REPUBLIQUE DU CAMEROON
PAIX-Travail-Patrie
MINISTRE DE L'ENSEIGNEMENT SUPERIEUR

FACULTE DE L'ENGINERIE ET TECHGNOLOGIE

REPUBLIC OF CAMEROON
Peace-Work-Fatherland
MINISTER OF HIGHER EDUCATION

FACULTY OF ENGINEERING AND TECHNOLOGY

********** UNIVERSITY OF BUEA **********

DEPARTMENT OF COMPUTER ENGINEERING

COURSE: INTERNET PROGRAMMING AND MOBILE APPLICATIONS

COURSE CODE: CEF 440

TASK 4: SYSTEM MODELING AND DESIGN

Facilitator: Dr. Valery Nkemeni

Academic year 2023/2024

GROUP MEMBERS

S/N	**Name**	Matricule Number
1	MUYANG ROSHELLA MBAMUZANG	FE21A243
2	TAMBONG KERSTEN MELENGFE	FE21A440
3	EBAI ENOWNKU JANE	FE21A176
4	AGBOR NKONGHO KELLY	FE21A126
5	NICCI NSE NCHAMI	FE21A268

Table of Content

Introduction

System diagrams play a crucial role in the effective modeling and design of any given system, serving as a common language for stakeholders as they support system analysis and optimization, facilitate deployment, maintenance and future enhancements. In this report we are going to present the various system diagrams i.e context, use-case, class, sequence and deployment diagrams of our disaster management system drafted from our requirement analysis to fully illustrate the strengths of our system.

1. Context Diagram

It is a diagram used to represent the high-level interactions between a system and its environment, providing a visual representation of the system and its interactions with external entities known as actors. It is typically made up of:

- The system which is the main focus of the diagram, represented as a single element using a box or circle.
- External entities which are the people or other systems that interact with the system, visually represented using boxes or other shapes connected to the system.
- Interactions that are represented as arrows or lines connecting the system external entities, showing the data flow or communication between them.

The context diagram of our disaster management system is shown and explained below.

