documentation.

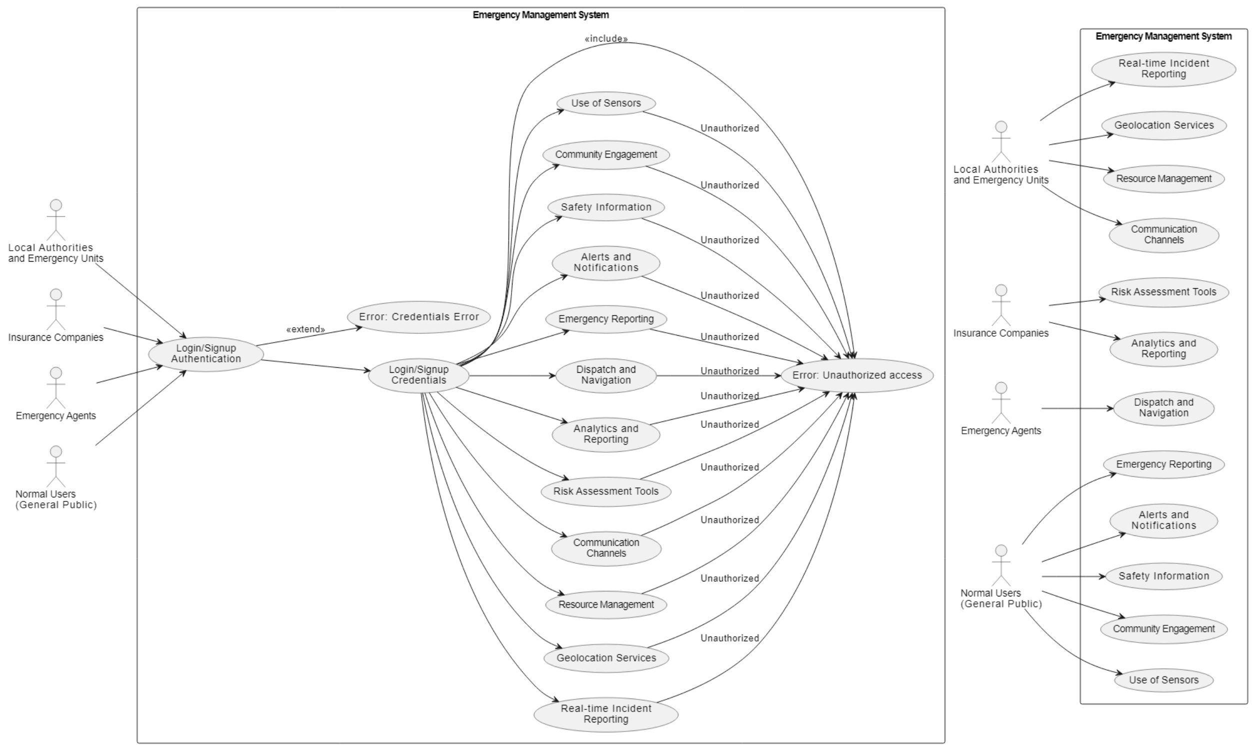
We shall have some key UML Diagrams for our Disaster Management System as follows

- Use Case Diagram
- Sequence Diagram
- Class Diagram
- Context Diagram
- Deployment Diagram

2. USE CASE DIAGRAM

This UML diagram outlines the structure and interactions of an Emergency Management System, featuring four main actors:

- **Local Authorities and Emergency Units:** These are government bodies and first responders who handle emergencies.
- **Insurance Companies:** Organizations that provide insurance coverage and need information for claims and assessments.
- **Emergency Agents:** Individuals or entities tasked with responding to emergencies, including private emergency response teams.
- **Normal Users (General Public):** General public who might use the system for reporting emergencies or accessing safety information.



The system encompasses a range of essential functionalities encapsulated within the "Emergency Management System" rectangle, including authentication, real-time incident reporting, geolocation services, resource management, communication channels, risk assessment tools, analytics and reporting, dispatch and navigation, emergency reporting, alerts and notifications, safety information, community engagement, and use of sensors. Interactions between actors and the system are facilitated through the authentication process, where all users must log in or sign up with their credentials. Access to system functionalities is then granted upon successful authentication, while unauthorized attempts trigger "Error: Unauthorized access" responses. Additionally, the system accounts for authentication failure due to incorrect credentials with the "Error: Credentials Error" branch, extending the authentication process. Through these relationships and error handling mechanisms, the diagram illustrates the robustness and security considerations inherent in the Emergency Management System's design.

