 **THE UNIVERSITY OF BUEA** **REPUBLIC OF CAMEROON**

P.O Box 63, PEACE-WORK-FATHERLAND

Buea, Southwest Region

Cameroon

Tel: (237) 674354327

Fax: (237) 3332 22 72

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CEF 440: Internet programming and mobile programming**

**Design and Implementation of a Road Sign and Road State Mobile Notification Application**

Group 15

**TASK 6: Database Design and Implementation**

[GitHub - Asumu22/group-15](https://github.com/Asumu22/group-15)

<https://www.figma.com/design/Qk0rRSDJMGu4xmreSQygLA/RoadGuard?node-id=0-1&t=9nFD8sr3nFZS3M6a-1>

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Matriculation No | Specialty |
| 01 | KONGNYUY RAYMOND AFONI | FE21A219 | NE |
| 02 | BESONG ELIAS ASUMU | FE21A149 | SE |
| 03 | KANKO KEMEDJEU DUPLEX | FE21A210 | NE |
| 04 | KENEDY MALLEY ITUKA | FE21A212 | NE |
| 05 | AZEFACK JUNIOR | FE21A146 | NE |

**Dr. NKEMENI Valery**

**Course Supervisors 2023 - 2024 Academic Year**

Abstract

This report outlines the design and implementation of a NoSQL database for a mobile application focused on real-time road sign recognition, traffic updates, and driver safety. The database design is crucial for efficiently storing, retrieving, and managing data related to road signs, traffic lights, user profiles, and various other elements essential for the application's functionality. Given the application's requirements, Firebase will be used as the backend database due to its real-time capabilities and scalability. It outlines the necessary collections, fields, relationships, and the rationale behind the design choices. Firebase, a NoSQL database, has been selected for its real-time capabilities, ease of integration with mobile platforms, and scalability.

table of content

[Introduction 4](#_Toc169801084)

[1. Database Requirements 4](#_Toc169801085)

[2. Database modelling and Design 5](#_Toc169801086)

[2.1 TABLES (Collections in Firebase) 5](#_Toc169801087)

[1. Users Collection 5](#_Toc169801088)

[2. RoadSigns Collection 5](#_Toc169801089)

[3. TrafficLights Collection 5](#_Toc169801090)

[4. Hazards Collection 6](#_Toc169801091)

[5. NavigationRoutes Collection 6](#_Toc169801092)

[6. WeatherUpdates Collection 6](#_Toc169801093)

[7. EmergencyAlerts Collection 7](#_Toc169801094)

[8. UserFeedback Table 7](#_Toc169801095)

[2.2 Entity-Relationship Diagram VISUALIZATION WITH Cardinalities 8](#_Toc169801096)

[3. Database Implementation with Firebase 13](#_Toc169801097)

[3.1 Steps for Implementation 13](#_Toc169801098)

[1. Set up Firebase Project: 13](#_Toc169801099)

[2. Define Collections and Documents: 13](#_Toc169801100)

[3. Integrate Firebase with the Application: 13](#_Toc169801101)

[4. Implement Data CRUD Operations: 13](#_Toc169801102)

[5. Testing and Deployment: 13](#_Toc169801103)

[Conclusion 14](#_Toc169801104)

Task 6: Database Design and Implementation

# Introduction

The objective of this task is to design a database that supports the functionalities outlined in previous tasks, particularly focusing on storing and managing data related to road signs, traffic lights, user interactions, and real-time updates. The use of Firebase, a NoSQL database, offers advantages such as real-time synchronization, ease of integration with mobile platforms, and flexible data modeling.

The chosen database, Firebase, is suitable for this application due to its real-time synchronization features, which are essential for updating road and traffic conditions dynamically, and its flexibility in data modeling which is ideal for handling diverse data types involved in this project.

# 1. Database Requirements

Based on the requirements analysis, system modelling and design and UI design documents, the database must support the following:

* User authentication and profile management.
* Storage of real-time road sign and traffic light data.
* Logging and reporting of road hazards and user feedback.
* Management of route and navigation data.
* Provision of weather updates and emergency alerts.

# 2. Database modelling and Design

Here we bring out the er different dables(collection) with their attributes, map out their cadinalities and draw out the ER diagram to visualise clearly how data is stored for proper quering.

