****

**REPUBLIC OF CAMEROON**

PEACE-WORK-FATHERLAND

THE UNIVERSITY OF BUEA

P.O Box 63,

Buea,South West Region

Cameroon

Tel: (237) 674354327

Fax: (237) 3332 22 72

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**Department of Computer Engineering**

**Specialty: Software Engineering**

**DATABASE DESIGN AND IMPLEMENTATION OF A MOBILE APPLICATION FOR ARCHIVAL AND RETRIEVAL OF MISSING OBJECTS VIA IMAGE MATCHING**

**Presented by**

**GROUP 21**

|  |  |  |
| --- | --- | --- |
| S\N | NAME | MATRICULE |
| 1 | CHOMOH SYLVIA FONCHENALLAH | FE21A162 |
| 2 | AKENGNI KEANLI EMMANUEL | FE21A132 |
| 3 | MBI AYAMBA DIANNA | FE21A230 |
| 4 | NGAUEN NDJOMOU LOICE VANELLE | FE21A258 |
| 5 | NGOBA STEVE JONES NTONSG | FE21A259 |

**Dr. Eng. NKAMENI VALERIE**

**Course Supervisor**

**2rd Semester**

**2023/2024 Academic Year**

**Table of Contents**

Contents

[1. **Introduction** 3](#_Toc169850722)

[1.1. Objectives 3](#_Toc169850723)

[2. **Database Design** 3](#_Toc169850724)

[Entity-Relationship Diagram (ERD) 3](#_Toc169850725)

[Schema Design 4](#_Toc169850726)

[3. **IMPLEMENTATION** 6](#_Toc169850727)

3.1. Database………………………………………………………………………………………………………………………………6

3.2. Authentication…………………………………………………………………………………………………………………..7

3.3. Image Upload…………………………………………………………………………………………………………………….9

3.4 API and Data Fetching ………………………………………………………………………………………………………..10

3.5. Notification ……………………………………………………………………………………………………………………….12

[4. **Application Integration** 13](#_Toc169850729)

[6. **Conclusion** 16](#_Toc169850730)

# **Introduction**

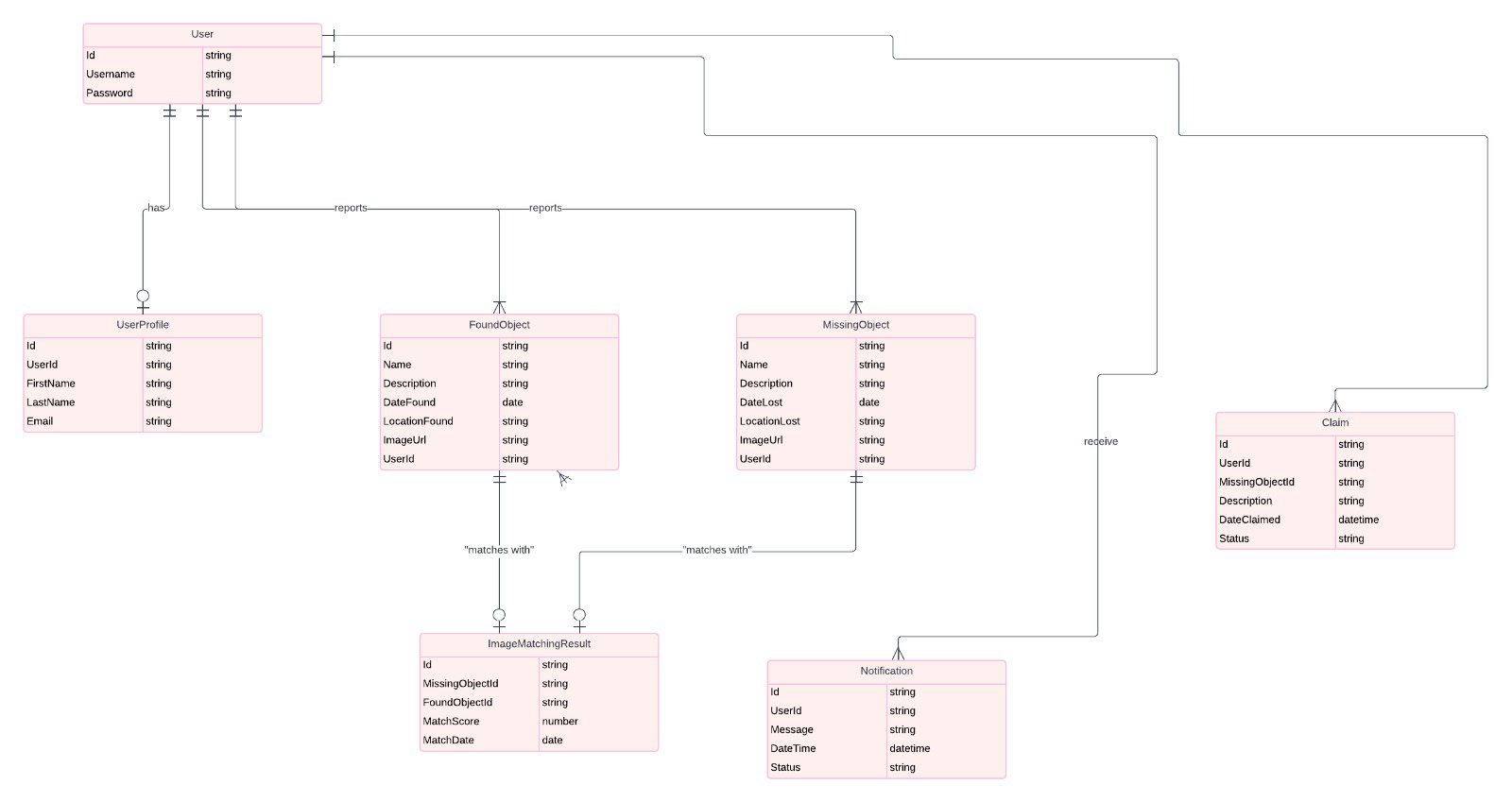
Database design is the process of organzing data in a structured way to make it easy to store, retrieve and manage informations. This data model include all the logical and physical design choices and storage parameters needed needed to generate the design.

