System Design And Modelling

PRESENTED BY
GROUP 22

Outline

1. Introduction
2. System Design
3. System Modelling

1. Introduction

Our system design and modeling process involved grouping functionalities based on similarities in implementation and breaking them down into modules, each providing a specific set of functions.

This approach was used to come up with the system design and the system model using UML diagrams.

2. System Design

In coming up with the system design for instance, we grouped the **Request for help and report disaster.** Them we came up with the following modules:

i. Validation of Users Statement
   1. Validate existence of location provided
   2. Ensure contact image is correctly provided
   3. Check disaster exists
   4. Ensure video format is correct, size limits
   5. Ensure description provided meets minimum length criteria

ii. Verification and validation of user statement

iii. Feedback based on authenticity

Choosing Architecture

In choosing our architecture, we mainly took into consideration the non-functional requirements, since the non-functional requirements will be directly affected by the architecture. We mainly considered the following:

1. Scalability
2. Performance
3. Offline Functionality

Based on this, we combined two architectures

- Client-sever Architecture

- Offline-first Architecture

Technologies

Considering the architecture and importance of the application, we chose to build a hybrid application using the following technologies:

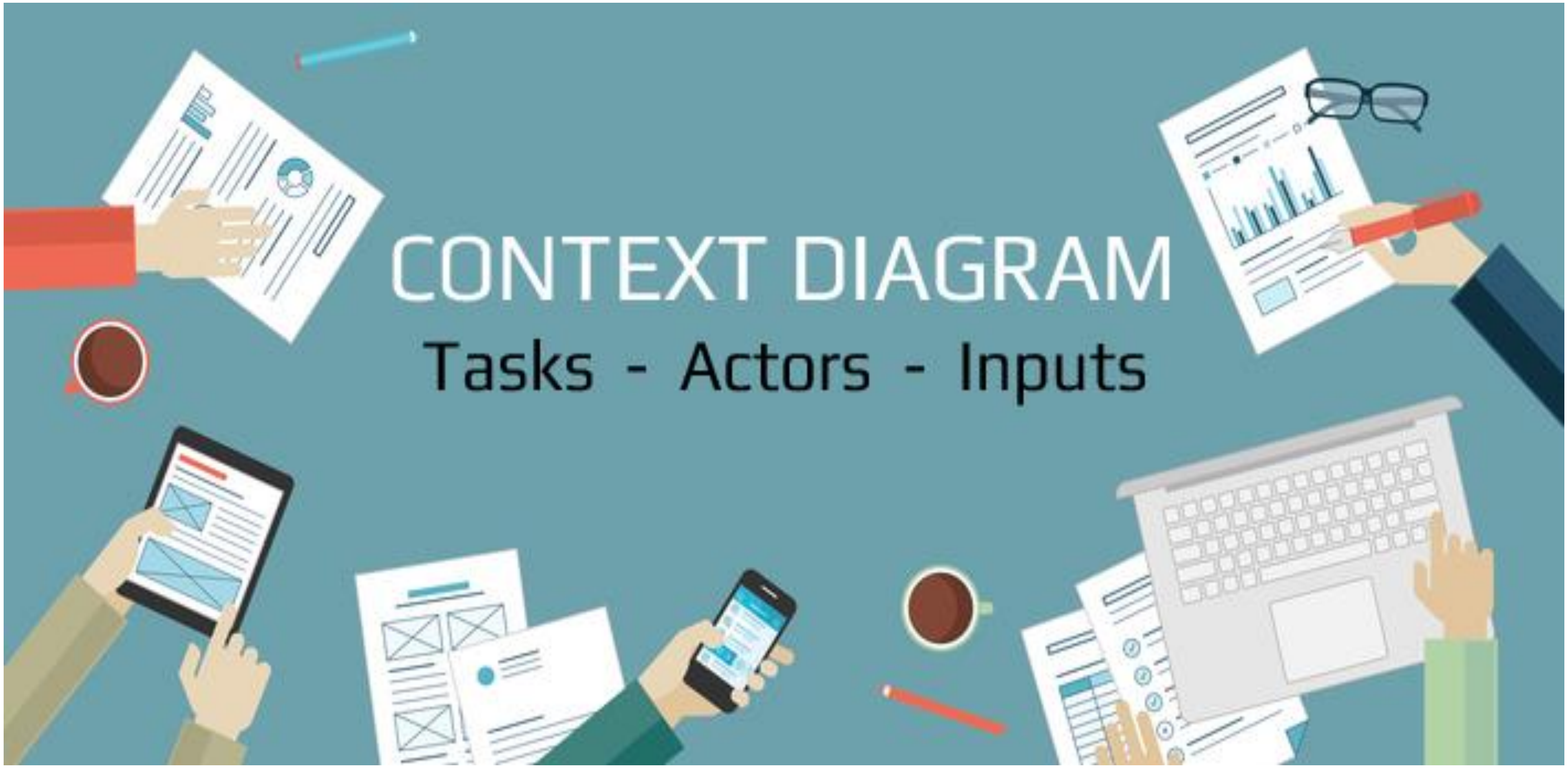
- Client: Ionic with ReactTs
- Server: Node Js
- Database: MongoDB