## 2.1 TABLES (Collections in Firebase)

The database design includes several collections and documents to store different types of data.

### 1. Users Collection

**Fields (Attributes):**

* + - userContact (Integer): Unique identifier for the user **– primary key🗝**.
    - name (String): Username of the user.
    - email (String): Email address and unique identifier for the user **– Alternate key🗝**.
    - password (String): Hashed password.
    - location (GeoPoint): Geographic location of the user.
    - LicenseNo (Integer): user License ID in case user is a driver **– Alternate key🗝**
    - gender (String): user gender
    - birthday (String): user’s birthday which help determine age.

### 2. RoadSigns Collection

**Fields (Attributes):**

* + - type (String): Type of road sign (e.g., stop sign, speed limit) **– primary key🗝**.
    - description (String): Description of the road sign.
    - location (GeoPoint): Geographic location of the sign.
    - imageUrl (String): URL of the sign's image.
    - detectedAt (Timestamp): Time of detection.

### 3. TrafficLights Collection

**Fields (Attributes):**

* + - color (String): Current color of the traffic light (red, yellow, green).
    - location (GeoPoint): Geographic location of the traffic light **- primary key🗝**.
    - description (String): Description of the traffic light.
    - detectedAt (Timestamp): Time of detection.

### 4. Hazards Collection

**Fields (Attributes):**

* + - hazardId (Integer): Unique identifier for the hazard **– primary key🗝**.
    - type (String): Type of hazard (e.g., accident, roadblock).
    - location (GeoPoint): Geographic location of the hazard.
    - userContact **(. – Foregn key🗝 to Users):** User ID of the reporter.
    - description (String): Description of the hazard.
    - reportedAt (Timestamp): Time of reporting.
    - likes (Integer): Number of likes.
    - comments (Array): Comments on the hazard.
    - media (Array): URLs of images or videos related to the hazard.

### 5. NavigationRoutes Collection

**Fields (Attributes):**

* + - startLocation (GeoPoint): Starting point of the route.
    - endLocation (GeoPoint): Destination point of the route.
    - waypoints (Array of GeoPoints): Intermediate points in the route.
    - trafficConditions (Array of Objects): Traffic conditions along the route.
    - createdAt (Timestamp): Time of route creation.

**– primary key🗝(startLocation, endLocation).**

### 6. WeatherUpdates Collection

**Fields (Attributes):**

* + - location (GeoPoint): Geographic location for the weather update.
    - temperature (Number): Current temperature.
    - conditions (String): Weather conditions (e.g., clear, rain).
    - updatedAt (Timestamp): Time of the update.

**– primary key🗝(location, updatedAt).**

### 7. EmergencyAlerts Collection

**Fields (Attributes):**

* + - message (String): Alert message.
    - type (String): Type of alert (e.g., weather, accident).
    - location (GeoPoint): Affected location.
    - issuedAt (Timestamp): Time of issuance.