2.1 USE CASES:

- **Login/Signup Authentication:** This use case allows all actors to log in or sign up to access the system.
- **Error: Credentials Error:** This use case is triggered when there is an error in login or signup credentials.
- **Login/Signup Credentials:** The process of entering and verifying login or signup details.
- **Error: Unauthorized Access:** This use case occurs when an actor attempts to access a part of the system they are not authorized to use.

2.2 USE CASES WITHIN THE EMS (EMERGENCY MANAGEMENT SYSTEM):

- **Use of Sensors:** Monitoring environmental sensors to detect potential emergencies.
- **Community Engagement:** Interacting with the community for preparedness and response.
- **Safety Information:** Providing information on safety measures and guidelines.
- **Alerts and Notifications:** Sending alerts and notifications about emergencies.
- **Emergency Reporting:** Reporting incidents and emergencies.
- **Dispatch and Navigation:** Coordinating and navigating emergency response units.
- **Analytics and Reporting:** Analyzing data and generating reports on emergencies and responses.
- **Risk Assessment Tools:** Tools to assess risk levels in different situations.
- **Communication Channels:** Facilitating communication among different actors during emergencies.
- **Resource Management:** Managing resources required for emergency responses.

- **Geolocation Services:** Providing location-based services and information.
- **Real-time Incident Reporting:** Reporting incidents in real-time for immediate action.

2.3 RELATIONSHIPS:

- **Extends Relationship:**

The "Login/Signup Authentication" use case is extended by the "Error: Credentials Error" use case, indicating that a credentials error might occur as part of the login/signup process.

- **Includes Relationship:**

The "Login/Signup Authentication" includes "Login/Signup Credentials", meaning that authentication cannot occur without entering credentials.

- **Generalization (Unauthorized Access):**

Each use case within the EMS that requires authorization has a relationship showing "Unauthorized" access leading to "Error: Unauthorized Access".

2.4 ACCESS AUTHORIZATION:

Each actor is shown to have access to specific use cases within the EMS:

- **Local Authorities and Emergency Units:** Have access to real-time incident reporting, geolocation services, resource management, communication channels, risk assessment tools, and analytics and reporting.
- **Insurance Companies:** Have access to risk assessment tools and analytics and reporting.
- **Emergency Agents:** Have access to dispatch and navigation, emergency reporting, alerts and notifications, and safety information.
- **Normal Users (General Public):** Have access to safety information, community engagement, and use of sensors.

Actors interact with the EMS primarily through the login/signup process, which determines their access to specific functionalities. Unauthorized attempts to access restricted areas result in an "Unauthorized Access" error.

3. SEQUENCE DIAGRAM

A sequence diagram for a disaster management system illustrates the interactions between users (actors) and the system components during a disaster event. It details the sequence of actions required to report, manage, and respond to disasters, highlighting the flow of information and the dependencies between various parts of the system.

3.1 SEQUENCE DIAGRAM CASE: NORMAL USER REPORTING AN EMERGENCY

Actors:

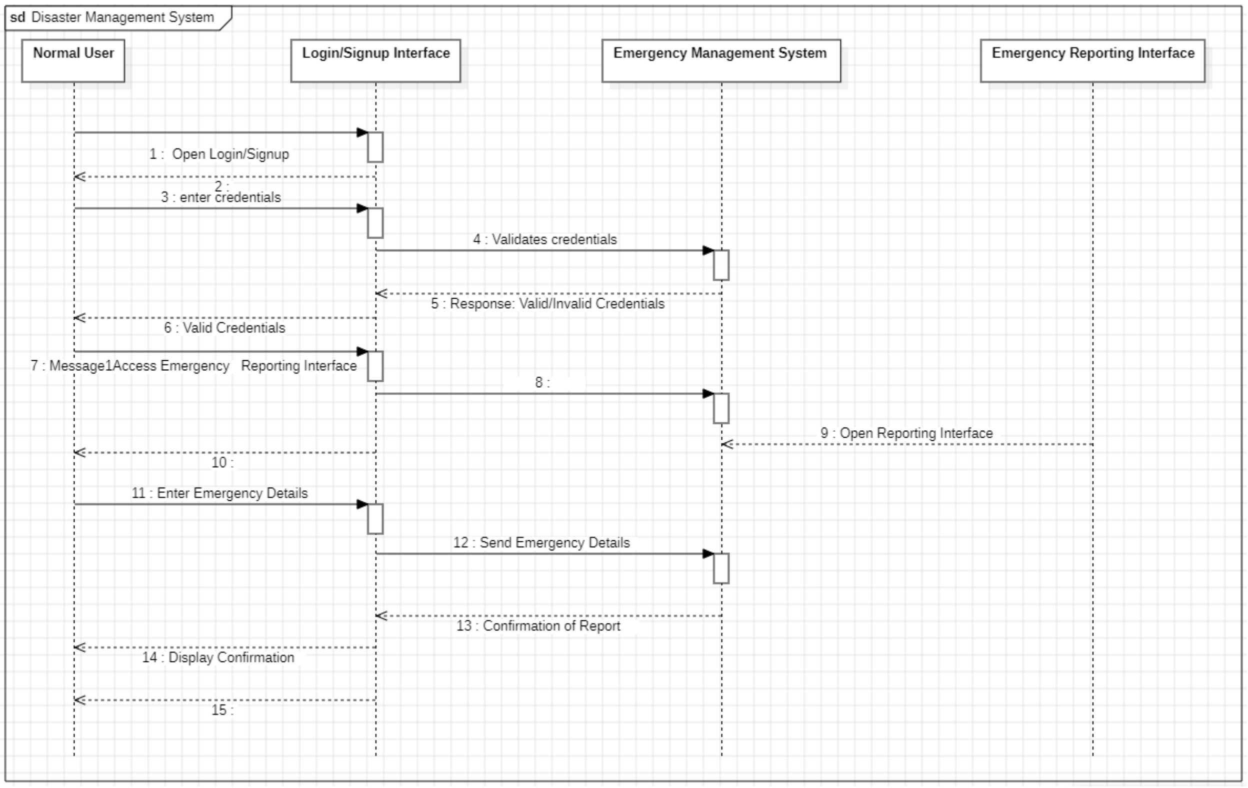
- Normal User (General Public)
- Emergency Management System

Objects:

- Login/Signup Interface
- Emergency Reporting Interface

For our sequence diagram, we have the following steps:

- Normal User accesses the system and initiates login.
- The system presents the Login/Signup Interface.
- Normal User enters login credentials.
- The Login/Signup Interface validates credentials with the system.
- If credentials are valid, the system grants access; if invalid, an error message is shown.
- Normal User accesses the Emergency Reporting Interface.
- Normal User enters emergency details.
- The Emergency Reporting Interface sends the emergency details to the system.
- The system processes the emergency report.
- The system sends confirmation of the report back to the Emergency Reporting Interface.
- The Emergency Reporting Interface displays confirmation to the Normal User.


sd Disaster Management System
Normal User
Login/Signup Interface
Emergency Management System
Emergency Reporting Interface
1 : Open Login/Signup
2 :
3 : enter credentials
4 : Validates credentials
5 : Response: Valid/Invalid Credentials
6 : Valid Credentials
7 : Message1Access Emergency Reporting Interface
8 :
9 : Open Reporting Interface
10 :
11 : Enter Emergency Details
12 : Send Emergency Details
13 : Confirmation of Report
14 : Display Confirmation
15 :

3.1.1 SEQUENCE OF INTERACTIONS:

- **Open Login/Signup Interface :**

The Normal User initiates the process by opening the Login/Signup Interface.

This action is represented by a solid line, indicating a synchronous call that requires immediate response.

- **Enter Credentials :**

The Normal User enters their login credentials (username and password) into the Login/Signup Interface.

This action is also represented by a solid line, as the system waits for the user to complete their input.

- **Validate Credentials :**

The Login/Signup Interface sends the entered credentials to the Emergency Management System for validation.