3. System Modelling

This is the process of creating diagrams and representations of a system to understand how it works and how its parts interact.

1. Context Diagram
2. Use Case Diagram
3. Sequence Diagram
4. Class Diagram
5. Deployment Diagram

3.1 Context Diagram



3.1 Context Diagram

A context diagram is a high-level visual representation of a system and its interactions with external entities. Components of a context diagram include:

i. **System**

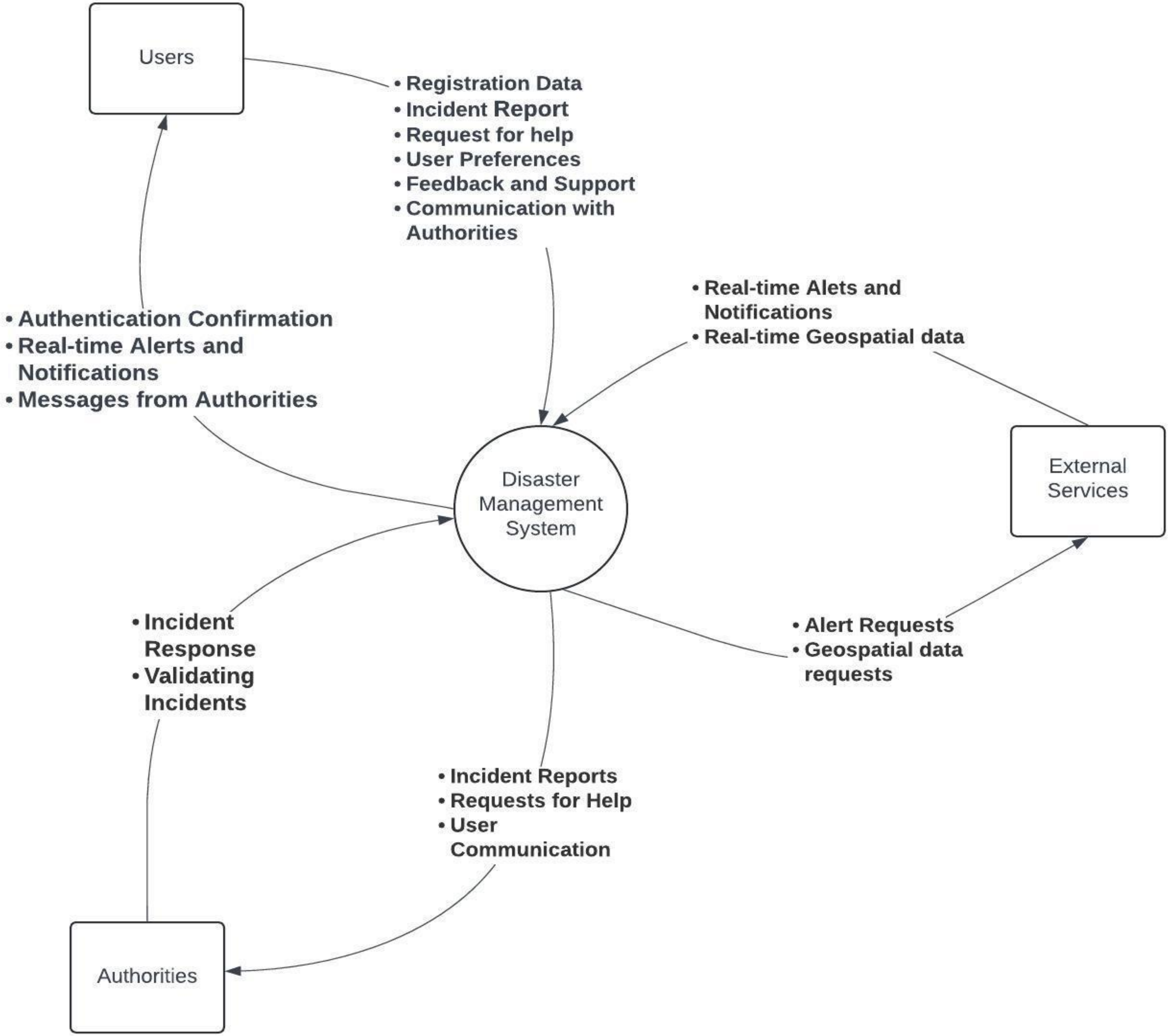
ii. **External Entities**

   a. Users

   b. Authorities

   c. External Services

iii. **Data Flows**



Users

- Registration Data
- Incident Report
- Request for help
- User Preferences
- Feedback and Support
- Communication with Authorities

- Authentication Confirmation
- Real-time Alerts and Notifications
- Messages from Authorities