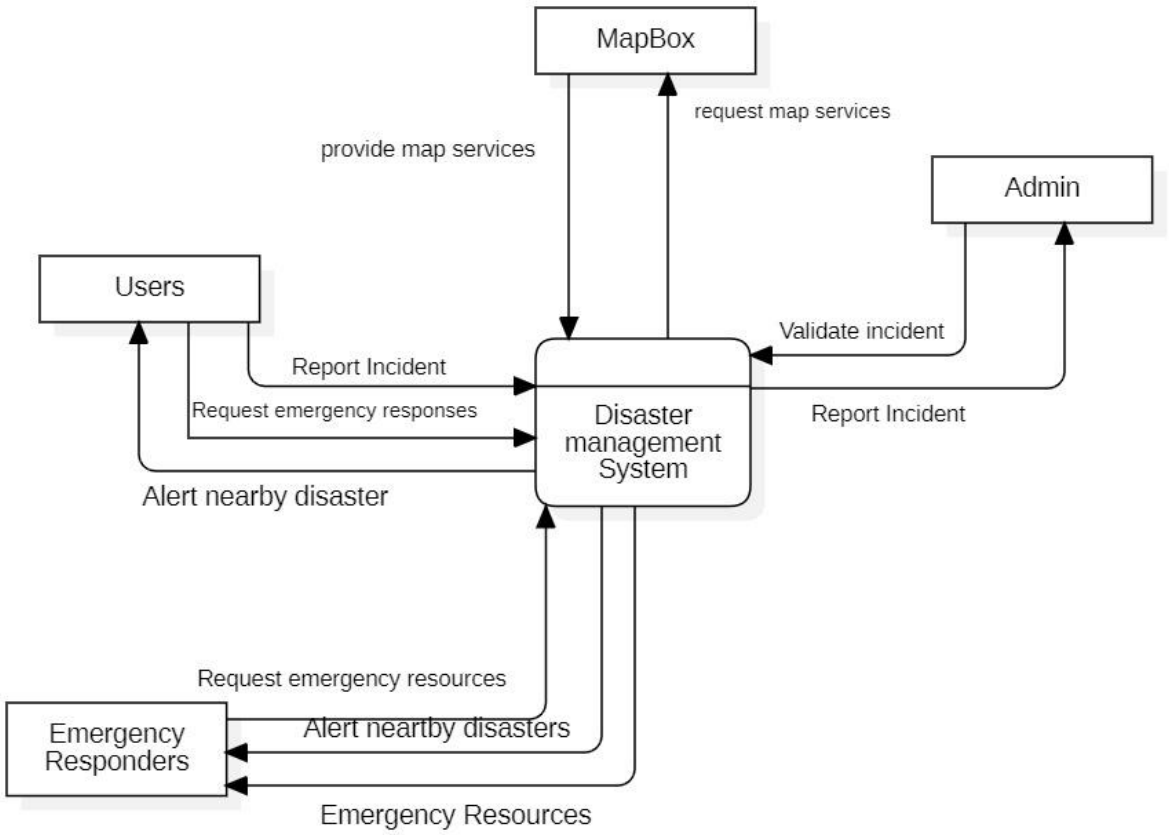

Fig1: Context Diagram

Context Diagram Explained

i. **Users**: These are the individuals who interact with the Disaster Management System. Users may include the general public, volunteers, or any other group that needs to access or provide information related to a disaster situation. They can report emergencies, receive alerts, and possibly access guidance on how to respond in different scenarios.

ii. **Emergency Responders**: This group includes professionals like firefighters, medical teams, police, and other emergency services who are directly involved in responding to disasters. They use the system to receive alerts and information about disaster events, which helps them in coordinating and executing response strategies efficiently.

iii. **Admin**: This refers to the administrators of the Disaster Management System. They are responsible for overseeing the operation of the system, managing data, configuring

settings, and ensuring that both the backend and the user interfaces are functioning correctly. Admins may also be involved in training users and emergency responders on how to effectively use the system.

iv. **Disaster Management System**: At the core, this is the software platform that facilitates communication and data exchange among all parties involved. It collects data from various sources, processes it, and disseminates information as needed. This system may include capabilities such as real-time data analytics, mapping of disaster areas, and automated alert systems.

v. **MapBox**: This component suggests the use of MapBox, a service known for providing detailed, customizable maps and location services. In the context of the Disaster Management System, MapBox is likely used to visualize geographic data on maps. This can include the locations of emergencies, routes for responders, and areas of interest such as hospitals, shelters, or high-risk zones.

The interaction between these components ensures that the system can effectively manage and respond to disasters. Users and emergency responders can interact with the system, which is centrally managed by admins using tools and data services such as MapBox to facilitate effective disaster management. This setup aims to enhance the speed and efficiency of the disaster response, improving overall safety and coordination.

2. Use Case Diagram

A use case diagram is a diagram used to model the functionality of a system from the perspective of the user or external actor, illustrating the different ways in which users can interact with the system. It is primarily made up of:

- Actors that represent external entities such as users that interact with the system and are visually represented by stick creatures.
- Use cases that represent the specific actions or functions that the system performs in response to an actor's request, visually represented using ovals.
- Relationships which are the connections between actors and use cases, represented using lines and arrows (depending on the type of relationship).

The use case diagram of our disaster management system is shown and explained below.