**– primary key🗝(location, issuedAt).**

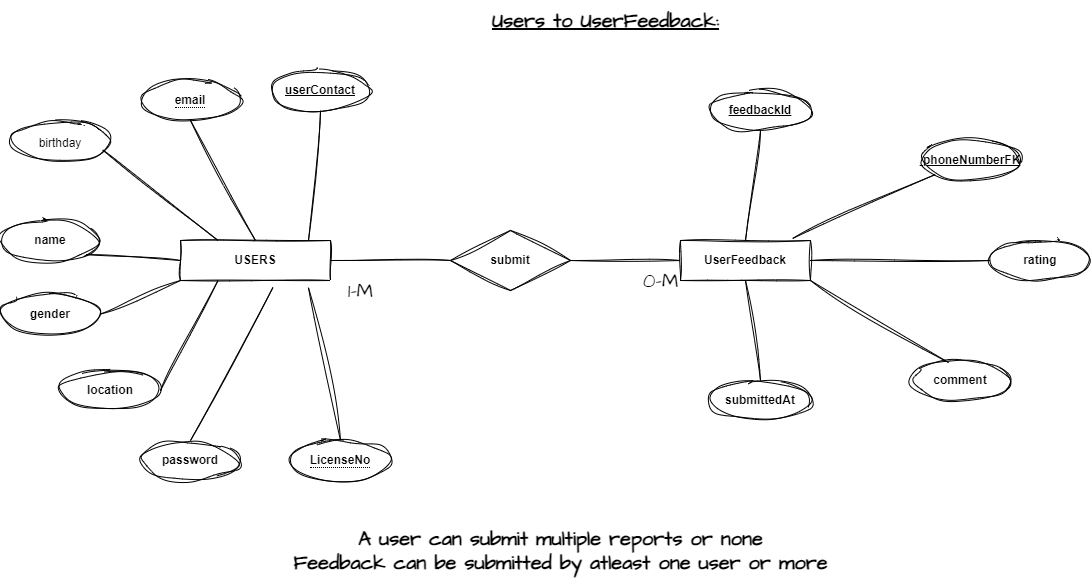
### 8. UserFeedback Table

**Fields (Attributes):**

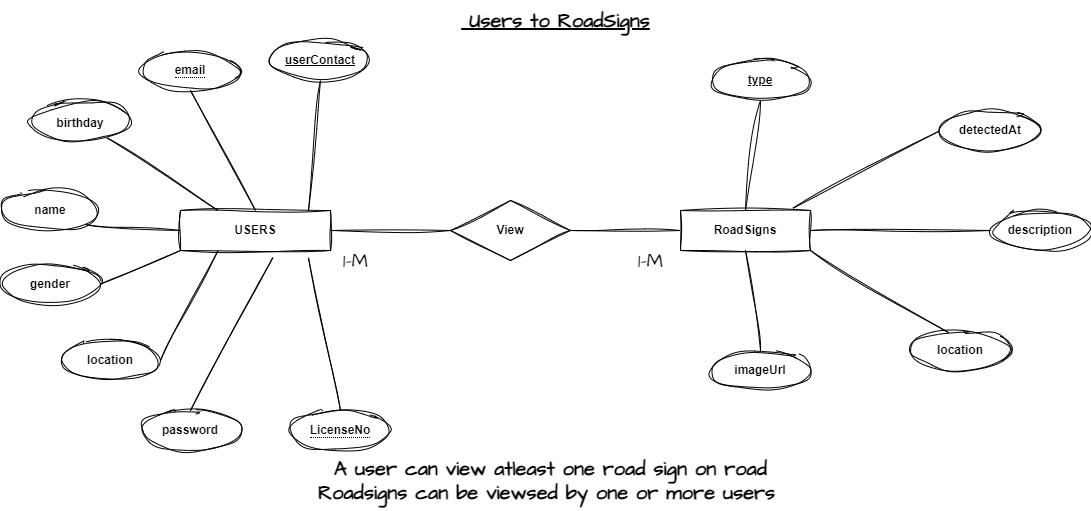
* feedbackId (Integer): Unique identifier for the feedback entry **– primary key🗝**.
* phoneNumber **(– Foregn key🗝 to Users):** Identifier of the user who provided the feedback.
* rating (Integer): Rating given by the user (on a scale of 1 to 5).
* comment (String): Textual feedback or comment provided by the user.
* submittedAt (Timestamp): Time when the feedback was submitted.