This interaction is synchronous, as the system must verify the credentials before proceeding.

- **Response: Valid/Invalid Credentials :**

The Emergency Management System validates the credentials and sends back a response indicating whether they are valid or invalid.

This is a return message indicated by a dashed line (synchronous), showing the result of the validation process.

- **Valid Credentials :**

Assuming the credentials are valid, the Login/Signup Interface confirms the successful login to the Normal User.

This interaction is depicted by a dashed line, indicating an asynchronous return message where the system updates the user about the successful validation.

- **Access Emergency Reporting Interface:**

The Normal User accesses the Emergency Reporting Interface after successfully logging in.

This action is synchronous, requiring immediate system feedback.

- **Open Reporting Interface :**

The Emergency Management System processes the request and opens the Emergency Reporting Interface.

This is indicated by a dashed line, showing an asynchronous return message where the system loads the reporting interface.

- **Enter Emergency Details :**

The Normal User enters the details of the emergency (e.g., type, location, severity) into the Emergency Reporting Interface.

This action is synchronous, as the user needs to input details to proceed.

- **Send Emergency Details:**

The Emergency Reporting Interface sends the entered emergency details to the Emergency Management System.

This interaction is synchronous, as the system needs to process the details immediately.

- **Confirmation of Report :**

The Emergency Management System processes the emergency details and sends a confirmation back to the Emergency Reporting Interface.

This is indicated by a dashed line, showing an asynchronous return message where the system confirms the successful submission of the emergency report.

- **Display Confirmation :**

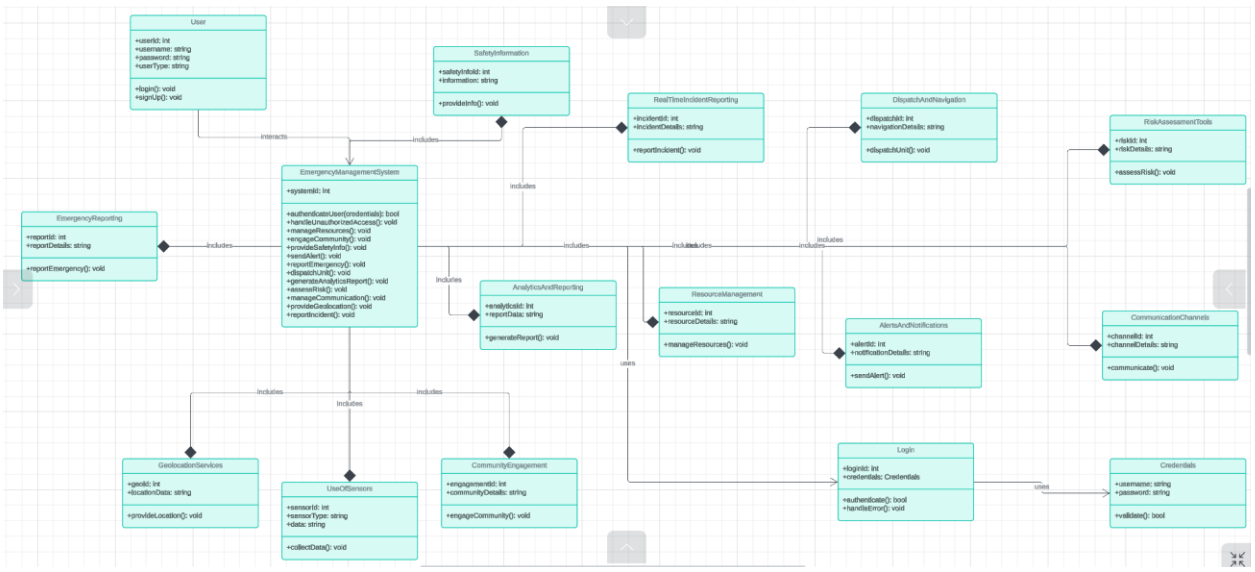
The Emergency Reporting Interface displays the confirmation message to the Normal User.

This is depicted by a dashed line, indicating an asynchronous return message confirming the report submission.