- Real-time Alets and Notifications
- Real-time Geospatial data

Disaster Management System

External Services

- Incident Response
- Validating Incidents

- Alert Requests
- Geospatial data requests

- Incident Reports
- Requests for Help
- User Communication

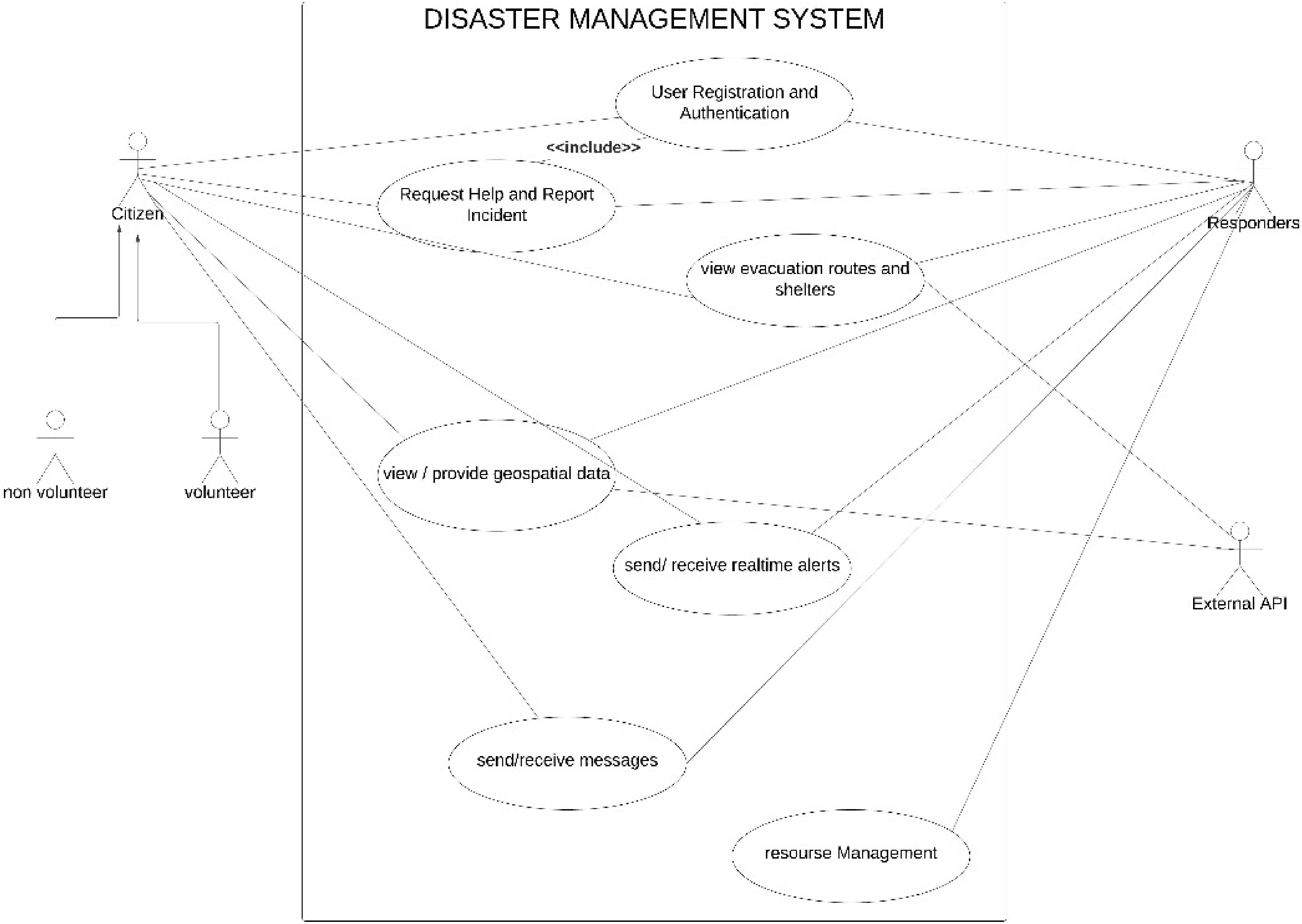
Authorities

3.2 Use Case Diagram

A use case diagram is a graphical representation that depicts how users interact with a system and its functionalities.

Actors:

1. User
2. Emergency Responders
3. External APIs (Geospatial Data Provider


DISASTER MANAGEMENT SYSTEM
