

REPUBLIC OF CAMEROON REPUBLIC DU CAMEROUN

Peace-Work-Fatherland Paix-Travail-Patrie

University of Buea Universite de Buea

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CEF440: Internet Programming and Mobile Programming**

Database Design of Mobile-Based Archival and Retrieval of Missing Objects Application using Image Matching

**Academic year 2023/2024**

SUPERVISED BY

**Dr Nkemeni Valery**

**PRESENTED BY MEMBERS OF GROUP 9**

|  |  |
| --- | --- |
| **MATRICLE** | **NAME** |
| FE21A128 | AHOUMO TEMATEU ROXANE PHILIPPINE |
| FE21A152 | BOUCHUKE BABILA DANIEL |
| FE21A166 | DERRICK MBUNBO FORCHA |
| FE21A325 | TIANI PEKINS EBIKA |
| FE21A303 | SAMUEL OSOH |

**ACKNOWLEDMENTS**

**A**cknowledging the assistances of DR NKEMENI VALERY the course instructor and an elderly for his marvelous work and knowledge mounted upon me through this years of studies, my team and some close friends for their guidance and care. May the good LORD continues to guide and protect you all .Thanks

TABLE OF CONTENT

[INTRODUCTION **3**](#_Toc21078)

[1. OBJECTIVES **4**](#_Toc21079)

[Overview: 5](#_Toc21081)

[report structure 5](#_Toc21082)

[2. CONCEPTUAL DESIGN **6**](#_Toc21083)

1. [entity relational diagram explanation 6](#_Toc21084)
2. [visual representation 6](#_Toc21085)

[3. IMPLEMENTATION **9**](#_Toc21088)

1. [logical design with supabase 9](#_Toc21089)

B. design's strategy [9](#_Toc21090)

[C. database storage 11](#_Toc21091)

[4. CONCLUSION **10**](#_Toc21092)

Database Design and Implementation of Mobile-Based Archival and Retrieval of Missing Objects Application using Image Matching

# **INTRODUCTION**

In an increasingly digital and interconnected world, the challenge of locating and managing missing objects ranging from personal belongings to institutional assets, has become more complex and pressing. Traditional methods of retrieval, often reliant on manual searches and limited databases, are time-consuming and prone to errors. To address these limitations, we propose a novel system that leverages advanced image matching algorithms for the efficient retrieval and archival of missing objects.

This system integrates cutting-edge image recognition technology with a comprehensive and scalable database to provide a robust solution for identifying, tracking, and archiving lost items. By converting visual information into actionable data, our approach transforms how missing objects are handled, ensuring quicker recovery and systematic archival.

# 1. OBJECTIVE

The aim of designing a database for an image matching algorithm is to create an efficient, scalable, and secure system that can store, manage, and retrieve data related to image matching processes. This includes handling large volumes of image data, metadata, matching results, and associated information in a way that supports the effective functioning of the image matching algorithm and provides valuable insights for stakeholder.

In this report, we will follow the **database design process** to come out with a full **database design and Implementation**

## Structure of the work

Throughout this document, we will follow up the database design process which is as follow:

* Conceptual design
* Entity relationship diagram explanation
* Visual representation
* Implementation
* Logical design and supabase
* Design’s strategy
* Database storage

# 2. CONCEPTUAL DESIGN

Conceptual design is the process of defining the high-level structure and key elements of a system, focusing on what the system should do rather than how it will be implemented. The Database conceptual design is a phase in the database design process that focuses on creating a high-level data model that defines the structure and organization of the data in the database.

# Entity Relational Diagram (ER Diagram)

An Entity-Relationship Diagram (ERD) is a visual representation of the entities, attributes, and relationships within a database system. It is used during the conceptual design phase to illustrate how data is structured and how different data elements interact with each other. ERDs help database designers, developers, and stakeholders understand the data requirements and relationships in a clear and concise manner.

The ERD is made up of entities, it attributes and the relationship between the various entities plus their constrains. We will break up everything under.

* Entity and attributes

An entity represents a table in the database. It represents a particular object and the different type of information or data that that object will hold and how it will hold it.

