REPUBLIC OF CAMEROON
PEACE-WORK-FATHERLAND
UNIVERSITY OF BUEA
BUEA, SOUTH-WEST REGION
P.O BOX 63.

RÉPUBLIQUE DU CAMEROUN
PAIX-TRAVAIL-PATRIE
UNIVERSITÉ DE BUEA
BUEA, RÉGION DU SUD-OUEST
B.P. 63.

FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING
SOFTWARE ENGINEERING

CEF440: Internet and Mobile Programming

DESIGN AND IMPLEMENTATION OF A DATABSE FOR A DISASTER MANAGEMENT MOBILE APPLICATION (TASK 6)

Course Facilitator:
Dr. NKEMENI Valery, PhD

Date Issued:
17th June 2024

Presented by:

GROUP 2

NAMES	MATRICULE
QUINUEL TABOT NDIP-AGBOR	FE21A300
SIRRI THERESIA ANYE	FE21A306
NGONCHI RAMATOU YOLAND	FE21A260
CHE BLAISE NJI	FE21A157
LIMA CHARLES	FE21A225

Table of Content

1. INTRODUCTION

In this phase, we will design and implement a database for a disaster management system using Firebase Firestore. Our system is intended to manage various types of data, including user profiles, emergency responder contacts, real-time alerts, incident reports, locations, language preferences, and preparedness resources.

Overview

The disaster management system aims to:

- Enable users to report incidents and receive real-time alerts.
- Store and manage contact information for emergency responders.
- Track user locations during incidents.
- Support multilingual communication.
- Provide users with access to preparedness resources.

2. PROBLEM STATEMENT/ REQUIREMENT: DESIGNING DATABASE FOR A DISASTER MANAGEMENT MOBILE APPLICATION

We are tasked with designing and implementing a disaster management mobile application database that assists users during emergencies. The application will provide real-time information, facilitate communication, and enhance coordination among users, emergency responders, and relevant authorities. Below are the key requirements:

2.1. **User Registration and Authentication**:

Users can create accounts and log in securely.

Implement authentication mechanisms (e.g., OAuth, JWT) to protect user data.

2.2. **Entities and Relationships**:

User: Represents individuals using the app. Users can report incidents, receive alerts, and manage their profiles.

Emergency Responder Contacts: Stores contact information for emergency responders (e.g., police, fire department, medical services).

Alert/Notification: Allows sending emergency alerts to users based on their location and preferences.

Location: Tracks user location during incidents.

Incident Report: Captures details about incidents (e.g., natural disasters, accidents).

Language: Supports multilingual communication.

2.3. **Functionality**:

Incident Reporting: Users can submit incident reports with relevant details (location, severity, type).

Alerts and Notifications: Send real-time alerts to users based on their proximity to incidents.

Emergency Contacts: Users can access emergency responder contacts.

Multilingual Support: Provide information in different languages.

Map Integration: Display incident locations on a map.

User Profiles: Allow users to manage their profiles and preferences.

3. CONCEPTUAL DESIGN: ER DIAGRAM

In the context of a **disaster management mobile application**, an **Entity-Relationship Diagram (ERD)** is a visual representation of different entities within the system and how they relate to each other. Below is a simplified representation of the entities and their relationships:

3.1. **User**: **Entities**:

User: Represents individuals using the app. Users can create accounts, log in, and manage their profiles.

Attributes: UserID (Primary Key), Username, Email, Password, LanguagePreference

- Relationships: One-to-Many with Incident (User can report multiple incidents)

3.2. **Emergency Responder Contacts**:

Emergency Responder Contacts: Stores contact information for emergency responders (e.g., police, fire department, medical services).

Attributes: ResponderID (Primary Key), ResponderName, ResponderType (e.g., police, fire, medical), Phone

Relationships: None (standalone entity)

3.3. **Alert/Notification**:

Alert/Notification: Allows sending real-time emergency alerts to users based on their location and preferences.

Attributes: AlertID (Primary Key), Message, Timestamp, Location

Relationships: Many-to-One with User (User receives multiple alerts)

3.4. **Location**:

Location: Tracks user location during incidents.

Attributes: LocationID (Primary Key), Address

Relationships: One-to-Many with Incident (Incidents occur at specific locations)

3.5. **Incident Report**:

Incident Report: Captures details about incidents (e.g., natural disasters, accidents).

Attributes: IncidentID (Primary Key), Type (e.g., earthquake, flood), Severity, Description, Timestamp, photo, video.