User Registration and Authentication
<<include>>
Request Help and Report Incident
view evacuation routes and shelters
view / provide geospatial data
send/ receive realtime alerts
send/receive messages
resourse Management
Citizen
non volunteer
volunteer
Responders
External API

3.3 Sequence Diagram

A sequence diagram is a visual representation of the message flow between objects in a system, arranged in chronological order. It shows process interactions arranged in time sequence.

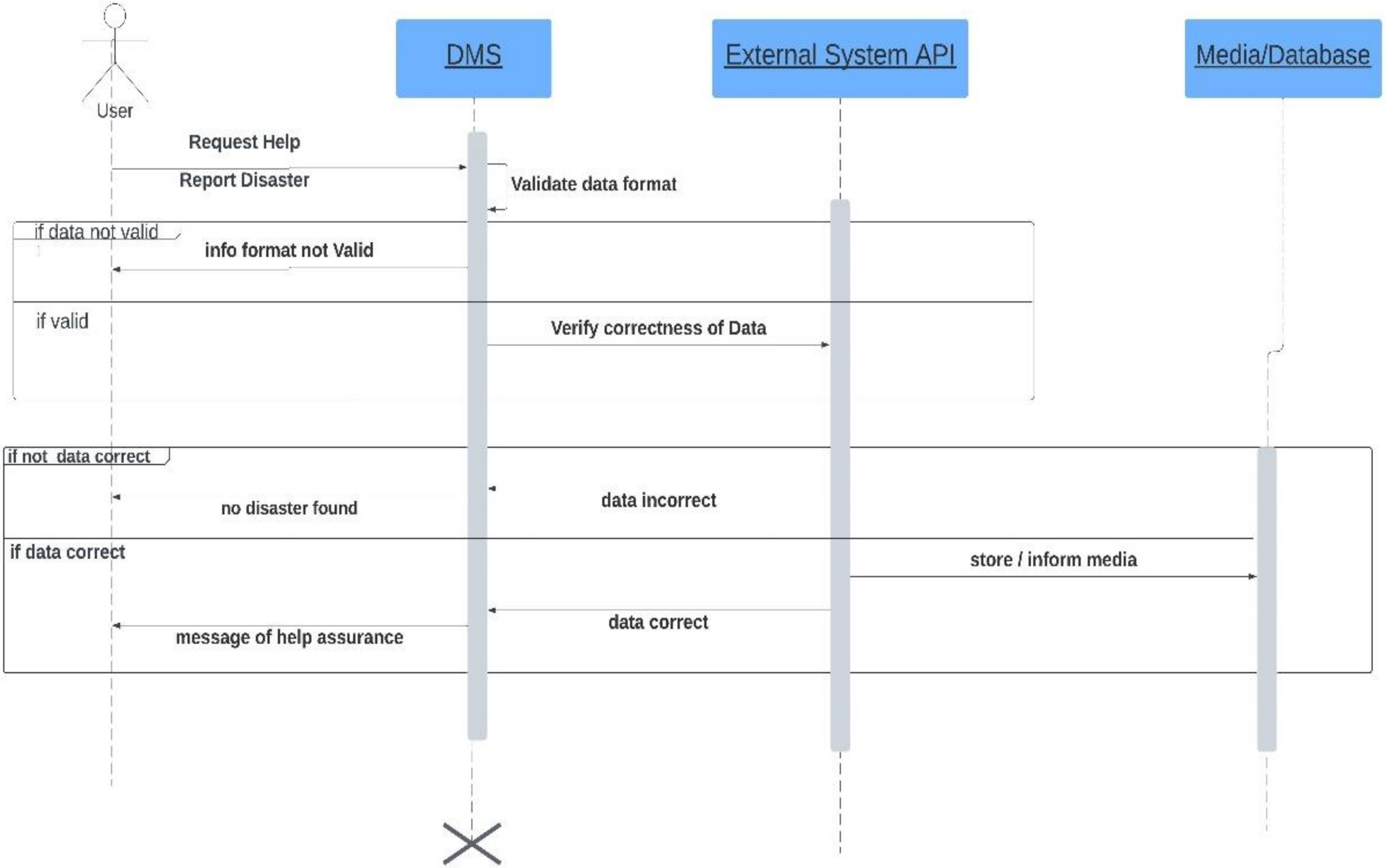
1. Report Disaster and Request For Help
2. Geospatial Data Integration