4. CLASS DIAGRAM

The class diagram provided illustrates the structure of a Disaster Management System, highlighting the system's classes, their attributes, methods, and relationships. Here is a detailed explanation of each element and its role in the system:

This class diagram outlines the structure and relationships within a Disaster Management System. Key components include user management, emergency reporting, real-time incident handling, resource management, and community engagement. Each class encapsulates specific attributes and methods relevant to its role, and the relationships between classes indicate how they collaborate to provide comprehensive disaster management functionality. The includes and uses relationships illustrate how the system's components integrate and interact, forming a cohesive and functional system.



4.1 CLASSES AND ATTRIBUTES

1. **USER**

Attributes:

userId: int - Unique identifier for the user.

username: string - Username for the user.

password: string - Password for user authentication.

userType: string - Type of user (e.g., Normal User, Emergency Agent, etc.).

Methods:

login(): void - Method for user login.

signUp(): void - Method for user registration.

2. **LOGIN**

Attributes:

loginId: int - Unique identifier for the login session.

credentials: Credentials - An instance of the Credentials class.

Methods:

authenticate(): bool - Method to authenticate user credentials.

handleError(): void - Method to handle login errors.

3. **CREDENTIALS**

Attributes:

username: string - Username for the user.

password: string - Password for user authentication.

Methods:

validate(): bool - Method to validate the provided credentials.

4. **EMERGENCYMANAGEMENTSYSTEM**

Attributes:

systemId: int - Unique identifier for the system.

Methods:

authenticateUser(credentials): bool - Method to authenticate user credentials.

handleUnauthorizedAccess(): void - Method to handle unauthorized access.

manageResources(): void - Method to manage emergency resources.

engageCommunity(): void - Method to engage the community.

provideSafetyInfo(): void - Method to provide safety information.

sendAlert(): void - Method to send alerts.

reportEmergency(): void - Method to report an emergency.

dispatchUnit(): void - Method to dispatch emergency units.

generateAnalyticsReport(): void - Method to generate analytics reports.

assessRisk(): void - Method to assess risks.

manageCommunication(): void - Method to manage communication channels.

provideGeolocation(): void - Method to provide geolocation data.

reportIncident(): void - Method to report incidents.

5. EMERGENCYREPORTING

Attributes:

reportId: int - Unique identifier for the emergency report.

reportDetails: string - Details of the emergency report.

Methods:

reportEmergency(): void - Method to report an emergency.

6. SAFETYINFORMATION

Attributes:

safetyInfoId: int - Unique identifier for the safety information.

information: string - Safety information details.

Methods:

provideInfo(): void - Method to provide safety information.

7. **REALTIMEINCIDENTREPORTING**

Attributes:

incidentId: int - Unique identifier for the incident report.

incidentDetails: string - Details of the incident.

Methods:

reportIncident(): void - Method to report an incident in real-time.

8. **DISPATCHANDNAVIGATION**

Attributes:

dispatchId: int - Unique identifier for the dispatch.

navigationDetails: string - Navigation details for dispatch.

Methods:

dispatchUnit(): void - Method to dispatch a unit.

9. **RISKASSESSMENTTOOLS**

Attributes:

riskId: int - Unique identifier for the risk assessment.

riskDetails: string - Details of the risk assessment.

Methods:

assessRisk(): void - Method to assess risks.

10. **ANALYTICSANDREPORTING**

Attributes:

analyticsId: int - Unique identifier for the analytics report.

reportData: string - Data of the analytics report.

Methods:

generateReport(): void - Method to generate a report.

11. RESOURCEMANAGEMENT

Attributes:

resourceId: int - Unique identifier for the resource.

resourceDetails: string - Details of the resource.

Methods:

manageResources(): void - Method to manage resources.

12. ALERTSANDNOTIFICATIONS

Attributes:

alertId: int - Unique identifier for the alert.

notificationDetails: string - Details of the notification.

Methods:

sendAlert(): void - Method to send alerts.

13. COMMUNICATIONCHANNELS

Attributes:

channelId: int - Unique identifier for the communication channel.

channelDetails: string - Details of the communication channel.

Methods:

communicate(): void - Method to communicate.

14. GEOLOCATIONSERVICES

Attributes:

geoId: int - Unique identifier for the geolocation service.

locationData: string - Geolocation data.