Our database will be made up of 5 entities which are

An Attribute is a property or characteristic of an entity that provides additional information about the entity. Attributes describe the entity and hold the data associated with it. They hold specific pieces of data related to an entity. Below are our entities

* Relationships

The relationship defines how an entity is related to the other

* Keys (primary and secondary keys)

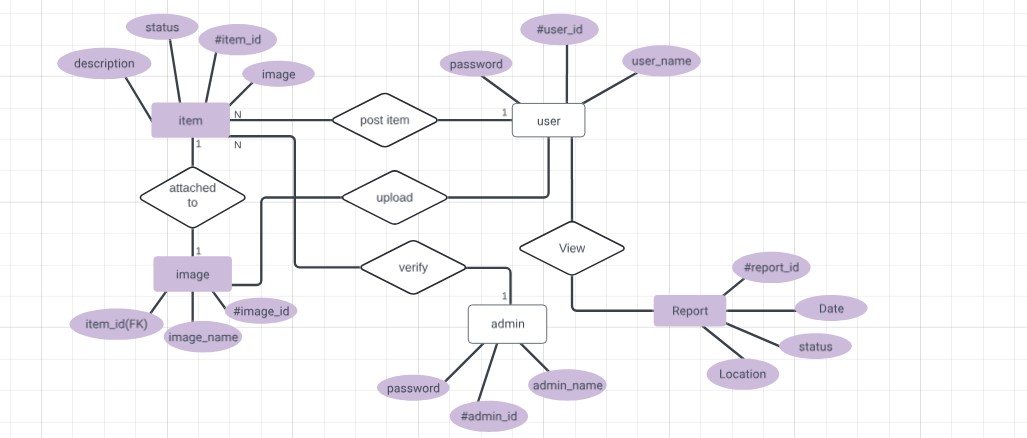
A key in the database is similar to an identifier. It is an attribute in an entity table among other attributes which is use to uniquely identify the entity. There are two types of key, which are

1. Primary key: A primary key in the context of databases is a unique identifier for each record (row) in a table. All the primary keys will be summarised on the table below.
2. Foreign key: A foreign key is a relational database concept that establishes and enforces a link between data in two tables. It is a field or a combination of fields in one table that refers to the primary key in another table.

* SUMMARY TABLE OF THE ENTITY AND CARDINALITY

|  |  |  |
| --- | --- | --- |
| ENTITY | ATTRIBUTES | CARDINALITIES and RELATION |
| User | - user ID (PK)  - username - email - password  - user type (admin or customer) | **-User** has one to many relationship with **Item** (a user can post multiple items ie either missing or found)  -**User** has one to many relationship with **Report** (a user can view multiple reports ie either missing or found) |
| Admin | - admin ID (PK) - admin name  -password | **-admin** has one to many relationship with **item** (an admin can have verify multiple items) |
| Item | -item ID(PK) -image -description  -status(lost or found) | - **Item** has a one to one relationship with **image** (only one image can be attached per status of the item) |
| Report | -report ID (FK)  -location -date -status | // |
| Image | -image ID (PK)  -item ID (FK)  -image name | - **Image** has a one to one relationship with **User** (a user can only upload one image per status of the item) |

# Visual representation



## BENRFITS OF ENTITY RELATIONAL DIAGRAM

* Provides a clear and organized visual representation of the data structure.
* Facilitates effective communication among stakeholders, designers, and developers.
* Breaks down complex database structures into manageable components.
* Helps in the normalization process by clearly showing how entities are related.
* Reduces development time and costs by providing a clear blueprint.

# 3.IMPLEMENTATION

The logical design in database design refers to the conceptual blueprint of how data elements (entities, attributes, and relationships) are organized and structured in a database management system (DBMS). It bridges the gap between the conceptual design and the physical implementation of the database.

This part deals with converting the ER diagram into relational schema in the database management system. Here, the choice of the DBMS is imposed, and the tool we have used in our project is **SUPABASE**.

## LOGICAL DESIGN WITH SUPABASE

Supabase is an open-source platform designed to streamline the development of modern web and mobile applications. It provides developers with a robust set of tools and services centered around database management, authentication, serverless functions, and real-time data synchronization.

Supabaseis backend as a service that provide the following function that makes it suitable for our project:

* For the retrieving system, supabase provide the folowing
* **Storage**:

Supabase provides ample storage options for storing large volumes of images securely. It supports various data types, including binary large objects (BLOBs), which are ideal for storing images.