## **Objectives**

* Efficiently store information about missing items.
* Enable quick retrieval of item details.
* Ensure data integrity and security.
* Support user-friendly data entry and search capabilities.

# **Database Design**

## 2.1 **Entity-Relationship Diagram (ERD)**



This is an ERD diagram, describing the entities, relationships, and cardinalities.

## **2.2 Schema Design**

Translating the ERD into a relational schema. Including table definitions, primary keys, foreign keys, and constraints.

* **Users Collection**

userID (String, Primary Key)

username (String)

email (String)

phone (String)

address (String)

profileImage (String, URL to the profile image)

* **FoundItems Collection**

itemID (String, Primary Key)

userID (String, Foreign Key referencing Users collection)

objectType (String)

natureColor (String)

locationFound (String)

description (String)

imageUrl (String, URL to the image of the found item)

timestamp (Timestamp)

status (String, e.g., 'Unclaimed', 'Claimed')

* **LostItems Collection**

itemID (String, Primary Key)

userID (String, Foreign Key referencing Users collection)

objectType (String)

natureColor (String)

locationLost (String)

description (String)

imageUrl (String, URL to the image of the lost item)

timestamp (Timestamp)

status (String, e.g., 'Lost', 'Found')

* **Claims Collection**

claimID (String, Primary Key)

itemID (String, Foreign Key referencing FoundItems collection)

userID (String, Foreign Key referencing Users collection)

details (String, Details provided by the user claiming the item)

imageUrl (String, URL to the image uploaded by the user claiming the item)

timestamp (Timestamp)

status (String, e.g., 'Pending', 'Approved', 'Rejected')

* **Match**

match\_id (primary key)

lost\_object\_id (foreign key references Lost\_Object.lost\_object\_id)

found\_object\_id (foreign key references Found\_Object.found\_object\_id)

match\_score (numerical value indicating the strength of the match based on the image matching algorithm)

match\_date (date the match was identified)

* **Notification**

notification\_id (string, primary key)

user\_id (foreign key references User.user\_id)

notification\_type (e.g., Lost Object Match, Claim Update)

message (string)

Status(String)

**Relationships:**

A User can have one User\_Profile.

A User can report many Lost\_Objects.

A User can find and report many Found\_Objects.

A User can make claims on many Lost\_Objects

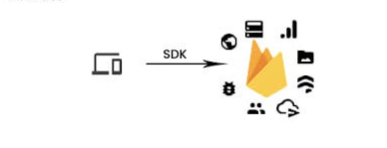
A Lost\_Object can have many Claim\_Objects. (Multiple users can claim the same lost object)

A Found\_Object can have many Claim\_Objects. (A found object can have claims from multiple users)

A User can receive many Notifications.