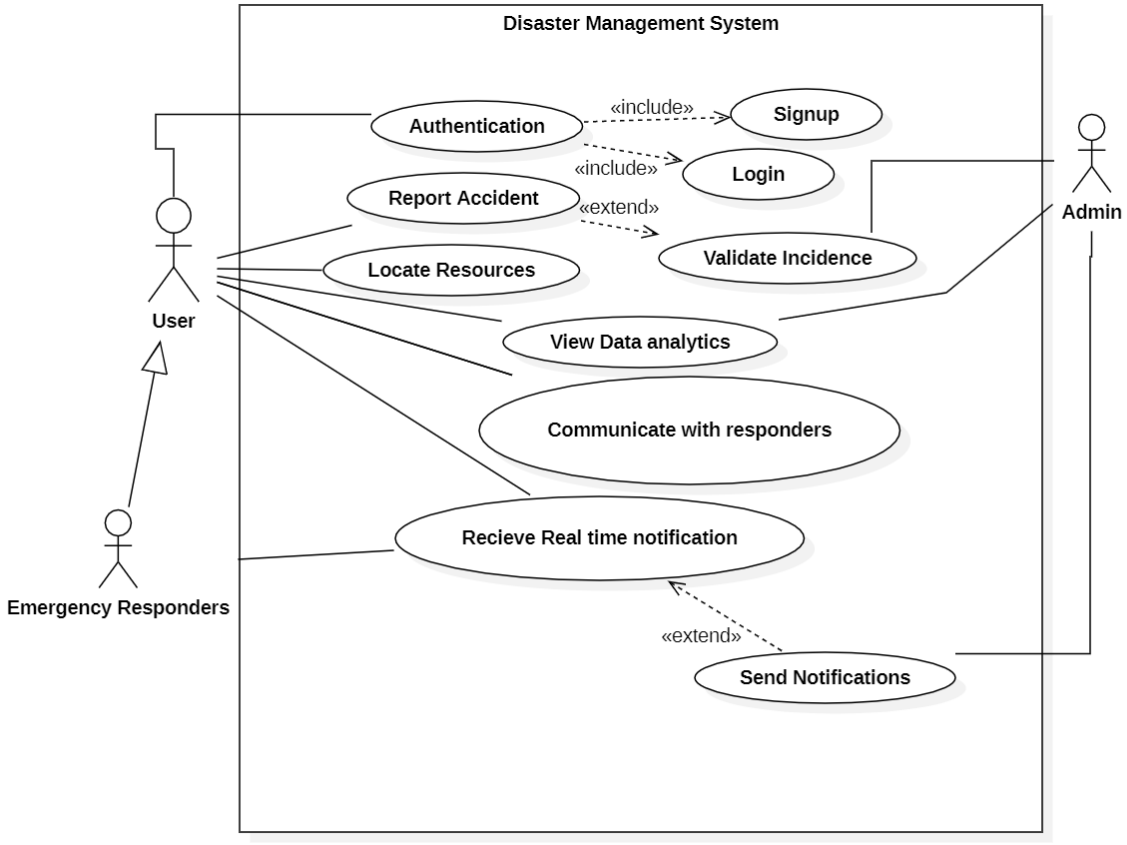

Fig2: Use case Diagram

Use Case Diagram Explained

Main Use Cases:

- **Authentication**: This use case encompasses two scenarios:
  - **Signup**: The process of creating a new user account is part of authentication and is essential for new users to access the system.
  - **Login**: Allows existing users to access their accounts. It is included in the Authentication use case, indicating that it is a necessary part of authenticating users.
- **Report Accident**: Users can report accidents or incidents through the system. It potentially extends to "Validate Incidence", which suggests that reporting an accident might sometimes require validation by an administrator or another entity.
- **Locate Resources**: Users can search for and locate necessary resources such as emergency supplies, shelters, etc.
- **View Data Analytics**: Users can view analytics related to disaster management, which may involve statistical data on past incidents, resource usage, etc.
- **Communicate with Responders**: This use case allows direct communication between the system's users and emergency responders.
- **Receive Real Time Notification**: Users receive notifications about relevant events or emergencies in real-time. This extends into the "Send Notifications" use case, which indicates that sending notifications might be a special case or an extended feature of receiving notifications.

Relationships:

- **«include»**: This relationship indicates that the included use case is an indispensable part of the base use case. For example, both Signup and Login are included in Authentication, meaning they are fundamental parts of the Authentication process.
- **«extend»**: This relationship indicates that the extended use case adds optional functionality or a special case scenario to the base use case. For example, "Send Notifications" is an extension of "Receive Real Time Notification", suggesting that under certain conditions, receiving notifications might lead to sending notifications as well.

Overall System Functionality:

The use case diagram effectively summarizes the system's key functionalities and how users interact with these functionalities. It helps developers and stakeholders understand the system

requirements and the scope of the project, ensuring that all necessary user interactions are considered during the development phase.