* **Metadata Management**:

Alongside images, metadata such as timestamps, descriptions, and associated tags can be stored in structured tables within Supabase. This metadata facilitates efficient indexing and searching.

* **Search and Retrieval**:

Image matching algorithms, such as feature-based matching (using keypoints and descriptors) or deep learning-based methods (like convolutional neural networks), can be integrated with Supabase for similarity search. Supabase's support for custom functions and extensions allows for seamless integration of such algorithms.

## DESIGN STRATEGY

1. **Relation schema modeling**:

This is done by using the ER diagram to generate the various tables, attributes and their various relationship using supabase. Then we will use features such as BLOBs to store the various images into the database, and associate metadata such as timestamps and description.

1. **Image Processing and Matching**

Our Implementation of image matching algorithms done API calls into supabase. The algorithm can compare incoming images against archived ones to find matches based on similarity metrics.

1. **Indexing and Optimization**:

We will use PostgreSQL's indexing capabilities to optimize queries for fast retrieval based on metadata attributes or similarity scores computed by image matching algorithms.

1. **Integration with Applications**

Integrate Supabase APIs and backend services for seamless archival and retrieval operations into our react-native code in vscode.

We will also implement Real-time capabilities can enhance user experience by updating search results in real-time as new images are added or matches are found.

# DATABASE STORAGE

* USER REGISTRATION INFORMATION

The user information will be store in one of the table of our database which is the ‘**users tables’.** Here, the user’s info such as **email, phone number, user name** (provided by the user) and **userId and timestamptz** (automatically generated by Supabase). Timestamptz is an integrated service that automatically take in the date and time at which the user was created.

* UPLOADED IMAGES

The image information will be store in one table called the **‘image’.**

Each image will have an image identifier (to uniquely identify the image), a ‘created\_at’ (that keeps the exact time at which the image was uploaded) and an ‘image\_url’ (that stores just the path to the image).

* FINDER AND MISSER PERSONAL INFORMATION

The finder information are stored in the ‘finder’ table. This information is collected at the moment the finder report a found item so that we will get the current data of the user.

The finder table hold some info such as the ‘finderid, finder\_name, location and phone number’

* MATCHING ALGORITHM

Implementing an image matching algorithm typically involves using machine learning models or feature extraction techniques to compare images. These algorithms can be executed within Supabase serverless functions or external services. The serverless functions process the image and run the image matching algorithm.

Each uploaded image can be stored in a bucket, and its metadata can be stored in a PostgreSQL table.

|  |  |  |  |
| --- | --- | --- | --- |
| PART | Tools and Libraries | Use | Example |
| supabase | Database | PostgreSQL for storing image metadata and other relational data. | User registration information |
| Storage API | For storing images | It collect the image you uploaded into the database when reporting a lost or a found object. |
| Authentication | For user authentication and authorization | Validate the user info when login |
| Realtime | For real time notification and updates |  |
| Selverless functions | For executing image processing and matching algorithms | When someone report a lost object into the system, the function executes the matching algorithm |
| CLI | For managing and deploying the supabase project | When we want to deploy the database |
| Backend | Node.js | Self as a primary backend environment for developing server-side logic and API endpoints | It permits API servers to handle requests from the frontend application and to perform some tasks such as image uploads, description storage, and image matching. |
| **Express.js** | Web framework for building the API endpoints | It is use for error handling |
| **Supabase-js** | Which is an Official JavaScript library for interacting Supabase services | Interacts with Supabase storage to upload, retrieve, and manage files (e.g., images). |
| Axios | Promise-based HTTP client for making requests to external services. |  |
| Image matching Processing | OpenCV | Open-source computer vision and machine learning library for image processing. | That will be called of matching image lost with list of images in the database. |

**HOW TO CONNECT TO YOUR FRONT END APPLICATION**.

Here's how to connect

**1. Get your Supabase connection details:**

* Log in to your Supabase project dashboard.
* Navigate to the **Settings** cog icon, then select **API**.
* You'll need two pieces of information:
  + **Project URL:** This is the base URL for your Supabase project, typically in the format https://your-project-id.supabase.co.
  + **Anon API Key:** This key allows read-only access to your Supabase database. You'll likely need a different key with more permissions for writes and authentication later on.