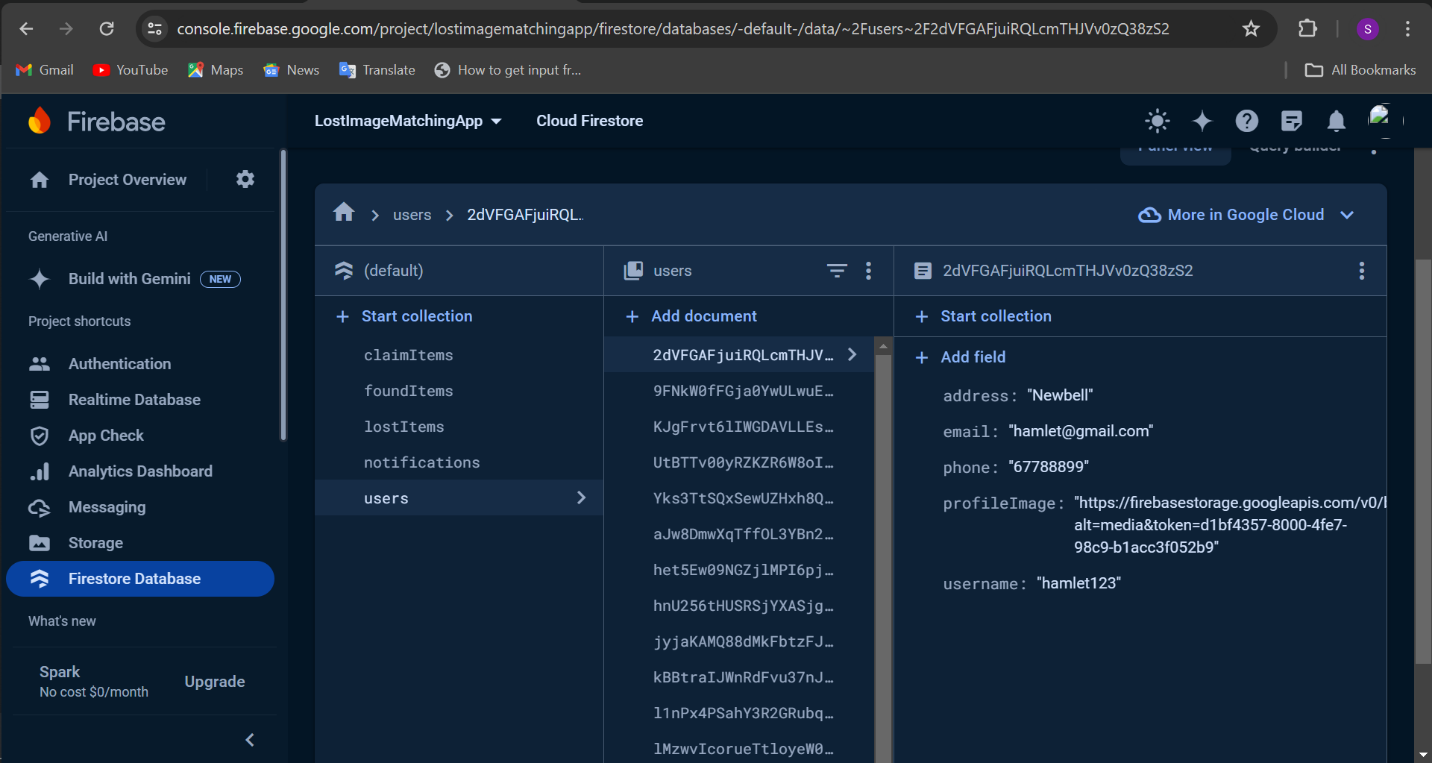
**3) DATABASE IMPLEMENTION**

Firebase is a comprehensive app development platform provided by Google, which includes various services such as Firebase Authentication, Firestore, Firebase Storage, and more. For our Lost Image Matching App, we utilized several of these Firebase services to build a robust and scalable backend infrastructure.

****

**3.1 Database**:

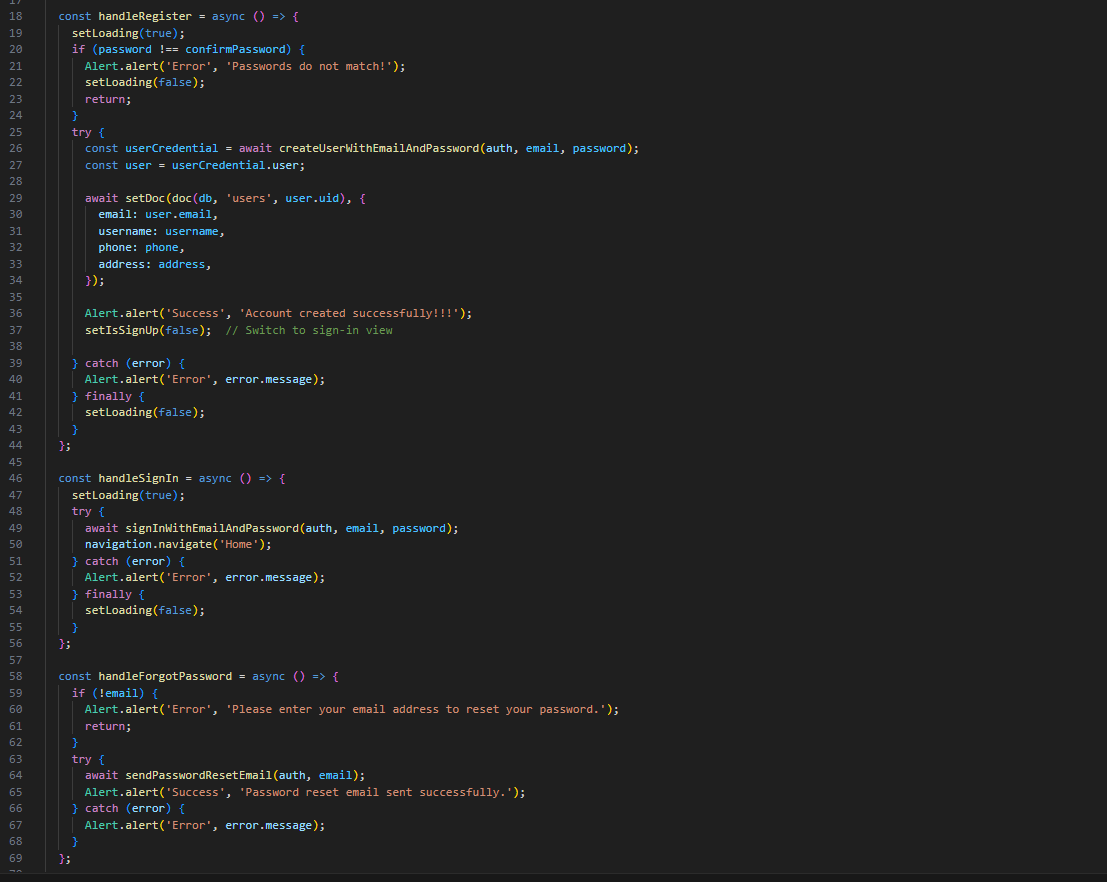
* Implemented using Firebase Firestore, a NoSQL cloud database that stores data in collections and documents.
* Collections used include users, foundItems, missingItems claims, and notifications.
* User details, found items, missing items, and notifications are stored and managed within these collections.