3. Sequence Diagram

A sequence diagram is a type of interaction diagram that shows how objects interact with each other in a specific scenario, emphasizing the time sequence of messages. They are particularly useful for understanding and documenting complex scenarios and help developers and even stakeholders to identify potential issues, optimize the design and ensure the system's behavior aligns with the requirements. It is made up of:

- Actors that represent the external entities interacting with the system, visually represented using stick creatures.
- Objects that represent the various components within the system that participate in the reaction, indicated by rectangles.
- Lifelines which are vertical lines used to represent the existence of each object over time.
- Messages which are arrows that represent the communication between objects.
- Time which is the horizontal axis representing the passage of time, with time flowing from top to bottom. The sequence diagram of our disaster management system is shown and explained below

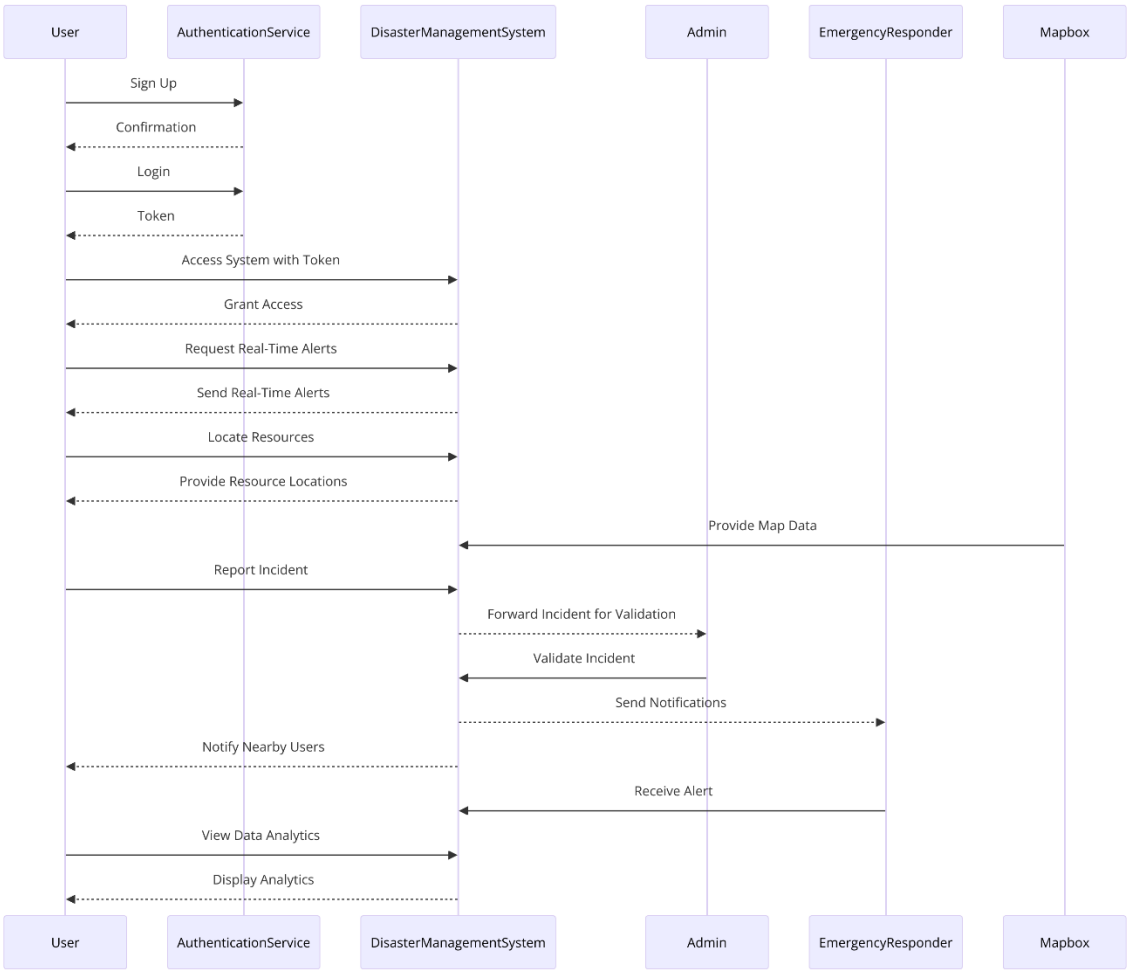

Fig3: Sequence diagram

Sequence Diagram Explained

i. **User to AuthenticationService**:

- **Sign Up**: The user registers for the disaster management system.
- **Confirmation**: The user receives a confirmation of successful registration.
- **Login**: The user logs into the system.
- **Token**: The authentication service provides a token after successful login.

ii. **User to DisasterManagementSystem**:

- **Access System with Token**: The user accesses the disaster management system using the token received.
- **Grant Access**: The system grants access based on the validity of the token.

