#### 

UNIVERSITY OF BUEA REPUBLIC OF CAMEROON

Buea, South West Region PEACE-WORK-FATHERLAND

Cameroon

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

**Task 6: Database Design and Implementation**

Course Master: Dr Valery Nkemeni

CEF440:Internet Programming and Mobile Programming

GROUP 22

|  |  |
| --- | --- |
| NAMES | MATRICULE |
| NGOUH KAMBI MARCBRYAN | FE22A264 |
| AMBASSA LISE-ASTRID | FE22A146 |
| EPIE-NGOLE MCCAULEY | FE22A445 |
| EPANDA RICHARD JUNIOR | FE22A206 |
| NKIANGWA NKEGOA KAREL | FE22A277 |

**Table of Contents**

**1. Introduction**………………………………………………………………………3

**2. Data Elements** 3

Core Data Entities 3

**3.Conceptual Design** 4

Database Requirements 5

Firebase Firestore Structure 5

**4. ER Diagram** 6

Entities and Descriptions 7

Relationships and Multiplicities 7

**5. Database Implementation** 8

Firestore Setup 8

**6. Connecting Database to Backend** 9

**7. Conclusion** 10

**1. Introduction**

This report presents a comprehensive overview of the database design and implementation for Motosure, a mobile application dedicated to enhancing road safety through real-time notifications of road signs and road conditions. The primary goal of the database is to effectively store, manage, and synchronize data related to users, road hazard reports, alerts, road signs, and user preferences.

The database system is designed to support key app functionalities such as immediate hazard reporting, dissemination of safety alerts, and personalization of notifications. The choice of a cloud-based NoSQL database (Firebase Firestore) ensures scalability, real-time synchronization, and offline access, all crucial for an app aimed at improving road safety across urban regions with varying network conditions.

**2. Data Elements**

* **Core Data Entities**

The database consists of six main entities, each capturing essential information required by Motosure’s functionality:

* **User:** This entity represents each individual user of the Motosure app. It stores unique identifiers, contact details, role information (e.g., driver, admin), and account creation timestamps. Maintaining accurate user data is fundamental for associating reports and preferences with the correct individual and ensuring secure access control.
* **UserPreference:** This entity records individual users' preferences regarding the types of notifications they wish to receive, such as alerts for accidents or adverse weather conditions. Allowing users to customize their notification settings enhances user experience by delivering relevant and timely information tailored to their needs.
* **RoadReport:** This entity captures detailed information about hazards or road state conditions reported by users, including the type of hazard (e.g., pothole, accident), geographic location, images, and report status. Real-time submission of such data is critical for maintaining an up-to-date hazard map and facilitating timely responses.
* **Alert:** Alerts are system-generated notifications created in response to certain road reports deemed significant enough to warrant notifying users. Each alert contains information on the type of hazard, severity level, location, and active status, enabling the app to prioritize and display critical warnings effectively.
* **RoadSign:** This entity stores metadata related to road signs, including sign names, categories (such as “warning” or “regulatory”), descriptions, and associated images. This data supports the app’s educational and navigation features, assisting drivers in understanding road rules and upcoming conditions.
* **Routes:** This entity stores information related to the routes suggessted by the system in situations of high traffic levels or other harzards, on a route a user is currently on.

**3. Conceptual Design**

* **Database Requirements**

To meet the functional goals of Motosure, the database design adheres to several key requirements:

* **Real-Time Synchronization:** The app must support immediate updates for hazard reports and alerts so that users receive the most current information without delay. This real-time sync is vital for road safety where timely warnings can prevent accidents.
* **Offline Support:** Considering that network connectivity can be inconsistent, especially in certain urban or rural areas in Cameroon, the app caches important data such as road sign information locally. This allows the app to function and provide critical information even when offline.
* **Scalability:** The database is built to handle thousands of concurrent users, particularly in urban regions with dense traffic and frequent reporting. Efficient indexing and querying mechanisms ensure performance remains optimal as the user base grows.
* **Firebase Firestore Structure**

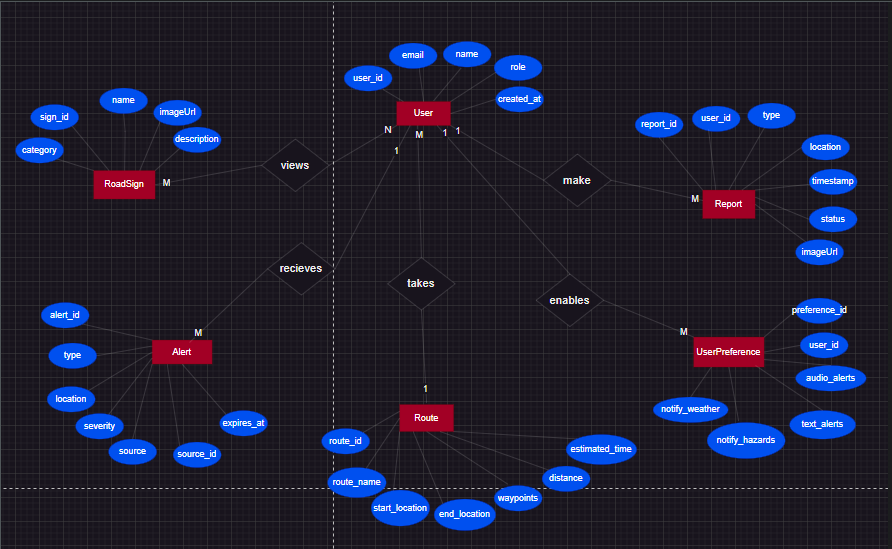
The Firestore database is organized into collections and documents representing the core entities:

* **Users Collection:** Stores user profiles with details such as email and account creation date.
* **RoadSigns Collection:** Contains road sign metadata, including names and categories.
* **Reports Collection:** Houses user-submitted hazard reports with location data and types.
* **Alerts Collection:** Contains system-generated alerts, marked with severity and active status flags.

This hierarchical, document-oriented structure facilitates efficient querying, real-time updates, and offline capabilities inherent in Firebase Firestore.

**4. ER Diagram and Relationships**

ER Diagram



This diagram was drawn using the draw.io software.

* **Entities and Their Descriptions**
* **User:** The central entity representing each Motosure app user.
* **UserPreference:** Stores customizable notification settings linked to users.
* **RoadReport:** Represents hazards or road conditions submitted by users.
* **Alert:** System-generated warnings associated with specific reports.

RoadSign: Repository of road sign information supporting app navigation.

* **Relationships and Multiplicities**
* **User to UserPrefs (1:N)**

Each user can have multiple preference entries to specify how they want to be notified. This one-to-many relationship allows the system to accommodate different notification types and user-specific settings flexibly.

* **User to RoadReport (1:N)**

Users can submit numerous road reports over time, enabling continuous and rich data input to monitor road conditions. This one-to-many relationship reflects the dynamic interaction between users and the app’s reporting feature.

* **RoadReport to Alert (1:1 or 0..1)**

Each hazard report may generate one alert if it meets severity criteria. However, not all reports lead to alerts, depending on the hazard's nature and current context. This optional one-to-one relationship ensures alerts are specifically tied to meaningful reports, preventing alert overload.

* **User to Route(1:1 or N:1 or 0:1)**

Each user may select one route suggested by the system or be on only one route at a time. Many users could also be on the same route at the same time, or no user may be in a route at a particular time.

* **User to Alert (1:0 or 1:N)**

Each user may recive no alert (probably at the time of just registering into the system), or the user may recieve many alerts.

* **User to RoadSign (1:0 or 1:N)**

Each user can view no road sign or many road signs

**5. Database Implementation**

* **Firestore Setup**

The Firebase Firestore database is initialized to provide a cloud-hosted NoSQL document database with built-in real-time synchronization, offline persistence, and scalability. Key implementation considerations include:

* **Security Rules:** Strict security policies are implemented to protect user data and enforce access control. For example, users are allowed to create reports but can only modify or delete their own reports, ensuring privacy and preventing misuse.
* **Collections and Documents:** Data is organized into collections reflecting the core entities. This approach supports efficient queries and real-time data streams for features like live alert updates and report submissions.
* **Offline Persistence:** Firestore’s offline capabilities are leveraged to cache data on the device, enabling the app to function smoothly even without an active internet connection. This is especially useful for users reporting hazards in areas with poor network coverage.

**6. Connecting Database to Backend**

* **Firebase Service Layer**

The app communicates with Firestore through a service layer that handles data operations such as submitting reports, fetching alerts, and syncing user preferences. This layer abstracts database interactions and maintains clean separation between the UI and data management.

* **Backend Logic with Cloud Functions**

Cloud Functions are used to implement backend logic that automatically responds to database events, such as:

Generating alerts when critical hazard reports are submitted.

Enforcing rate-limiting to prevent alert spam.

Updating the status of alerts based on evolving road conditions.

* **Data Flow Overview**

The flow of data in Motosure operates as follows:

* Users submit hazard reports via the mobile app.
* Reports are stored in Firestore.
* Cloud Functions monitor new reports and trigger alerts if needed.
* Alerts are stored back in Firestore.
* The app listens for active alerts and displays them to users in real time.

This cyclical data flow ensures timely, reliable communication between users and the backend system.

1. **Conclusion**

This database design and implementation approach ensures that Motosure is equipped with a robust, scalable, and secure backend infrastructure. It supports real-time hazard reporting and alerting with high availability and offline capabilities, fully aligned with the app’s mission to listen and respond to the road’s needs effectively.

* Link to the Figma File: <https://www.figma.com/design/0Zgvpq8JvnQYcquFHWLmuL/MotoSure?node-id=0-1&p=f&t=Ab1Xzn4Gg2BQmfce-0>
* Link to code’s Github Repo: [NgouhKambiMarcB/MotoSure: Road Sign and Road State Notification Mobile Application. Motosure is going to help users learn about common road signs and also provide users with real time notifications about the current condition of the road.](https://github.com/NgouhKambiMarcB/MotoSure)