3.3.1 Report Disaster and Request for Help

This sequence diagram shows the steps involved in a user reporting a disaster event and requesting assistance during a disaster.

Objects

- Disaster Management System
- External System API
- Media/Database

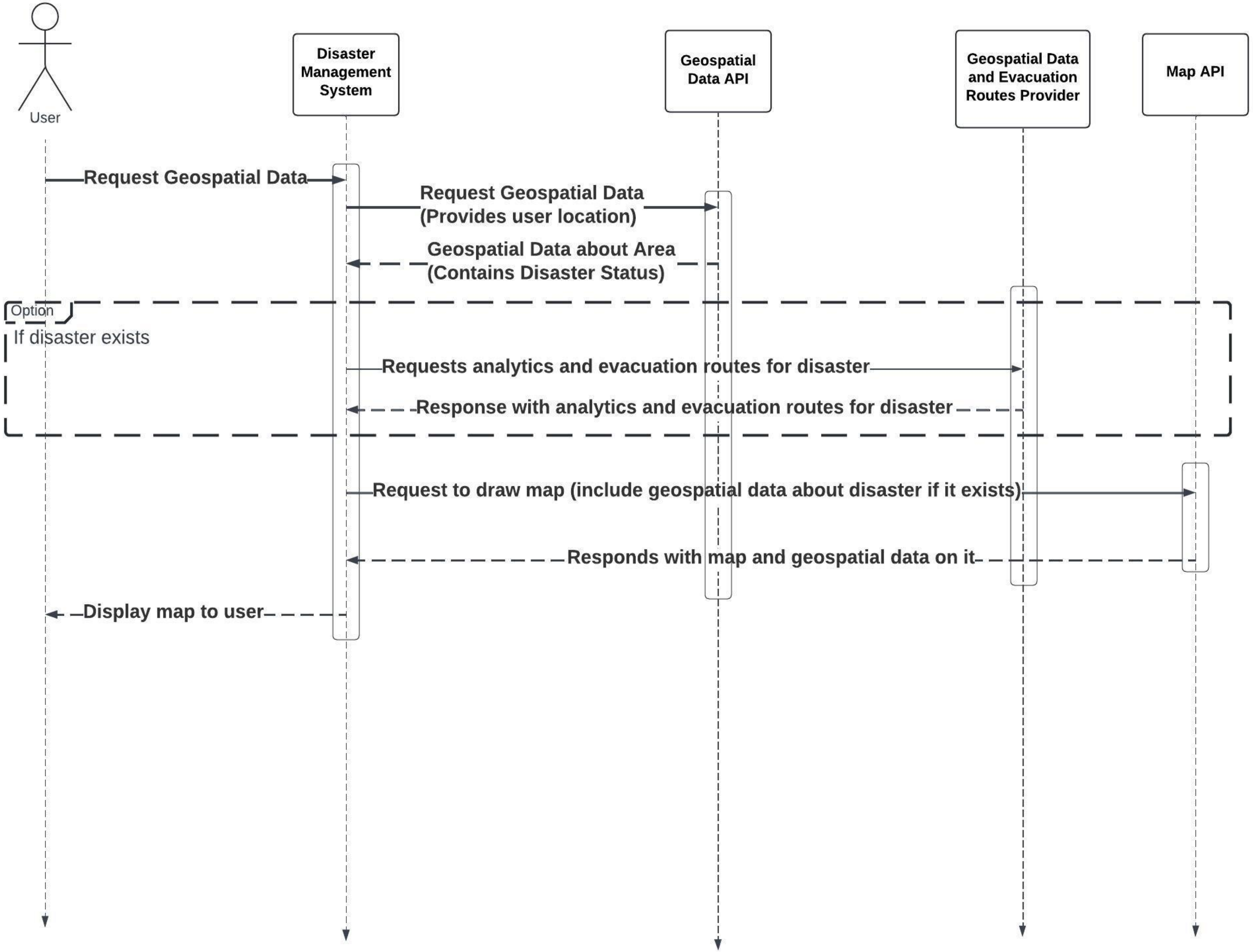

User
DMS
External System API
Media/Database
Request Help
Report Disaster
Validate data format
if data not valid