Relationships: Many-to-One with User (User reports incidents), Many-to-One with Location (Incident occurs at a specific location)

3.6. **Language**:

Language: Supports multilingual communication.

Attributes: LanguageID (Primary Key), LanguageName

Relationships: Many-to-Many with User (User preferences for multilingual support)

3.7. **PreparednessResources**

Attributes: **ResourceID**: A unique identifier for each preparedness resource.

ResourceType: Describes the type of resource (e.g., guide, checklist, emergency kit).

Description: Provides details about the resource (e.g., content, steps, recommendations).

Relationships: User-PreparednessResources: Users can access multiple preparedness resources and each resource can be accessed by multiple users.

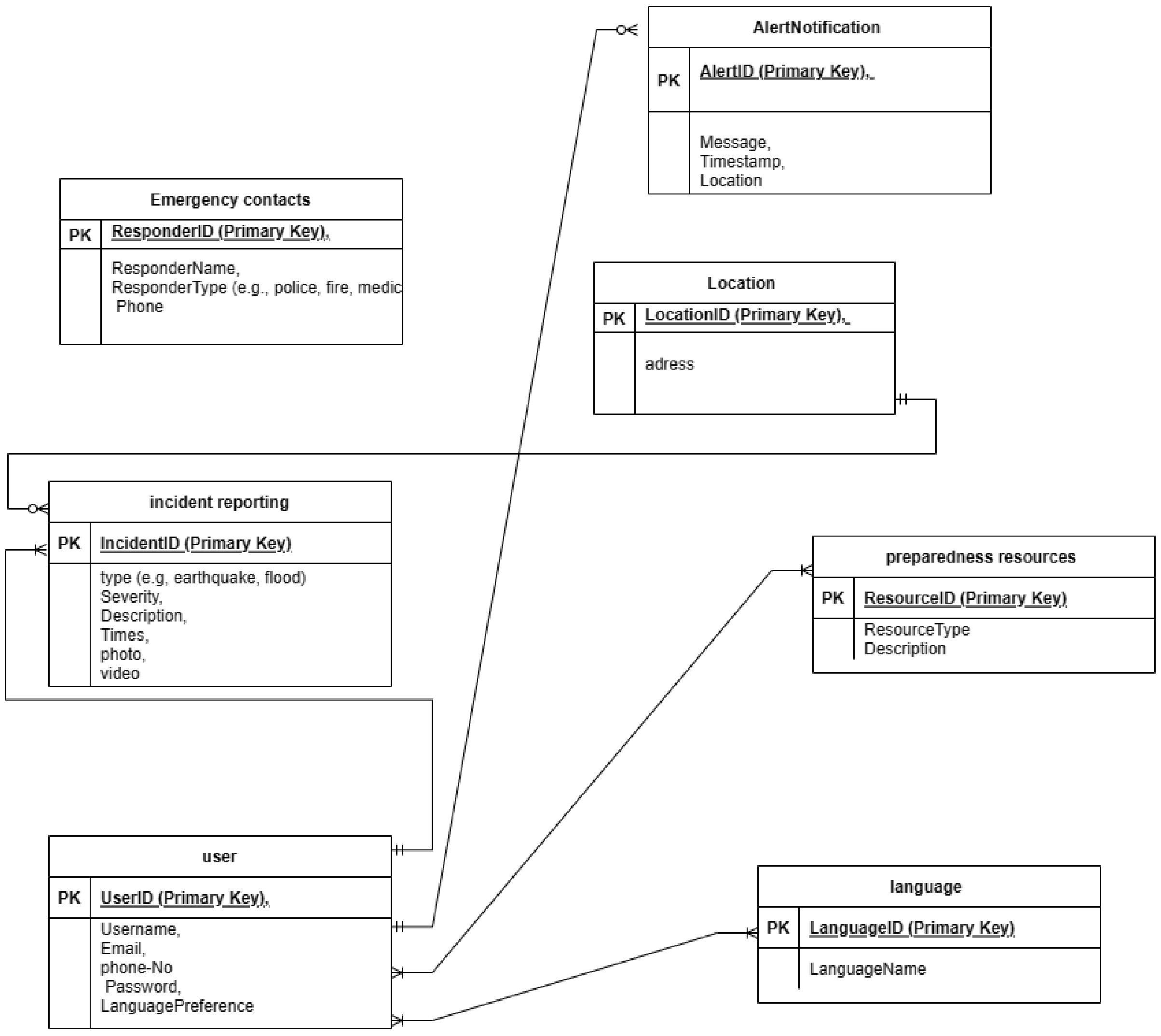


Figure 1: ER Diagram

4. IMPLEMENTATION: FIREBASE (LOGICAL AND PHYSICAL DESIGN)

Firebase is a cloud-based platform that offers a suite of services for mobile and web app development. One of its key components is the Firebase Realtime Database, which is a NoSQL database that provides real-time data synchronization.