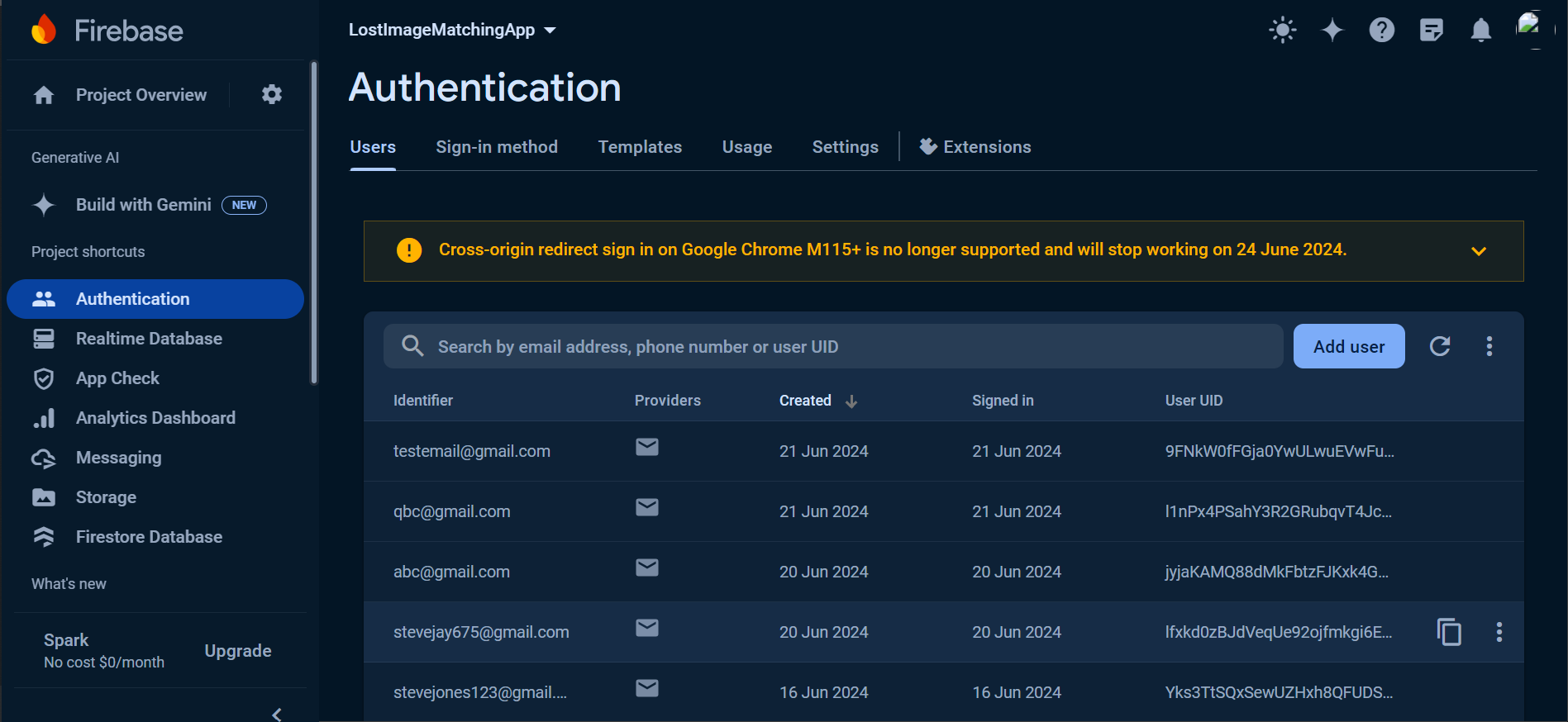
**3.2** **Authentication**:

* Managed using Firebase Authentication, providing user sign-up, sign-in, and password reset functionalities.
* Ensures secure access and management of user accounts

**Code to handle signUp and SignIn**



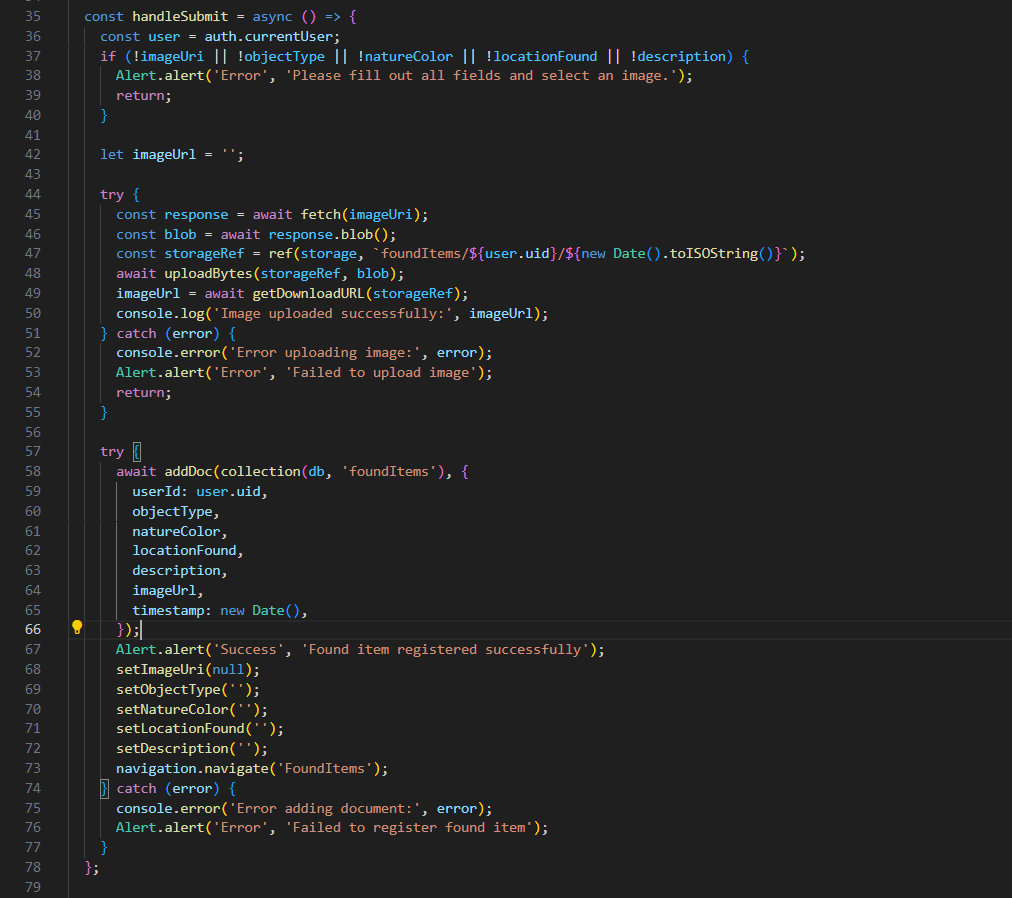
**Firebase Authentication console for Users**