- **Request Real-Time Alerts**: Similar to the previous diagram, the user requests alerts.
- **Send Real-Time Alerts**: The system sends alerts back to the user.
- **Locate Resources**: The user requests the location of necessary resources.
- **Provide Resource Locations**: The system provides these locations.
- **Report Incident**: The user reports an incident to the system.
- **Notify Nearby Users**: The system notifies users nearby the incident location.
- **View Data Analytics**: The user requests to view analytics related to disaster management.
- **Display Analytics**: The system displays the requested analytics.

iii. **DisasterManagementSystem to Admin**:

- **Forward Incident for Validation**: The system forwards the incident reported by the user to the admin for verification.
- **Validate Incident**: The admin validates the incident.

iv. **Admin to EmergencyResponder**:

- **Send Notifications**: After the incident is validated, the admin sends notifications to emergency responders.

v. **EmergencyResponder to Mapbox**:

- **Receive Alert**: The emergency responders receive the alert.
- **Provide Map Data**: Mapbox provides relevant map data to assist emergency responders in navigating to the incident location.

4. Class Diagram

A class diagram is a diagram that shows the structure of a system by depicting the system's classes, their attributes, operations (or methods), and the relationships among the classes.

Class diagrams are very vital when it comes to understanding the system's domain, identifying the key concepts and their relationships and providing a foundation for the implementation of a system's functionality. Typically, they are made up of:

- Classes which are the fundamental building blocks of the system, represented as rectangles. Each class contains information about its name, attributes and methods.
- Attributes which are the properties of a class.
- Methods which are the functions or operations that a class can perform
- Relationships which are the connections between classes.

The class diagram of our disaster management system is shown and explained below:

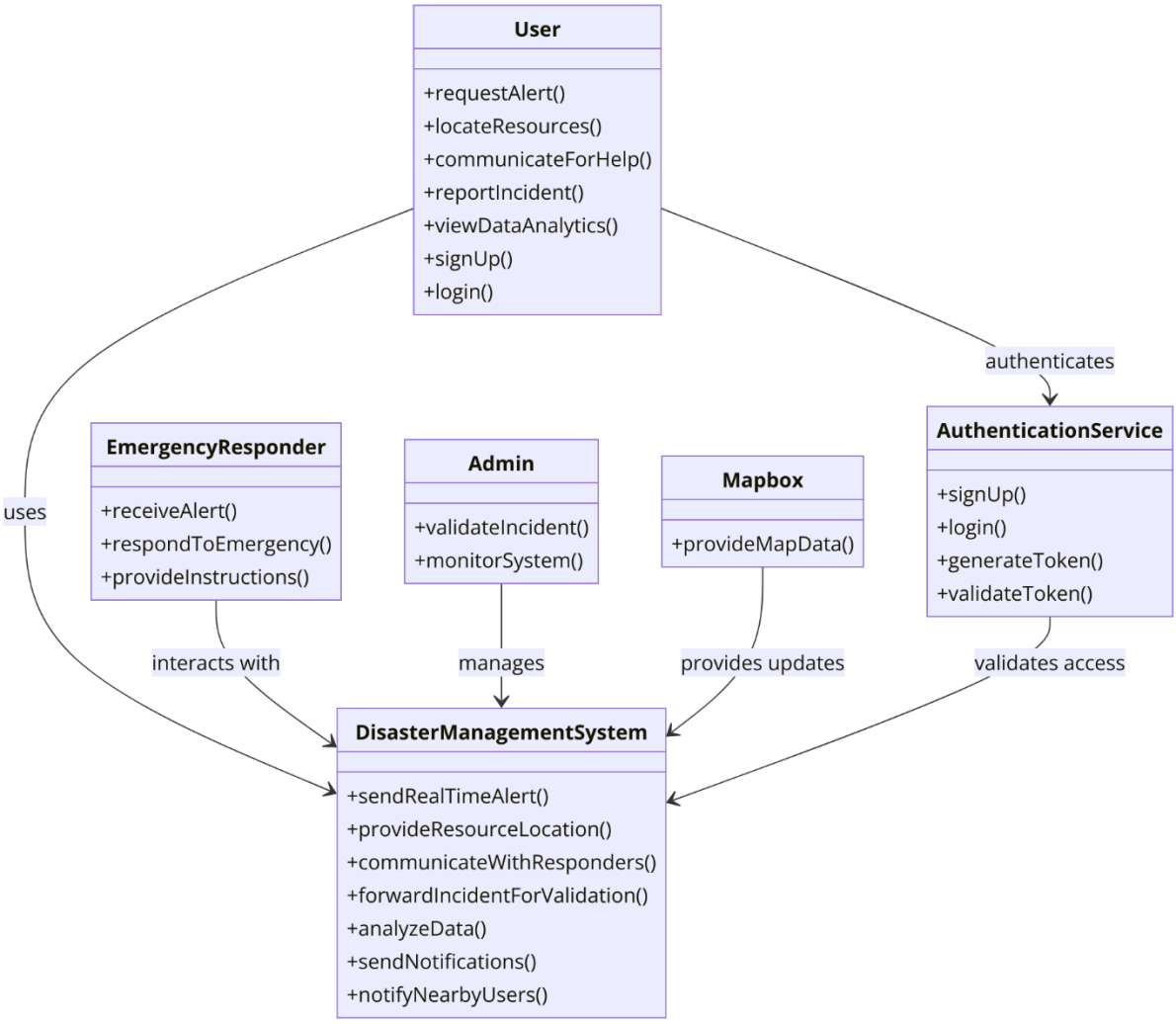

Fig4: Class Diagram

Class Diagram Explained

i. **User Functions:** Users can request alerts, locate resources, communicate for help, report incidents, view analytics and manage their account through signup and login functions.
   - **Interaction:** Users authenticate through the Authentication Service.

ii. **Emergency Responder Functions:** They receive alerts, respond to emergencies, and provide instructions to the users or other responders.
   - **Interaction:** Emergency Responders use the Disaster Management System to get information and interact with other parts of the system.

iii. **Admin Functions:** Admins validate incidents reported by users and monitor the overall system to ensure its effectiveness and integrity.
   - **Management:** Admins have managerial control over the Disaster Management System, which allows them to send real-time alerts and manage system operations.

iv. **Authentication Service Functions:** Manages user authentication by enabling signup, login, token generation, and token validation to ensure secure access to the system.

v. **Mapbox Function:** Provides geographic data that might be necessary for locating incidents or resources within the disaster management process.
   - **Updates:** Mapbox interacts with the Disaster Management System to update or provide necessary map data.

vi. **Disaster Management System Core Functions:** Includes sending real-time alerts, providing resource locations, communicating with responders, validating incidents, analyzing data, sending notifications, and notifying nearby users.

- **Interaction:** This system is the central unit that connects all other components and user roles, ensuring the efficient flow of information and operations necessary for managing disasters.

5. Deployment Diagram

A deployment diagram is a diagram used to model the physical deployment or execution environment of a software system. It shows how the software components are deployed on the hardware infrastructure. By providing a clear visual representation of the system's physical deployment, deployment diagrams facilitate communication, enable the identification of potential issues and support overall design and implementation of the software system.

It is typically made up of:

- Nodes that represent the physical hardware or computing devices such as servers, workstations or mobile devices on which the software components are deployed.
- Artifacts which represent the deployable software components such as libraries or files that reside on the nodes.
- Relationships

The deployment diagram of our disaster management system is shown and explained below:

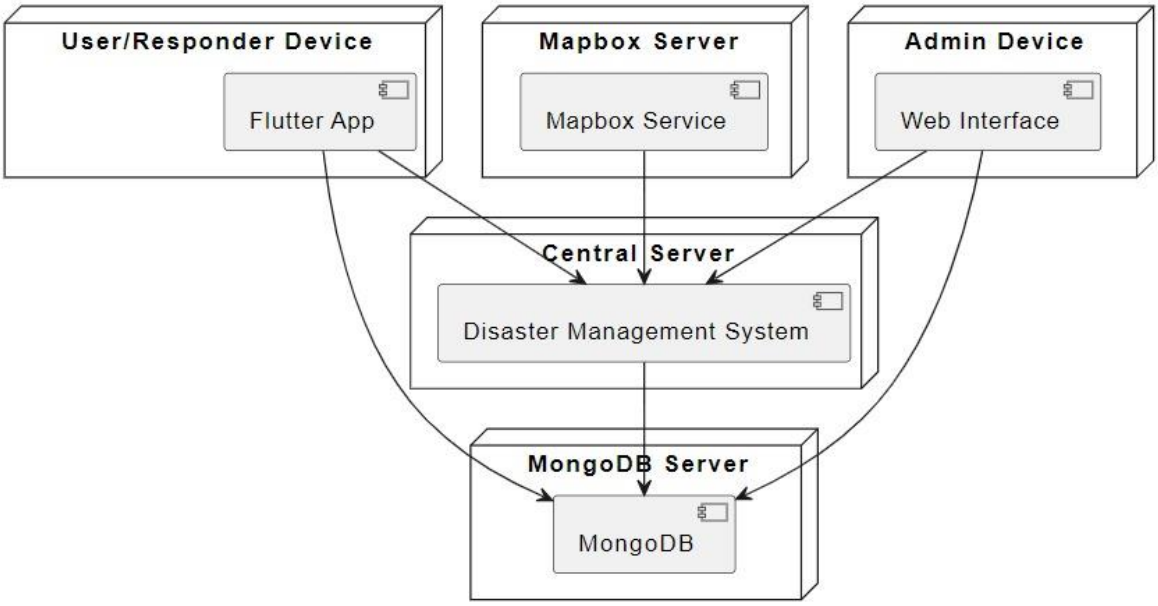

Fig5: Deployment diagram

Deployment Diagram Explained

Components:

i. **User/Responder Device**:

- **Flutter App**: This represents the application used by both users and responders. Flutter is a framework by Google that allows the development of natively compiled applications for mobile, web, and desktop from a single codebase. This device interacts directly with the Central Server.

ii. **Mapbox Server**:

- **Mapbox Service**: This server provides mapping services that are probably used within the Flutter app for things like mapping incidents, navigation, and locating resources. It interacts with the Central Server, providing necessary geographical data.

iii. **Admin Device**:

- **Web Interface**: This device is used by administrators to manage and monitor the system, possibly handling tasks like incident validation, resource management, and system configurations. It communicates with the Central Server.

iv. **Central Server**:

- **Disaster Management System**: The core component where most of the processing and data handling occurs. This server acts as the hub for all communications between the user/responder devices, the admin device, and the MongoDB Server. It also interfaces with the Mapbox Server to retrieve or send geographical data.

v. **MongoDB Server**:

- **MongoDB**: A NoSQL database used to store all data relevant to the disaster management system, including user data, incident reports, resource locations, etc. This server supports the Central Server by providing data storage and retrieval capabilities.

❖ **Interactions:**

- **User/Responder Device ↔ Central Server**: The Flutter app on the user/responder device sends and receives data from the Central Server. This could include sending incident reports, receiving alerts, and fetching resource locations.
- **Mapbox Server ↔ Central Server**: The Central Server queries the Mapbox Service for map-related data which is necessary for both user and administrative functions, such as displaying incident locations or planning responder routes.
- **Admin Device ↔ Central Server**: The web interface used by administrators interacts with the Central Server for managing and monitoring system operations, such as validating incident data and updating system settings.
- **MongoDB Server ↔ Central Server**: The Central Server interacts with the MongoDB Server to store and retrieve all necessary data, ensuring that the system has timely access to updated and persistent information.

Conclusion

Throughout this report, we have explored the key aspects of system modeling and design using the Unified Modeling Language (UML). The various UML diagrams presented provide a comprehensive view of the system's structure, behavior and deployment, allowing stakeholders to gain a deep understanding of the system's functionality and architecture, and lays a strong foundation for the implementation and deployment of our disaster management system. By leveraging these UML techniques, the team has been able to thoroughly analyze the system, identify potential issues and make informed decisions to optimize the overall architecture. Moving forward these UML models will continue to be refined and updated as the system evolves, ensuring that the design remains aligned with the changing requirements and technological landscape.