Firebase Realtime Database provides a powerful, flexible, and easy-to-use NoSQL database with real-time data synchronization, making it ideal for building modern mobile and web applications that require instant data updates and seamless user experiences

i. Data Modeling:

> Document Structure: How will we structured our documents? We define the fields and their types (string, number, boolean, etc.) that represent your data.

> Relationships: How will you link documents together? You'll use references within documents to establish relationships. For example, a User document might contain a reference to a Post document.

ii. Data Flow and Relationships:

* How Data is Accessed and Updated: How does our app interact with the database? What operations (read, write, delete) are needed?

* Data Relationships: What are the relationships between different types of data? Think about how users, events, resources, or other entities in your app are related.

iii. Security Rules:

* Data Access Control: Who has access to different parts of your database? You need to define security rules in Firebase to control read and write permissions, ensuring data integrity and security.

iv. Entities and Attributes

1. User
   - Attributes: UserID, Username, Email, Password, LanguagePreference
2. Emergency Responder Contacts
   - Attributes: ResponderID, ResponderName, ResponderType, Phone
   - Relationships: None
3. Alert/Notification
   - Attributes: AlertID, Message, Timestamp, Location
   - Relationships: Many-to-One with User
4. Location
   - Attributes: LocationID, Address
   - Relationships: One-to-Many with Incident Report
5. Incident Report
   - Attributes: IncidentID, Type, Severity, Description, Timestamp
   - Relationships: Many-to-One with User, Many-to-One with Location
6. Language
   - Attributes: LanguageID, LanguageName
   - Relationships: Many-to-Many with User
7. Preparedness Resources
   - Attributes: ResourceID, ResourceType, Description
   - Relationships: Many-to-Many with User

v. Relationships

- User and Incident Report: One user can report multiple incidents.
- User and Alert/Notification: One user can receive multiple alerts.
- Incident Report and Location: Each incident occurs at a specific location.
- User and Language: Users can have preferences for multiple languages.
- User and Preparedness Resources: Users can access multiple preparedness resources and each resource can be accessed by multiple users.

vi. Mapping to Firebase Firestore

User Collection

- Attributes:
  - UserID: Automatically generated by Firestore as Document ID
  - Username
  - Email
  - Password
  - LanguagePreference: Array of LanguageIDs
- Subcollections:
  - IncidentReports (Documents representing individual incidents reported by the user)
  - Alerts (Documents representing alerts received by the user)
  - PreparednessResources (Documents representing resources accessed by the user)

Emergency Responder Contacts Collection

- Attributes:
  - ResponderID: Automatically generated by Firestore as Document ID
  - ResponderName
  - ResponderType
  - Phone

Alert/Notification Collection

- Attributes:
  - AlertID: Automatically generated by Firestore as Document ID
  - Message
  - Timestamp
  - Location: Reference to Location document

Location Collection

- Attributes:

- LocationID: Automatically generated by Firestore as Document ID
  - Address

Incident Report Collection

- Attributes:
  - IncidentID: Automatically generated by Firestore as Document ID
  - Type
  - Severity
  - Description
  - Timestamp
  - UserID: Reference to User document
  - LocationID: Reference to Location document

Language Collection

- Attributes:
  - LanguageID: Automatically generated by Firestore as Document ID
  - LanguageName

Preparedness Resources Collection

- Attributes:
  - ResourceID: Automatically generated by Firestore as Document ID
  - ResourceType
  - Description
- Subcollections:
  - Users (References to User documents accessing this resource)