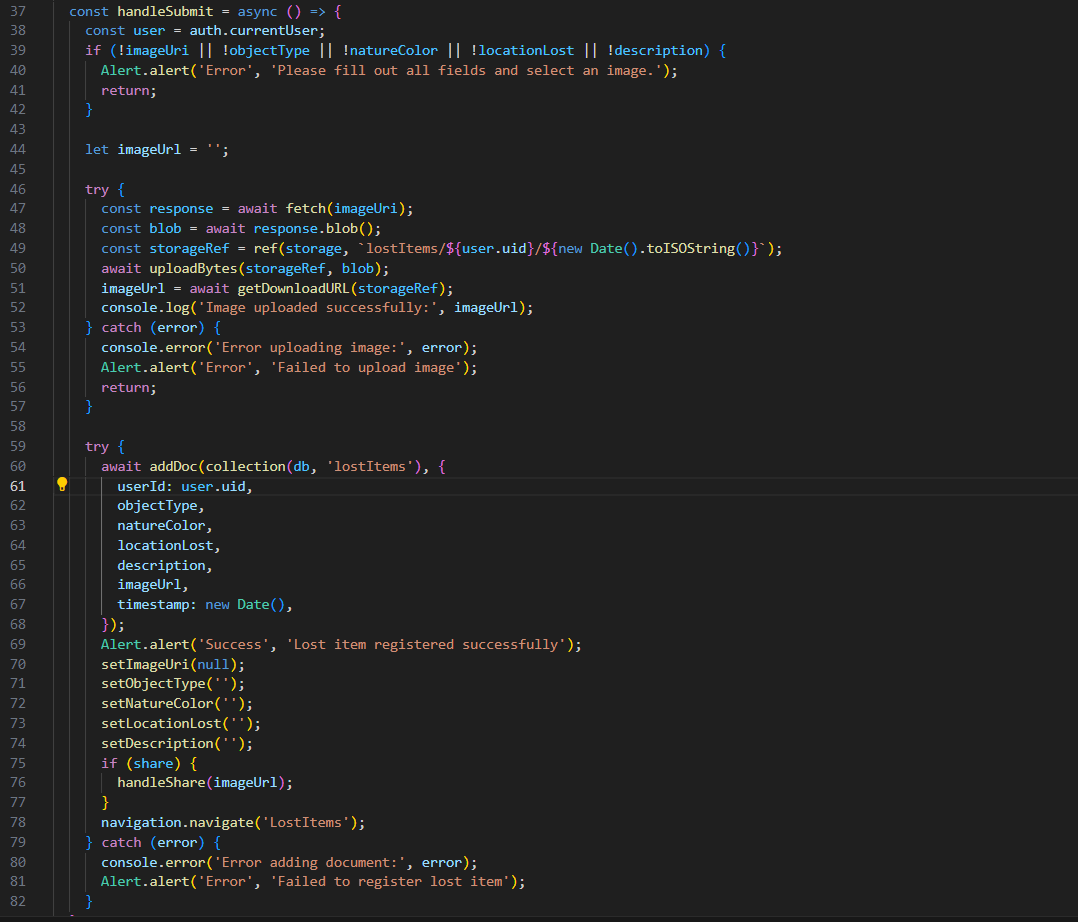
**3.3**  **Image Upload**:

* Images are uploaded to Firebase Storage, a cloud storage solution.
* URLs of uploaded images are stored in Firestore to link images with corresponding data entries.
* Images of the Found Items, Lost Items and the profile Images are uploaded and stored in the FoundItems, LostItems and the ProfileImages collections respectively