## 2.2 Entity-Relationship Diagram VISUALIZATION WITH Cardinalities

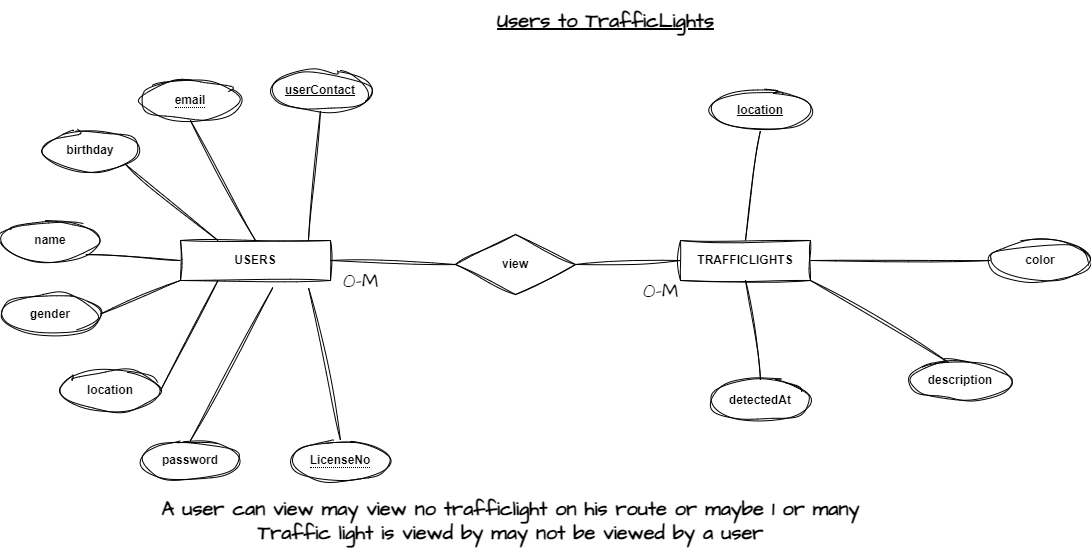
* **Users to UserFeedback**:



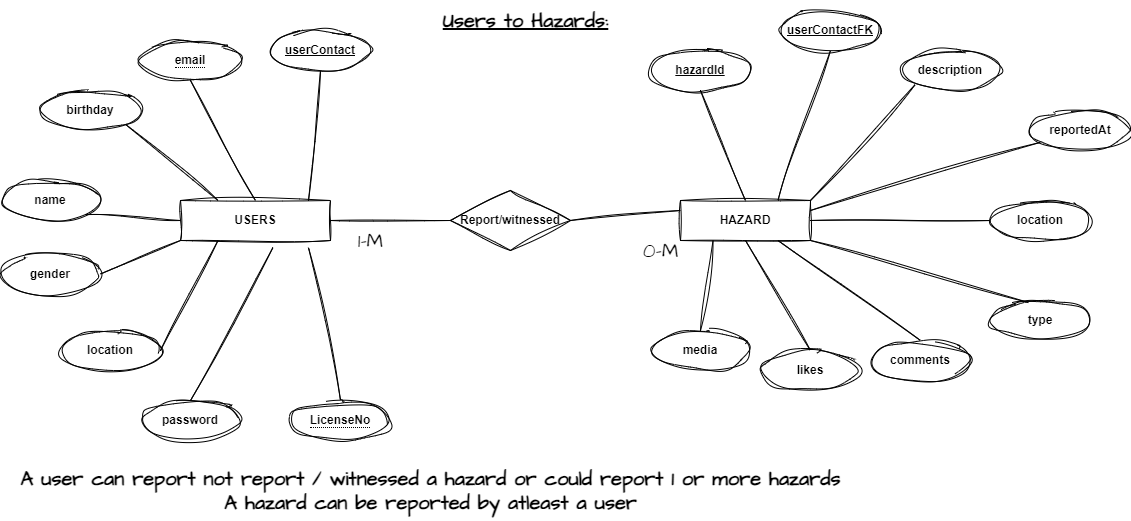
* **Users to RoadSigns**:



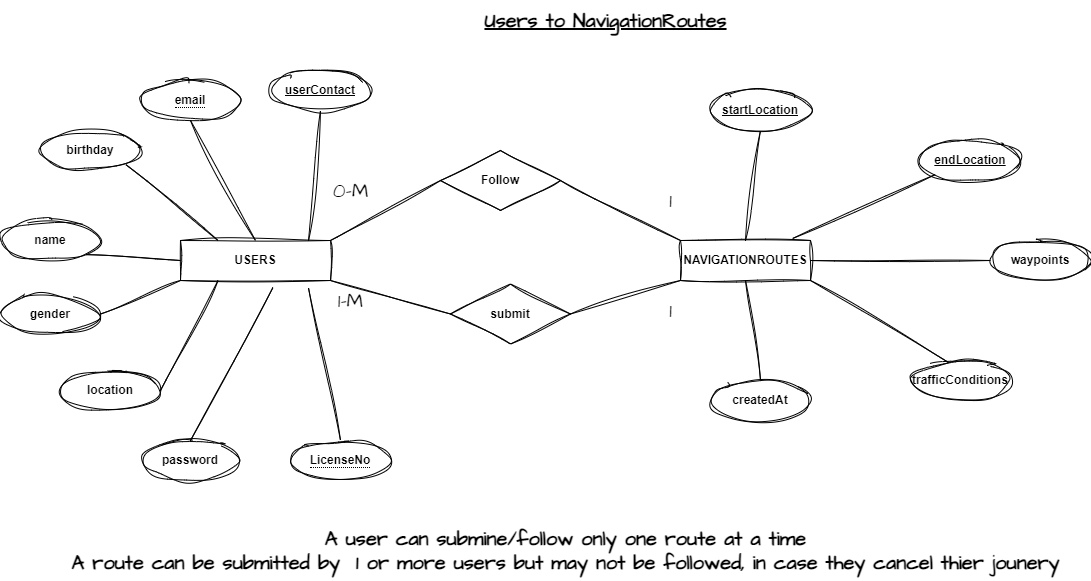
* **Users to TrafficLights**:



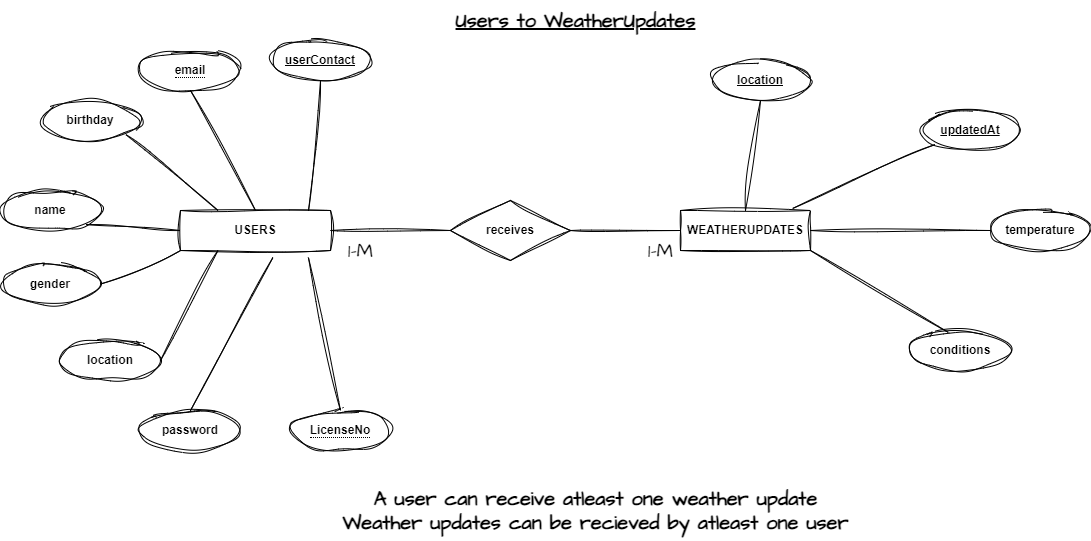
* **Users to Hazards**:



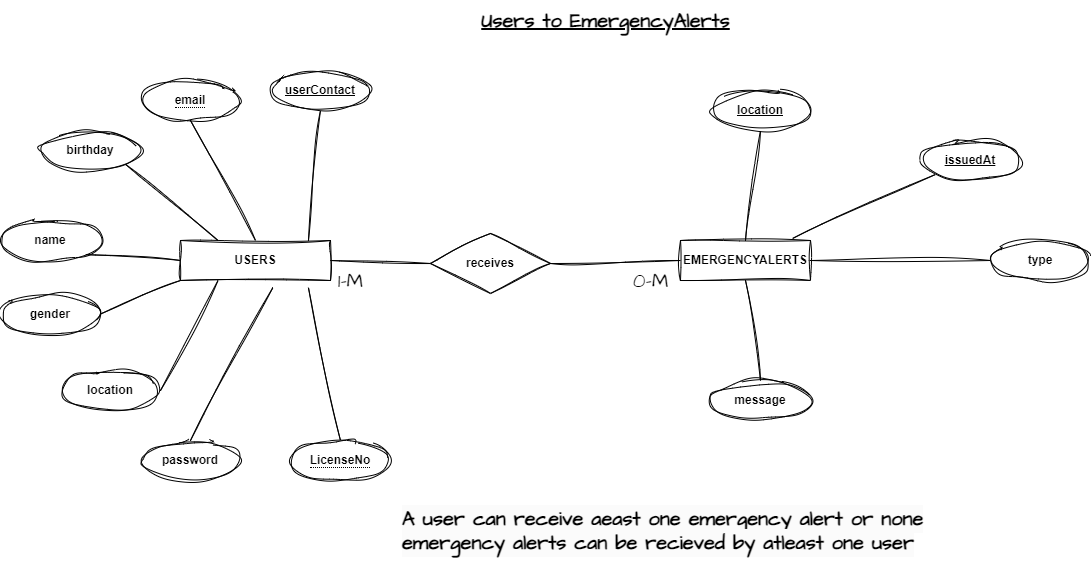
* **Users to NavigationRoutes**:



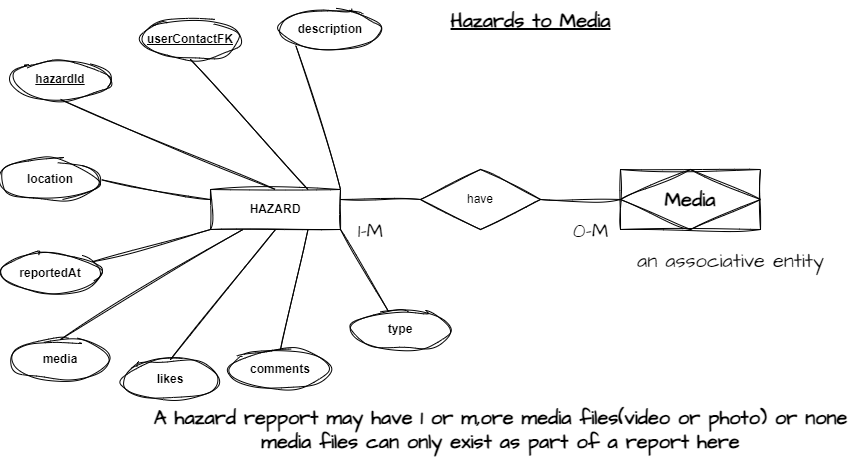
* **Users to WeatherUpdates**:



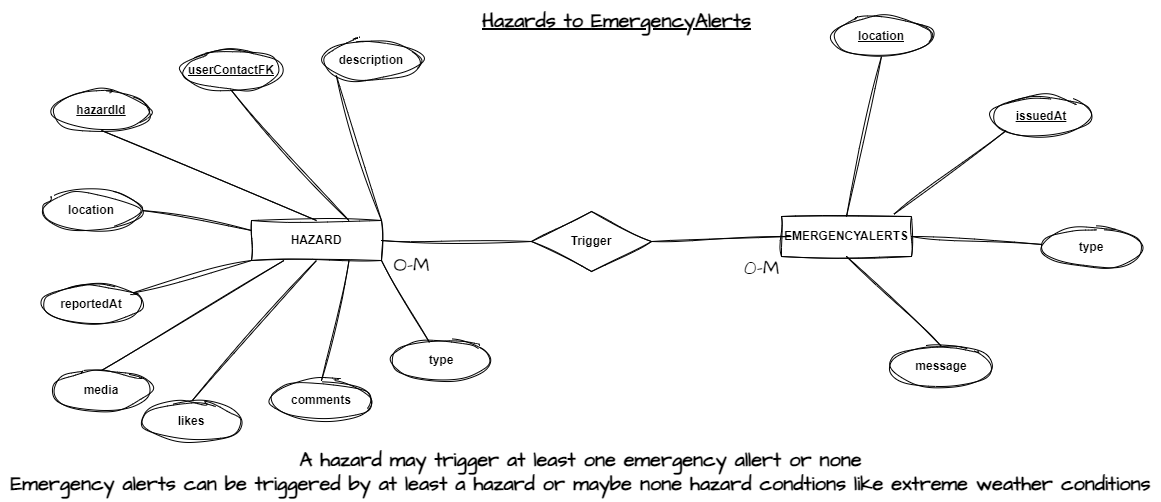
* **Users to EmergencyAlerts**:



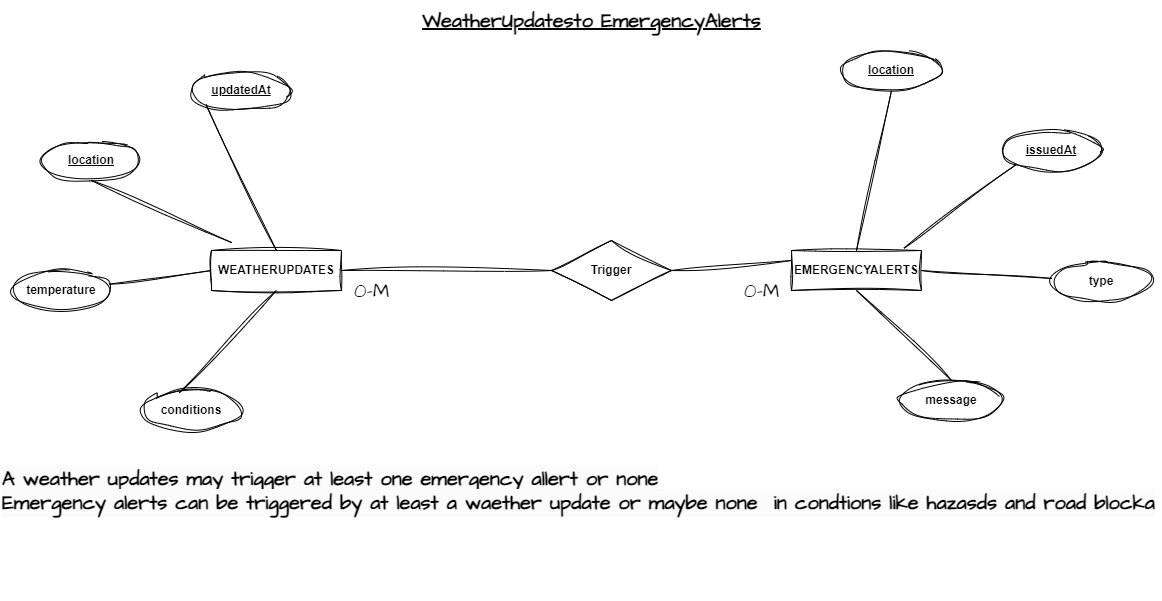
* **Hazards to Media**:

****

* **Hazards to Emergencyalert**:



* **WeatherUpdate to Emergencyalert**:



# 3. Database Implementation with Firebase

Firebase provides a flexible and scalable NoSQL cloud database that supports real-time data synchronization and offline capabilities. The collections and documents designed will be implemented in Firebase Firestore, enabling efficient data management and real-time updates to the application users.

## 3.1 Steps for Implementation

### 1. Set up Firebase Project:

- Create a Firebase project in the Firebase Console.

- Enable Firestore database and configure security rules.

### 2. Define Collections and Documents:

- Set up the defined collections (Users, RoadSigns, TrafficLights, Hazards, NavigationRoutes, WeatherUpdates, EmergencyAlerts).

- Implement the fields as specified in the design.

### 3. Integrate Firebase with the Application:

- Use Firebase SDKs to integrate Firestore into the mobile application.

- Implement authentication using Firebase Authentication.

- Set up real-time listeners for collections to receive updates.

### 4. Implement Data CRUD Operations:

- Create, read, update, and delete operations for each collection.

- Ensure data validation and security rules are applied.

### 5. Testing and Deployment:

- Test the database interactions within the application.

- Deploy the Firebase Firestore rules and monitor usage.

# Conclusion

The database design for the road sign and road state mobile notification application ensures efficient data management and real-time updates, enhancing driver safety and navigation experience. By leveraging Firebase Firestore, the application benefits from a scalable, flexible, and real-time backend, ensuring robust performance and reliability. This design lays a solid foundation for further development and integration, paving the way for a fully functional and user-friendly application.

This comprehensive design and implementation plan provides a clear roadmap for building the database component of the application, ensuring all critical data is efficiently managed and readily available to users.