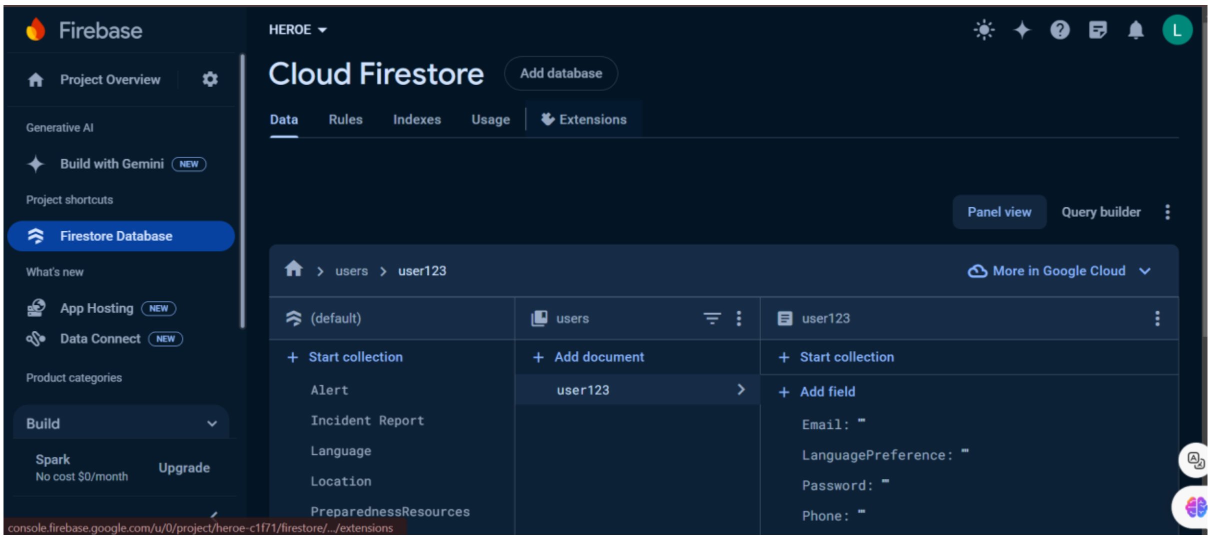
vii. Explanation of the Mapping

- User Collection: Stores user profiles with subcollections for incident reports, alerts, and accessed resources.
- Emergency Responder Contacts: Standalone collection for storing contact information.
- Alert/Notification Collection: Stores alerts with references to user and location.
- Location Collection: Stores locations with references in incidents and alerts.
- Incident Report Collection: Stores incident details with references to user and location.
- Language Collection: Stores language preferences.
- Preparedness Resources: Stores resource details with subcollections for users accessing these resources.

5. RESULTS

A. Collection and Document

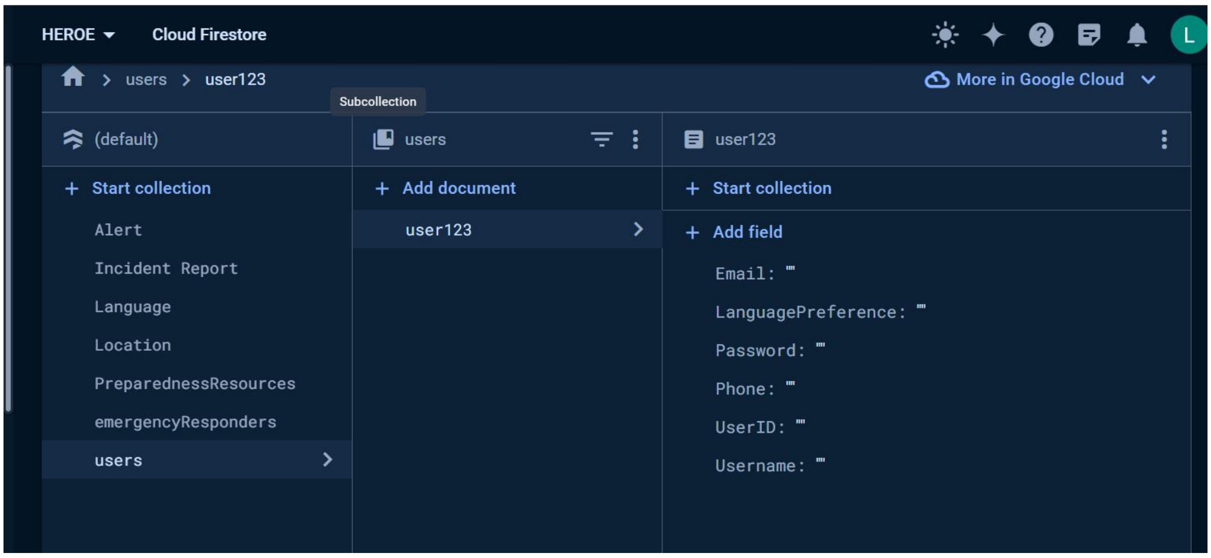
It has the results gotten from each collection and Documents that have their various attributes



users > user123

(default)	users	user123
+ Start collection	+ Add document	+ Start collection
Alert	user123	+ Add field
Incident Report		Email: ""
Language		LanguagePreference: ""
Location		Password: ""
PreparednessResources		Phone: ""