**2. Install the Supabase client library:**

* install the appropriate Supabase client library using your package manager. here, with npm for React:

**Bash**

npm install @supabase/supabase-js

**3. Initialize the Supabase client in your frontend code:**

* Import the create **Client function** from the Supabase library.
* Pass your Supabase project URL and Anon API Key to the **createClient** function.
* Store the created client object in a variable for future use.

Here's an example using React:

**JavaScript**

import { createClient } from '@supabase/supabase-js';

const supabaseUrl = 'https://your-project-id.supabase.co';

const supabaseAnonKey = 'your\_anon\_key';

const supabase = createClient(supabaseUrl, supabaseAnonKey);

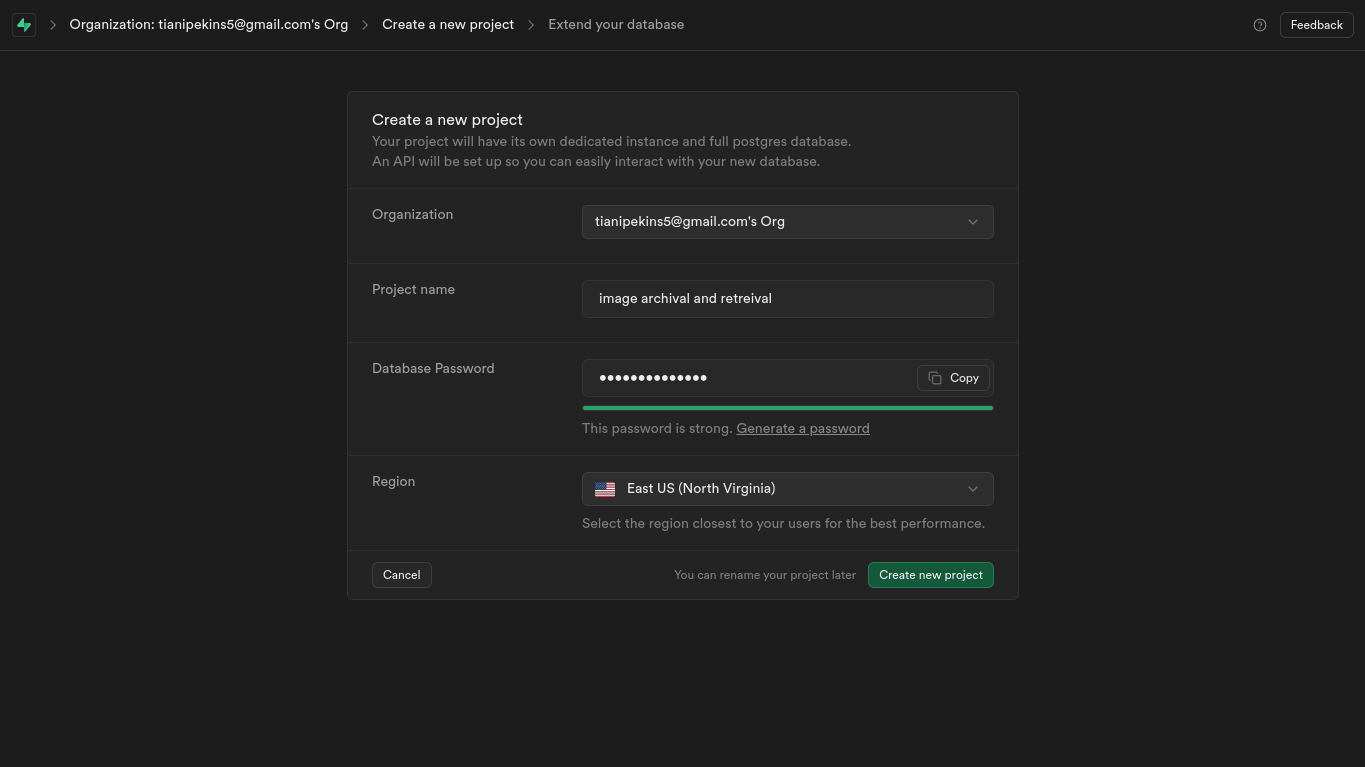
// Use the supabase object to interact with Supabase services