Methods:

provideLocation(): void - Method to provide location data.

15. USEOFSENSORS

Attributes:

sensorId: int - Unique identifier for the sensor.

sensorType: string - Type of the sensor.

data: string - Data collected by the sensor.

Methods:

collectData(): void - Method to collect data from sensors.

16. COMMUNITYENGAGEMENT

Attributes:

engagementId: int - Unique identifier for the community engagement.

communityDetails: string - Details of the community engagement.

Methods:

engageCommunity(): void - Method to engage the community.

4.2 RELATIONSHIPS

- **INTERACTION:**

User interacts with *EmergencyManagementSystem.*

- **INCLUDES:**

EmergencyManagementSystem includes multiple functionalities:

- *EmergencyReporting*
- *SafetyInformation*
- *RealTimeIncidentReporting*
- *DispatchAndNavigation*

- *RiskAssessmentTools*
- *AnalyticsAndReporting*
- *ResourceManagement*
- *AlertsAndNotifications*
- *GeolocationServices*
- *UseOfSensors*
- *CommunityEngagement*

- **USES:**

Login uses *Credentials* for authentication.

5. CONTEXT DIAGRAM

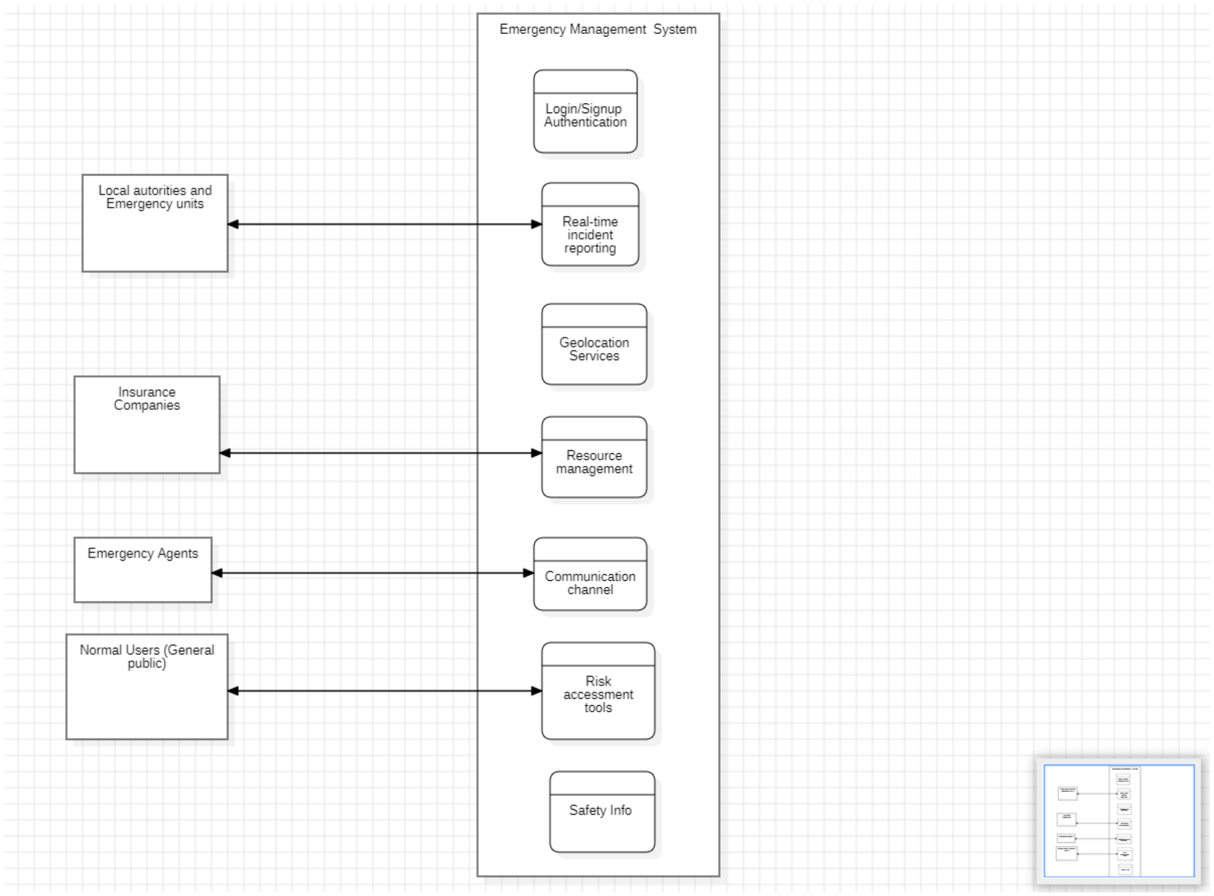
A context diagram provides a high-level overview of a system and its interactions with external entities. It typically shows the system as a single process and highlights its interactions with various external actors. Here's a context diagram for the Emergency Management System (EMS) based on the use case diagram provided.