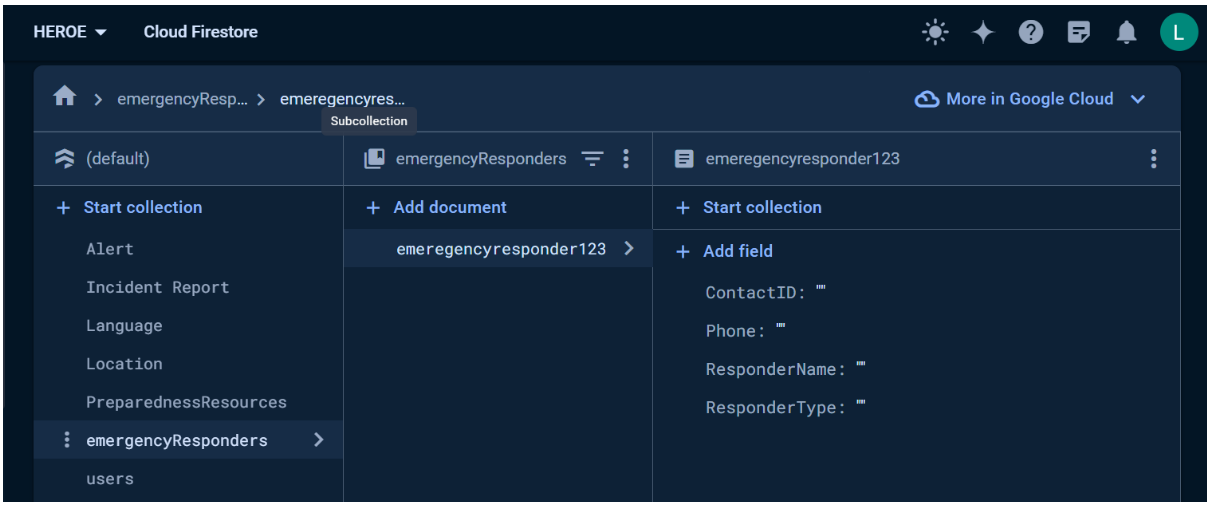
i. user/admin collection



users > user123

(default)	users	user123
+ Start collection	+ Add document	+ Start collection
Alert	user123	+ Add field
Incident Report		Email: ""
Language		LanguagePreference: ""
Location		Password: ""
PreparednessResources		Phone: ""
emergencyResponders		UserID: ""
users		Username: ""

ii. Emergency Responders Collection



HEROE Cloud Firestore

emergencyResp... > emeregencyres... Subcollection

More in Google Cloud

(default) | emergencyResponders | emeregencyresponder123

Start collection | Add document | Start collection

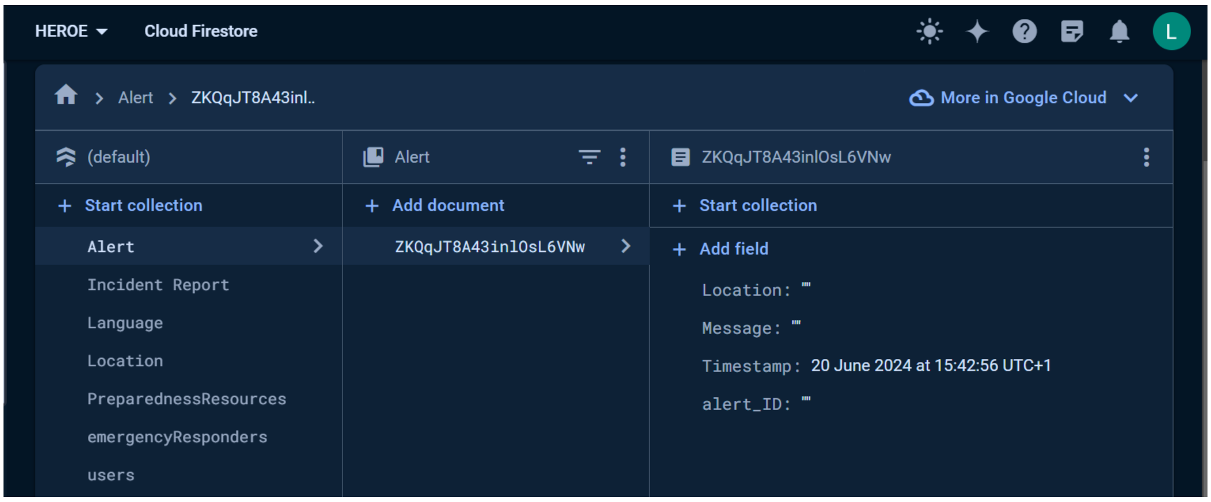
Alert
Incident Report
Language
Location
PreparednessResources
emergencyResponders
users

emeregencyresponder123

Add field

ContactID: ""
Phone: ""
ResponderName: ""
ResponderType: ""

iii. Alert/Notification Collection



HEROE Cloud Firestore

Alert > ZKQqJT8A43inl..