info format not Valid
if valid
Verify correctness of Data
if not data correct
no disaster found
data incorrect
if data correct
store / inform media
data correct
message of help assurance

3.3.2 Geospatial Data Integration

This sequence diagram illustrates how the DMS retrieves and integrates geospatial data to enhance situational awareness during a disaster.

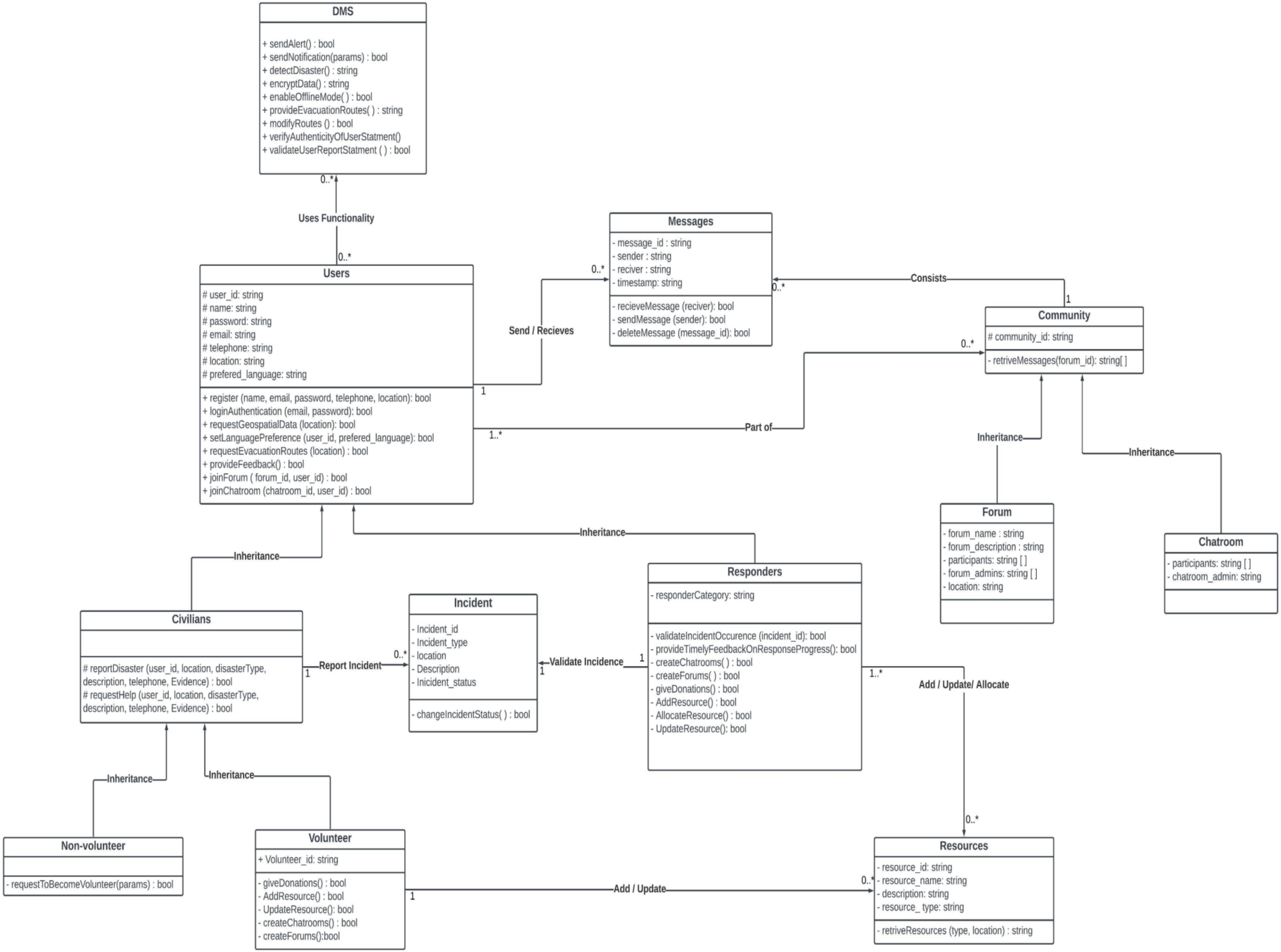
Objects:

1. Disaster Management System (DMS)
2. Geospatial Data Provider API
3. Geospatial Data Analytics Provider API and Evacuation Route Service Provider API (Optional
4. Mapping Service Provider API

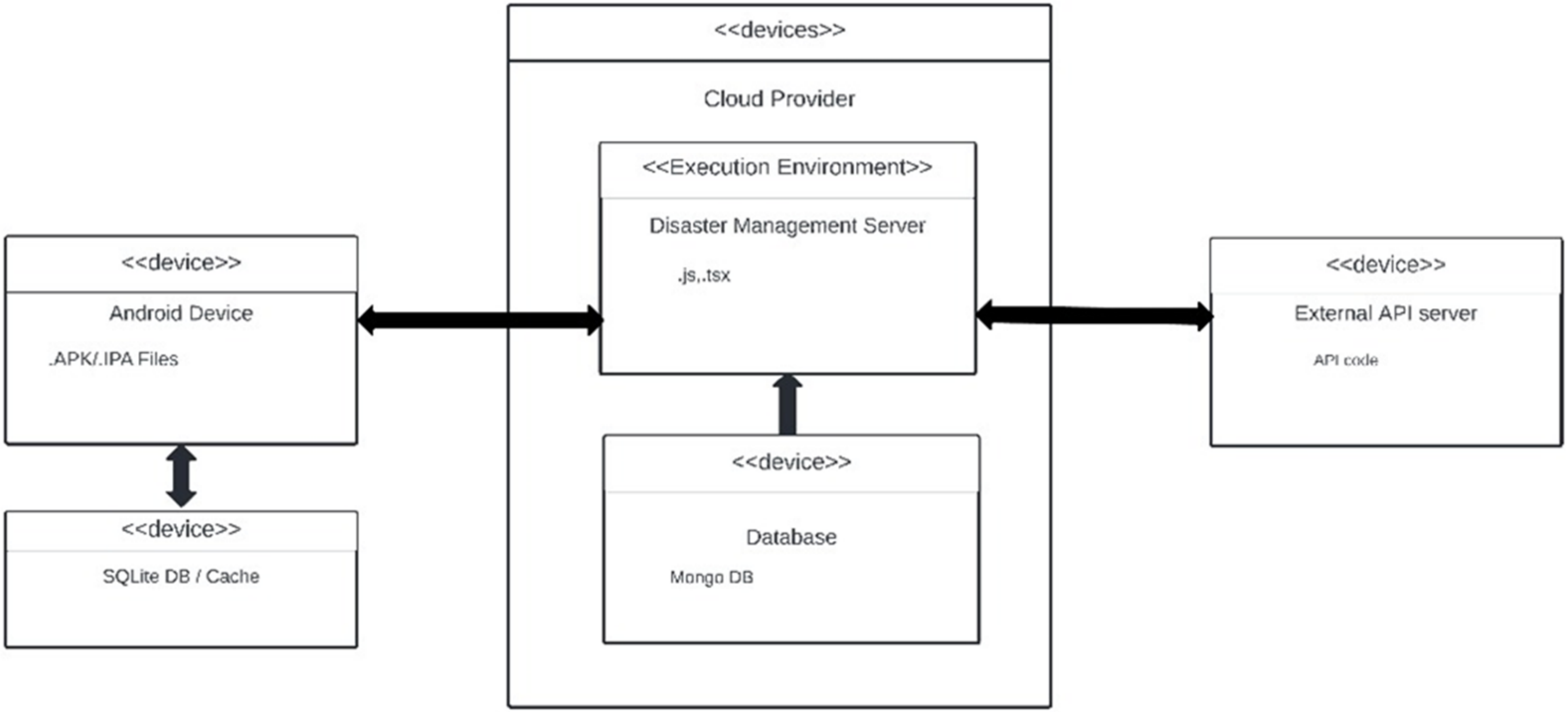

User
Disaster Management System
Geospatial Data API
Geospatial Data and Evacuation Routes Provider
Map API
Request Geospatial Data
Request Geospatial Data (Provides user location)
Geospatial Data about Area (Contains Disaster Status)
Option
If disaster exists
Requests analytics and evacuation routes for disaster
Response with analytics and evacuation routes for disaster
Request to draw map (include geospatial data about disaster if it exists)
Responds with map and geospatial data on it
Display map to user

3.4 Class Diagram

The class diagram of our disaster management system shows the various classes their attributes, methods and relationship to other classes.


DMS
+ sendAlert() : bool
+ sendNotification(params) : bool
+ detectDisaster() : string
+ encryptData() : string
+ enableOfflineMode( ) : bool
+ provideEvacuationRoutes( ) : string
+ modifyRoutes () : bool
+ verifyAuthenticityOfUserStatment()
+ validateUserReportStatment ( ) : bool
0..*
Uses Functionality
0..*
Users
# user_id: string
# name: string
# password: string
# email: string
# telephone: string
# location: string
# prefered_language: string
+ register (name, email, password, telephone, location): bool
+ loginAuthentication (email, password): bool
+ requestGeospatialData (location): bool
+ setLanguagePreference (user_id, prefered_language): bool
+ requestEvacuationRoutes (location) : bool
+ provideFeedback() : bool
+ joinForum (forum_id, user_id) : bool
+ joinChatroom (chatroom_id, user_id) : bool
Send / Recieves
1
0..*
Messages
- message_id : string
- sender : string
- reciver : string