5.1 DESCRIPTION:

The Emergency Management System (EMS) interacts with four main external entities:

- Local Authorities and Emergency Units
- Insurance Companies
- Emergency Agents
- Normal Users (General Public)

Each entity interacts with the EMS for specific functions such as logging in, reporting emergencies, accessing safety information, and other functionalities.


Emergency Management System
Login/Signup Authentication
Local autorities and Emergency units
Real-time incident reporting
Geolocation Services
Insurance Companies
Resource management
Emergency Agents
Communication channel
Normal Users (General public)
Risk accessment tools
Safety Info

- **Emergency Management System (EMS):**

This is the central system that interacts with all external entities. It handles login/signup authentication and provides various services such as real-time incident reporting, geolocation services, resource management, communication channels, risk assessment tools, and safety information.

- **Local Authorities and Emergency Units:**

These entities interact with the EMS for real-time incident reporting, geolocation services, resource management, and communication channels.

- **Insurance Companies:**

They use the EMS to access risk assessment tools and analytics and reporting for handling insurance claims and risk evaluations.

- **Emergency Agents:**

They interact with the EMS for dispatch and navigation, emergency reporting, alerts and notifications, and safety information.

- **Normal Users (General Public):**

These users interact with the EMS for accessing safety information, community engagement, and using sensors for emergency reporting.

5.2 INTERACTION DETAILS:

Each external entity is depicted as an actor outside the central system, represented by the EMS.

Arrows indicate the flow of information or services between the actors and the EMS.

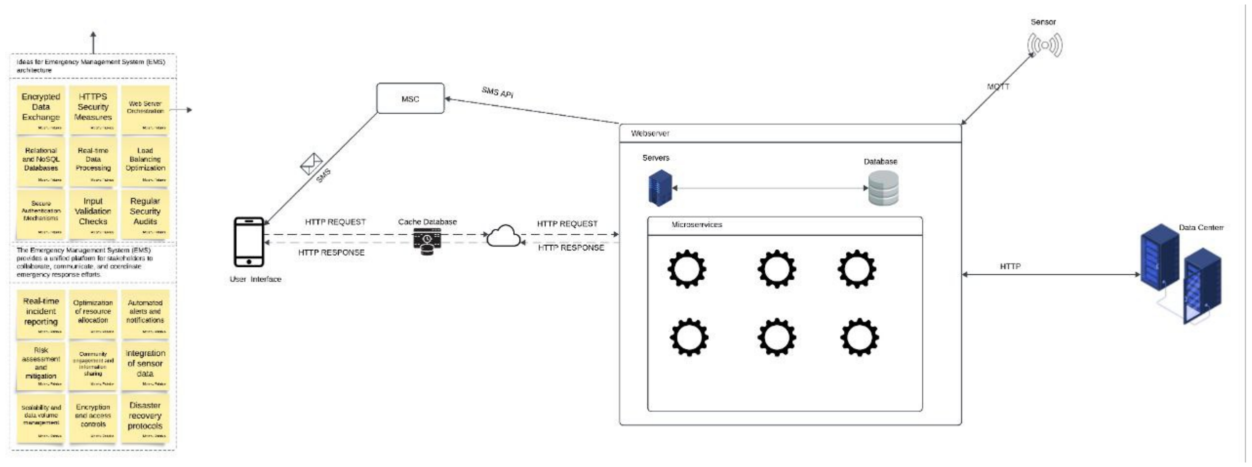
The EMS handles authentication through the login/signup process before providing access to its functionalities.