More in Google Cloud

(default) | Alert | ZKQqJT8A43inlOsL6VNw

Start collection | Add document | Start collection

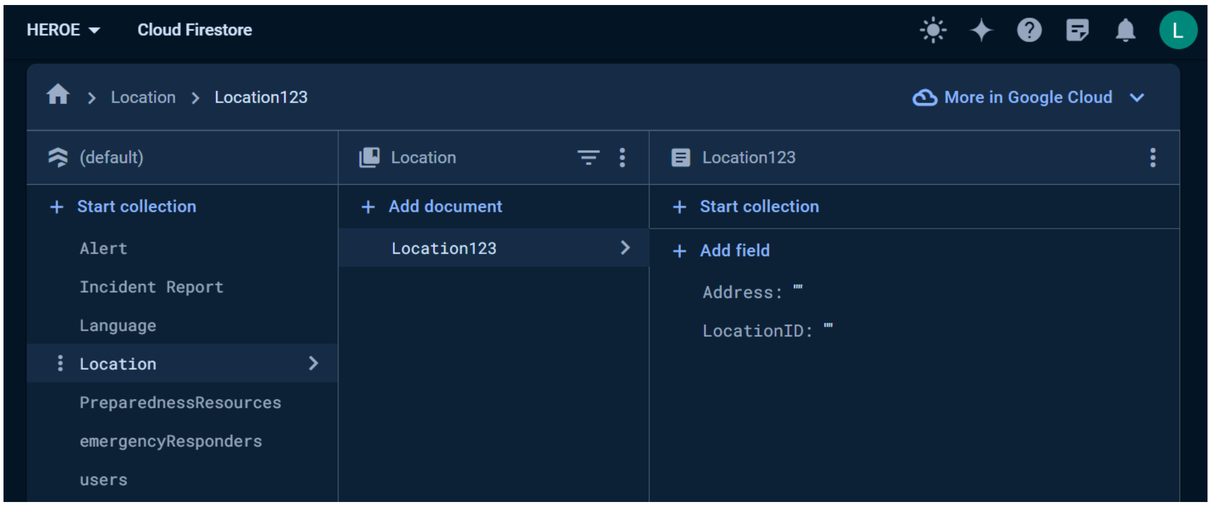
Alert
Incident Report
Language
Location
PreparednessResources
emergencyResponders
users

ZKQqJT8A43inlOsL6VNw

Add field

Location: ""
Message: ""
Timestamp: 20 June 2024 at 15:42:56 UTC+1
alert_ID: ""

iv. Location Collection



HEROE Cloud Firestore

Location > Location123

More in Google Cloud

(default) | Location | Location123

Start collection | Add document | Start collection

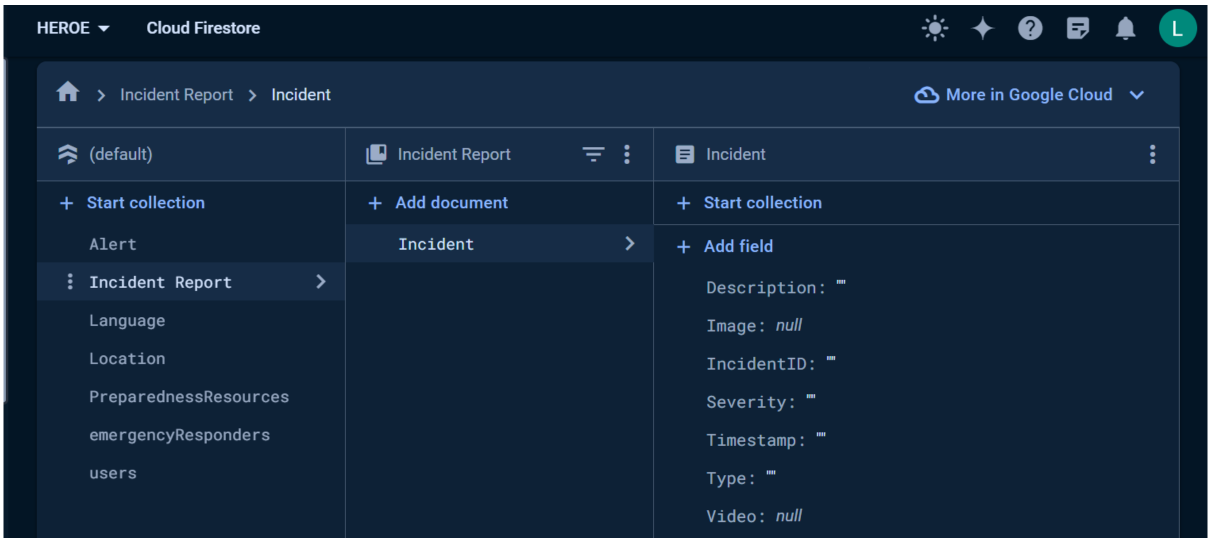
Alert
Incident Report
Language
Location
PreparednessResources
emergencyResponders
users