**4. Interact with Supabase:**

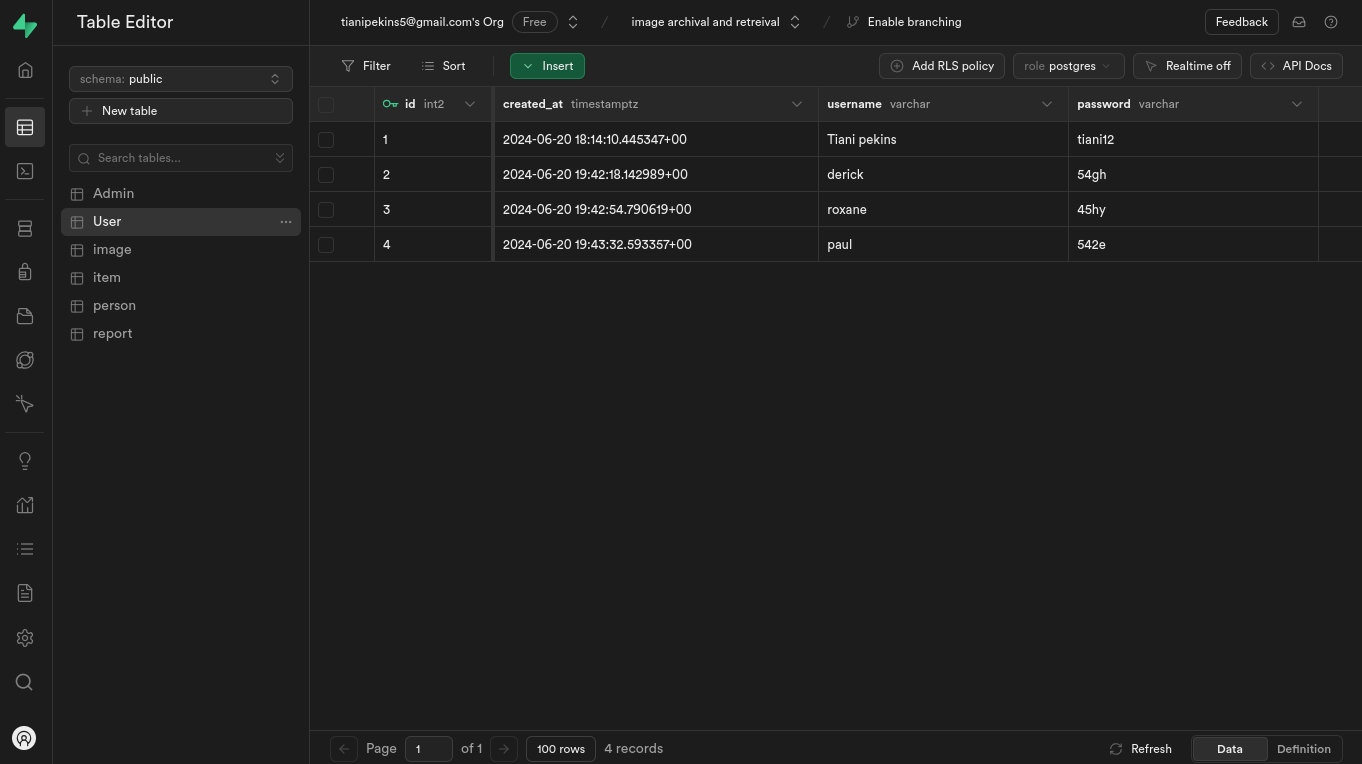
Use the Supabase client object to interact with Supabase services like the database and authentication. The Supabase client library provides methods for CRUD operations, user manageme nt, and real-time subscriptions.

# Visual representation

**CREATING AND CONNECTING THE DATABASE**



**POPULATING THE DATABASE**



# 4. CONCLUSION

This project showcases the effective use of modern database design principles and technologies to create a powerful system for image retrieval and archival. The combination of Supabase, Node.js, and Express.js provides a comprehensive solution that meets the current needs for efficient image management while offering the flexibility to adapt and grow with future technological advancements. This system not only addresses the immediate requirements of image archival and retrieval but also sets the stage for ongoing innovation and improvement in the field of image processing and database management.