- timestamp: string
- recieveMessage (reciver): bool
- sendMessage (sender): bool
- deleteMessage (message_id): bool
0..*
Consists
1
Community
# community_id: string
- retriveMessages(forum_id): string[ ]
0..*
Part of
1..*
Inheritance
Inheritance
Forum
- forum_name : string
- forum_description : string
- participants: string [ ]
- forum_admins: string [ ]
- location: string
Chatroom
- participants: string [ ]
- chatroom_admin: string
Inheritance
Inheritance
Responders
- responderCategory: string
- validateIncidentOccurence (incident_id): bool
- provideTimelyFeedbackOnResponseProgress(): bool
- createChatrooms( ) : bool
- createForums( ) : bool
- giveDonations() : bool
- AddResource() : bool
- AllocateResource() : bool
- UpdateResource(): bool
Civilians
# reportDisaster (user_id, location, disasterType, description, telephone, Evidence) : bool
# requestHelp (user_id, location, disasterType, description, telephone, Evidence) : bool
1
Report Incident
0..*
Incident
- Incident_id
- Incident_type
- location
- Description
- Inicident_status
- changeIncidentStatus( ) : bool
1
Validate Incidence
1
1..*
Add / Update/ Allocate
0..*
Inheritance
Inheritance
Non-volunteer
- requestToBecomeVolunteer(params) : bool
Volunteer
+ Volunteer_id: string
- giveDonations() : bool
- AddResource() : bool
- UpdateResource(): bool
- createChatrooms() : bool
- createForums():bool
1
Add / Update
0..*
Resources
- resource_id: string
- resource_name: string
- description: string
- resource_ type: string
- retriveResources (type, location) : string

3.5 Deployment Diagram



<<devices>>

Cloud Provider

<<Execution Environment>>

Disaster Management Server

.js,.tsx

<<device>>

Database

Mongo DB

<<device>>

Android Device

.APK/.IPA Files

<<device>>

SQLite DB / Cache

<<device>>

External API server

API code

THE END!!!