Location123

Add field

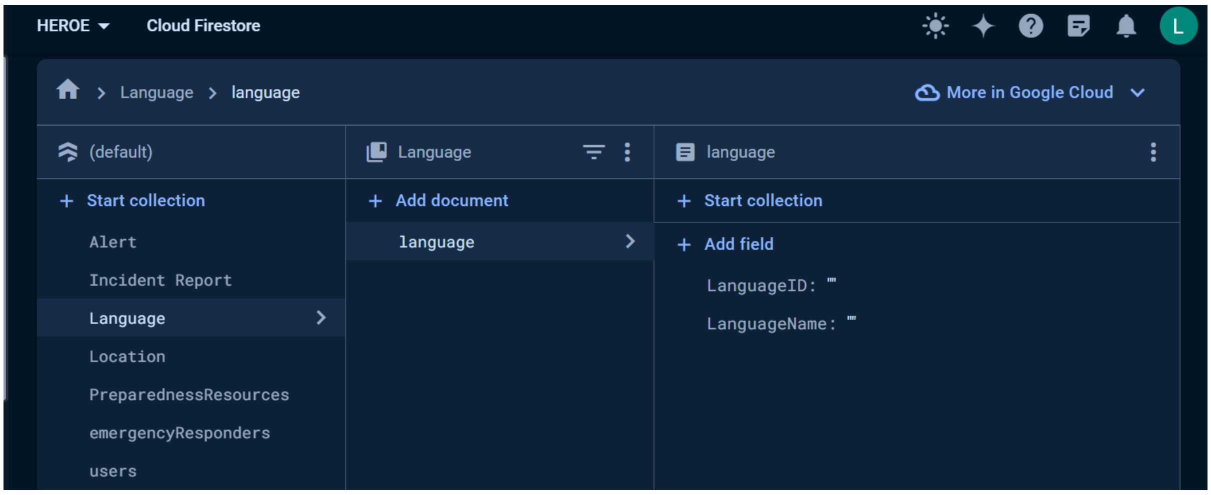
Address: ""
LocationID: ""

v. Incident Report Collection



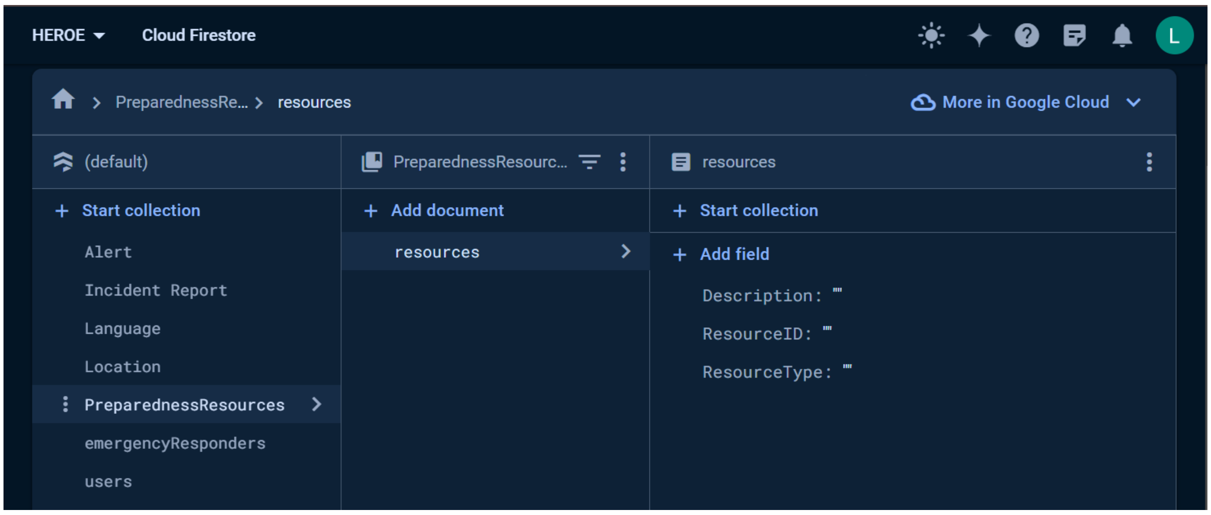
```
Description: ""
Image: null
IncidentID: ""
Severity: ""
Timestamp: ""
Type: ""
Video: null
```