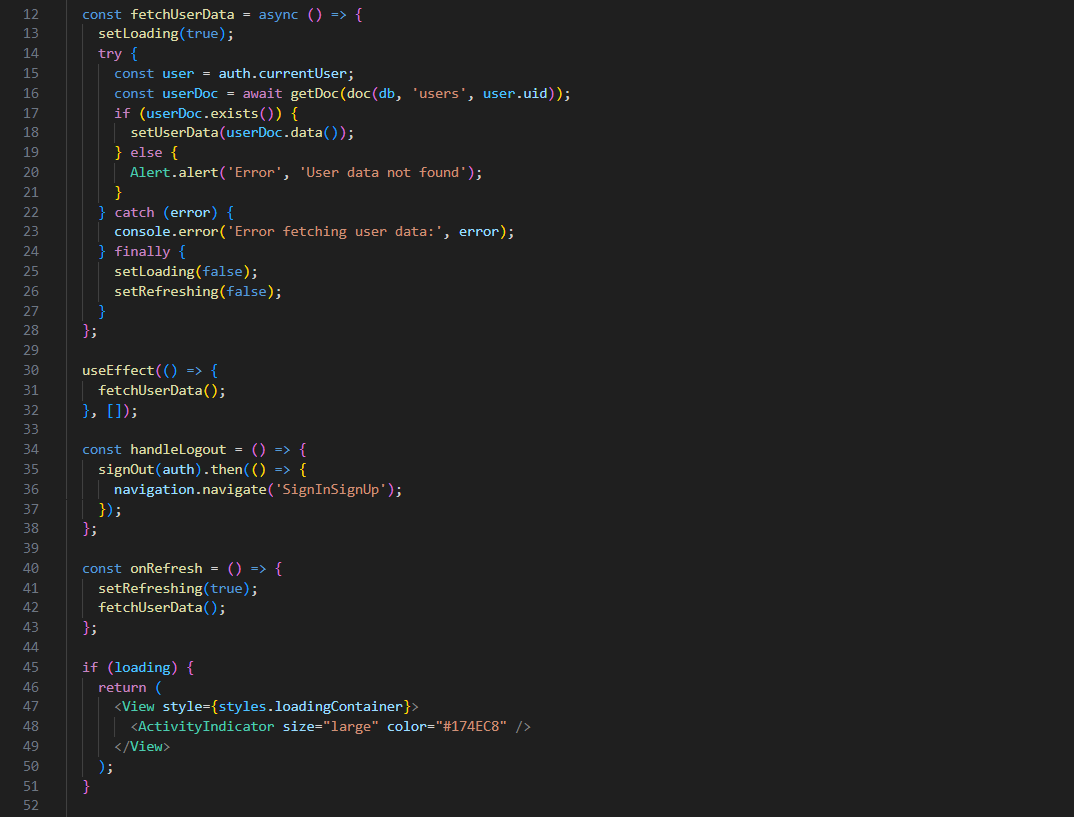
**Code to Upload Found Image to Database**

****

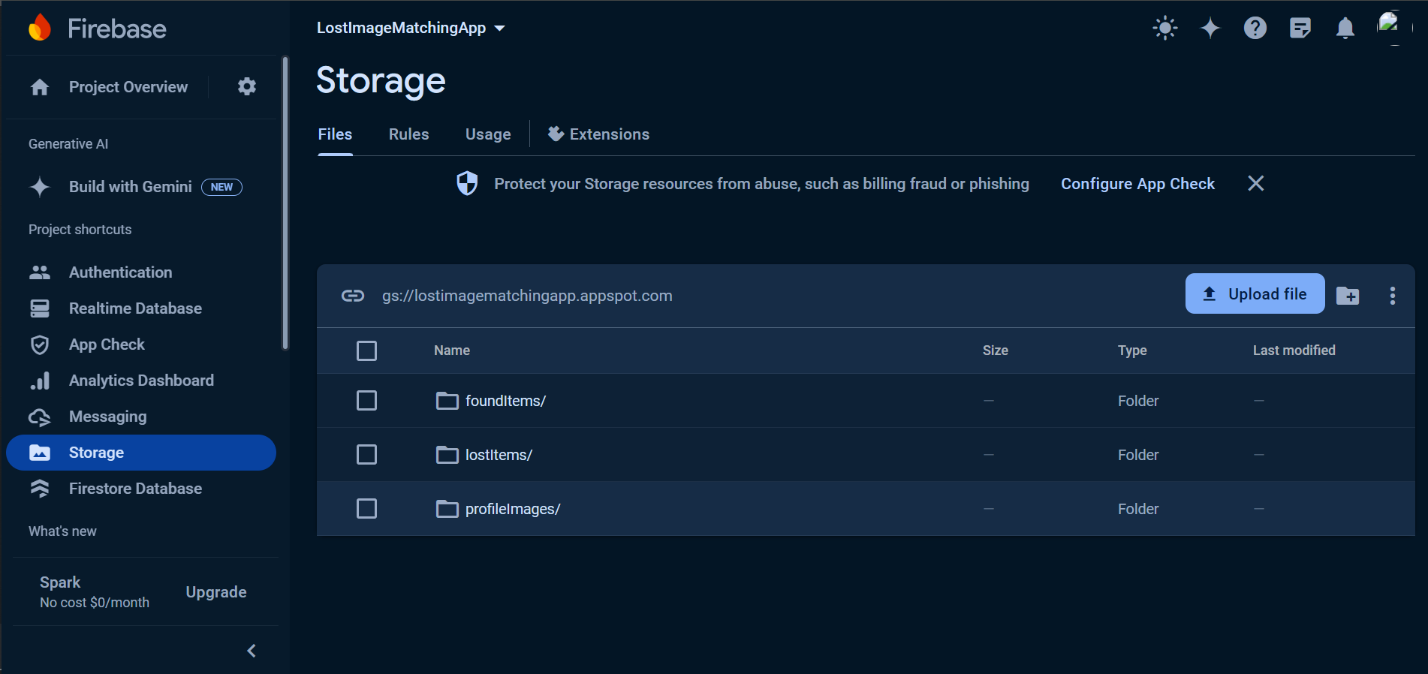
**Code to Upload Lost Image to Database**