Specific services are available to specific entities based on their role and needs within the system.

This context diagram offers a simplified yet comprehensive view of how the EMS interacts with its external environment, encapsulating the high-level processes and their relationships with different actors.

6. DEPLOYMENT DIAGRAM

A deployment diagram in the context of a Disaster Management System represents the physical deployment of the software system's components across the hardware infrastructure. It shows how different pieces of software (nodes and artifacts) are mapped onto the physical hardware, network devices, and other components required for the system's operation. This diagram helps visualize the system's architecture and how components interact in a real-world environment.


Ideas for Emergency Management System (EMS) architecture
Encrypted Data Exchange
HTTPS Security Measures
Web Server Orchestration
Relational and NoSQL Databases
Real-time Data Processing
Load Balancing Optimization
Secure Authentication Mechanisms
Input Validation Checks
Regular Security Audits
The Emergency Management System (EMS) provides a unified platform for stakeholders to collaborate, communicate, and coordinate emergency response efforts.
Real-time incident reporting
Optimization of resource allocation
Automated alerts and notifications
Risk assessment and mitigation
Community engagement and information sharing
Integration of sensor data
Scalability and data volume management
Encryption and access controls
Disaster recovery protocols
MSC
SMS API
SMS
Sensor
MQTT
Webserver
Servers
Database
Microservices
User Interface
HTTP REQUEST
HTTP RESPONSE
Cache Database
HTTP REQUEST
HTTP RESPONSE
HTTP
Data Center

6.1 CENTRAL ARCHITECTURE

- **User Interface (UI):**

The user interface, accessible via mobile or web applications, is the primary point of interaction for users. Users can send HTTP requests to report incidents, request information, or receive updates. The UI then processes these requests and displays the corresponding HTTP responses.

- **Mobile switching Center (MSC):**

The MSC handles SMS communications, using an SMS API to send and receive alerts. This allows the EMS to notify users via text messages, ensuring that critical information reaches stakeholders even if internet access is unavailable.

- **Webserver:**

The webserver is a crucial component that manages all HTTP requests from the user interface and other system elements. It processes these requests and forwards them to the appropriate microservices or databases for further action.

- **Microservices:**

The architecture employs a microservices approach, where individual services handle specific tasks. This modular design enhances scalability and maintainability, allowing the system to efficiently manage different aspects of emergency response.

- **Database:**

The database stores all essential data for the EMS, using both relational and NoSQL databases to handle various data types and storage needs. This ensures that the system can manage structured and unstructured data effectively.

- **Cache Database:**

To improve performance, the EMS includes a cache database that provides temporary storage for frequently accessed data. This reduces the load on the main database and speeds up data retrieval.

6.2 COMMUNICATION WITH EXTERNAL COMPONENTS

- **Sensors:**

Sensors deployed in the field collect real-time data and communicate with the webserver using the MQTT (Message Queuing Telemetry Transport) protocol. This allows the system to receive immediate updates from the environment, which is critical for real-time monitoring and response.

- **Data Center:**

The EMS communicates with a data center via HTTP for large-scale data processing and storage. The data center supports extensive computational tasks and ensures that the system can handle significant volumes of data.

6.3 FLOW OF INFORMATION

The flow of information within the EMS is designed to be efficient and responsive:

- **User Interface to Webserver:**

Users interact with the system by sending HTTP requests through the user interface. These requests are processed by the webserver, which sends back the appropriate HTTP responses.

- **Webserver to Microservices and Database:**

The webserver routes requests to the relevant microservices for processing or retrieves data from the database as needed. This modular approach ensures that each service can operate independently and efficiently.

- **Webserver and Cache Database:**

Frequently accessed data is stored in the cache database, which the webserver can quickly retrieve. This enhances system performance by reducing the need to repeatedly access the main database.

- **Webserver to Data Center:**

For extensive data processing and storage, the webserver communicates with the data center. This ensures that the system can manage large datasets and complex computations.

- **Sensors to Webserver:**

Sensors provide real-time data through MQTT to the webserver. This data is crucial for monitoring environmental conditions and detecting emergencies, allowing the EMS to respond promptly.