vi. Language Collection



```
LanguageID: ""
LanguageName: ""
```

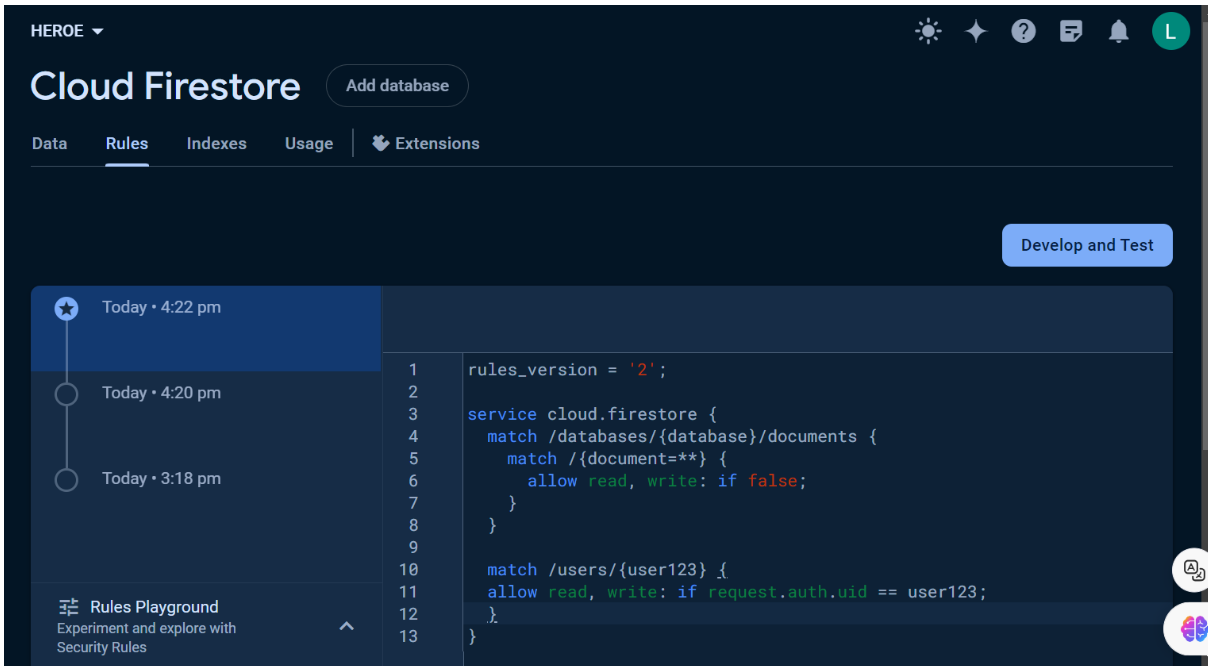
vii. Preparedness Resources



```
Description: ""
ResourceID: ""
ResourceType: ""
```

B. Rules

Set of rules on how users can read, write and perform other actions in the database



```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }

  match /users/{user123} {
  allow read, write: if request.auth.uid == user123;
  }
}
```

5. CONCLUSION

This report provides a detailed database design and implementation for a disaster management mobile application using Firebase Firestore. The design includes well-defined entities, attributes, and relationships, mapped effectively to Firestore collections and documents. This structure ensures efficient data management and real-time capabilities essential for a disaster management application.

6. REFERENCES

A) Firebase: https://console.firebase.google.com/

B) **Building Real-time Disaster Response Applications with Firebase:** https://firebase.google.com/docs/firestore/disaster-recovery

C) Firebase Medium Review: https://medium.com/@louisjaphethkouassi/how-to-use-google-firebase-realtime-database-9df362548de7