**Code to Upload Profile Images to Firestore Database**



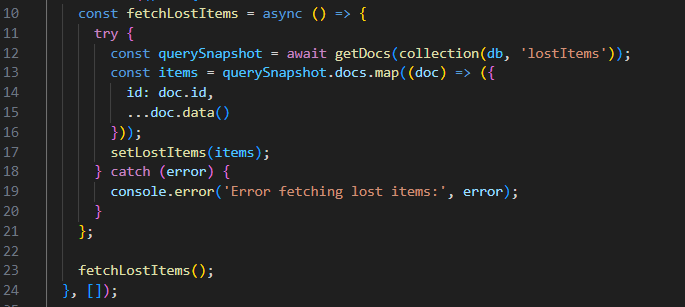
**Stored images in firebase(foundItems, lostItems, ProfileImages)**



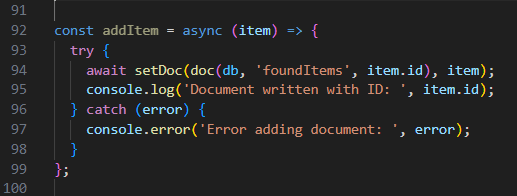
**3.4** **APIs and Data Fetching**:

* Firebase SDK (Software Development Kit) functions are used to interact with Firestore, allowing us to fetch, add, update, and delete data.
* Ensures seamless communication between the front end and the backend.

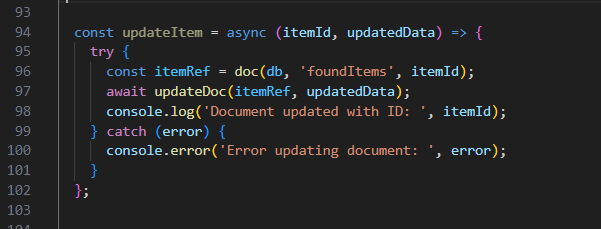
**Fetching Data with “ getDocs”**



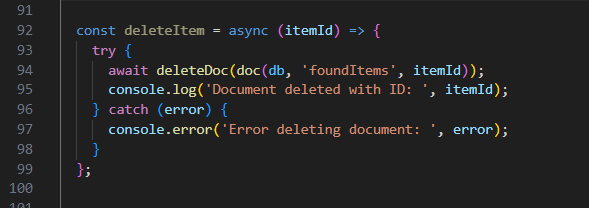
**Adding Data with “setDoc”**

****

**Updating Data with updateDoc**

****

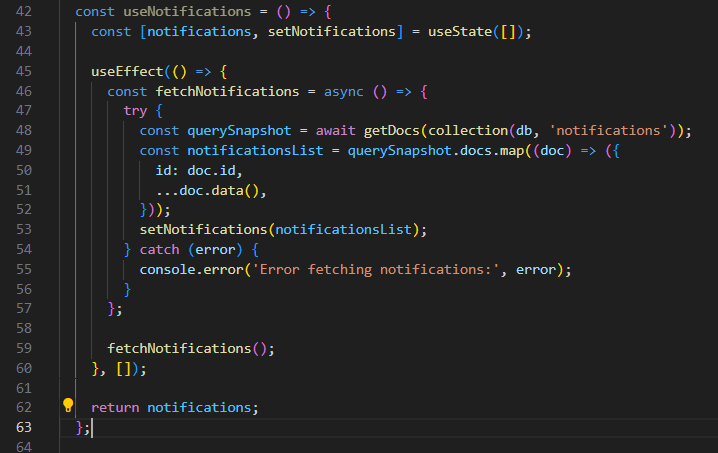
**Deleting Data with deleteDoc**

****

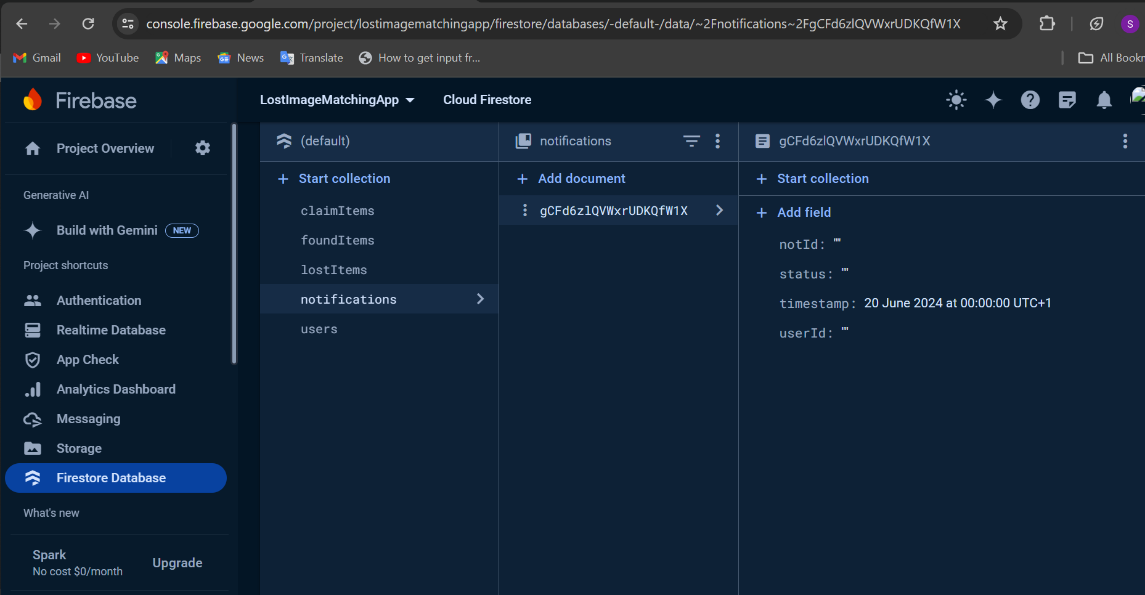
**3.5 Notifications**:

* Implemented a notification system to alert users about updates related to their items (e.g., matches found, status updates).
* Notifications are stored in the notifications collection in Firestore.

**Code to fetch notification from Database**

****

**Notification display is Database**

****

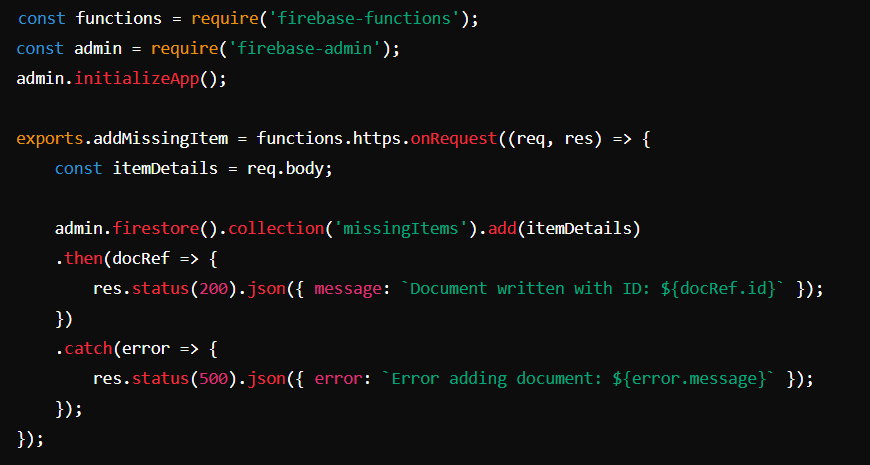
**4. Application Integration**

Integration between the mobile application and Firebase Firestore involves defining API endpoints, managing data flow, and implementing error handling to ensure smooth operation and reliability.

**4.1 API Endpoints**

Firebase Firestore, as a NoSQL database, primarily interacts with mobile applications through its SDKs rather than traditional REST API endpoints. However, Firebase Functions can be used to create custom endpoints for more advanced operations or integrations.

**Example Cloud Function Endpoint for Adding a Missing Item:**



In this example:

**Endpoint:** https://us-central1-your-project-id.cloudfunctions.net/addMissingItem

**Operation:** POST request to add a new missing item document to the Firestore collection missingItems.

**Error Handling:** Responds with appropriate status codes and error messages if the operation fails.

* **Data Flow**

Describes the flow of data between the mobile application and Firestore database. It covers how user-generated data (such as missing item reports) is transmitted to Firestore and subsequently retrieved and displayed within the application.

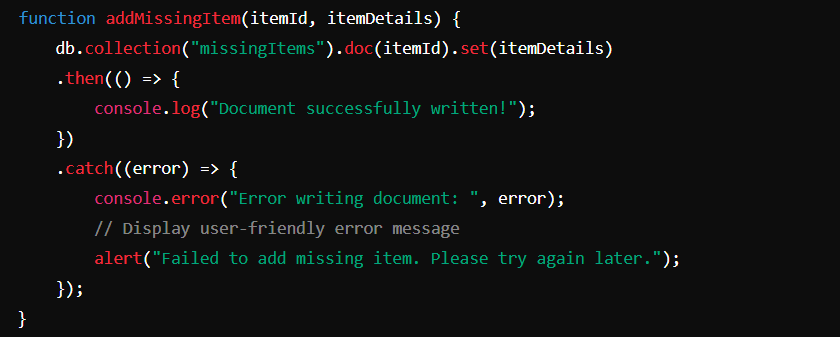
* **User Interaction:** User interacts with the mobile application to report or retrieve missing items.
* **Application Logic:** Mobile application communicates with Firebase SDKs (Firestore in this case) to perform CRUD operations or fetch data.
* **Firebase Integration:** Firebase SDK manages data synchronization between the mobile app and Firestore database.
* **Data Storage and Retrieval:** Data is stored and retrieved from Firestore collections based on user actions or scheduled tasks.
* **Real-time Updates:** Firestore provides real-time updates, ensuring that changes made in the database are reflected immediately in the application
* **Error Handling**

Explains how errors are managed and reported within the context of data operation .

And involves:

* **Firebase SDK:** Automatically manages common errors such as network issues, permission errors, and concurrency problems.
* **Custom Error Handling:** Implemented within Firebase Functions or client-side code to handle specific application logic errors.

**Example Client-side Error Handling :**

****

In this example:

* Errors encountered during document write operations are caught and logged.
* A user-friendly alert is displayed to notify the user of the issue, ensuring a smooth user experience even in the event of backend errors.

technologies used in the project, including:

* Database Management System (DBMS): Firebase.
* Frontend Framework: React Native.

# 

# **5. Conclusion**

This report has detailed the database design and implementation for our mobile application focused on archiving and retrieving missing items. Through careful analysis, design, and implementation, we have created a robust and scalable solution for archival and retrieval of missing objects.