**THE UNIVERSITY OF BUEA**

**REPUBLIC OF CAMEROON**

PEACE- WORK-FATHERLAND

PAIX-TRAVAIL-PATRIE

P.O Box 63,

Buea, South West Region

Cameroon



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**Department of Computer Engineering**

**CEF 440: INTERNET PROGRAMMING AND MOBILE PROGRAMMING**

**MOBILE APPLICATION FOR ARCHIVAL AND RETRIEVAL OF MISSING OBJECTS VIA IMAGE MATCHING**

PRESENTED BY: By Group 21

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

Course Instuctor:

Dr. NKEMENI VALERY

Contents

[**ABSTRACT** 4](#_Toc171223757)

[**1.** **INTRODUCTION** 5](#_Toc171223758)

[1.1. PROBLEM STATEMENT 5](#_Toc171223759)

[**2.** **LITERATURE REVIEW** 7](#_Toc171223760)

[**3.** **METHODOLOGY** 8](#_Toc171223761)

[3.1. Model of the Mobile Application Architecture used 8](#_Toc171223762)

[3.2. REQUIREMENT GATHERING 9](#_Toc171223763)

[3.2.1 IDENTIFYING STAKEHOLDERS 9](#_Toc171223764)

[3.2.2 REQUIREMENT GATHERING TECHNIQUES AND ACTIVITIES 9](#_Toc171223765)

[INTERVIEWS 9](#_Toc171223766)

[ONLINE SURVEY 10](#_Toc171223767)

[3.2.3. CONTEXTUAL ANALYSIS: 21](#_Toc171223768)

[Prioritization Methodology 23](#_Toc171223769)

[Won’t Have (Lowest Priority): 25](#_Toc171223770)

[System Requirements Specification 26](#_Toc171223771)

[Non-Functional Requirement 26](#_Toc171223772)

[Hardware Requirements 27](#_Toc171223773)

[Software Requirements 27](#_Toc171223774)

[**i.** Integration of Third-Party Services 29](#_Toc171223775)

[**ii.** Testing and Quality Assurance 30](#_Toc171223776)

[3.3 UML DIAGRAMS 30](#_Toc171223777)

[3.3.1 USE CASE DIAGRAM 31](#_Toc171223778)

[3.3.2 CLASS DIAGRAM 32](#_Toc171223779)

[3.3.3 SEQUENCE DIAGRAM 33](#_Toc171223780)

[3.3.4 CONTEXT DIAGRAM 34](#_Toc171223781)

[3.3.5 DEPLOYMEMT DIAGRAM 35](#_Toc171223782)

[UI DESIGN 35](#_Toc171223783)

[Design Principles 35](#_Toc171223784)

[Color Scheme: 36](#_Toc171223785)

[Reason for Color Choice: 36](#_Toc171223786)

[UI IMPLEMENTATION 44](#_Toc171223787)

[Tools and Technologies 44](#_Toc171223788)

[Database Design 45](#_Toc171223789)

[Entity-Relationship Diagram (ERD) 46](#_Toc171223790)

[DATABASE IMPLEMENTATION 49](#_Toc171223791)

[**CONCLUSION** 61](#_Toc171223792)

[**REFERENCES** 62](#_Toc171223793)

# **ABSTRACT**

In today's digital age, mobile technology plays a crucial role in enhancing our daily lives and addressing complex issues. One such challenge is the retrieval and archival of missing objects, a widespread problem causing significant financial and emotional distress. This report explores the development of a Mobile-Based Archival and Retrieval of Missing Objects Application using Image Matching technology. By leveraging advanced image recognition and the ubiquity of smartphones, this application aims to streamline the process of locating lost or stolen items. The system utilizes a microservice architecture, enabling users to upload images of lost objects and search through a database of found items using image matching algorithms. This innovative approach promises to improve the efficiency and success rate of recovering lost belongings, thereby reducing the inconvenience and emotional impact of such losses.

# **INTRODUCTION**

In the contemporary digital era, mobile technology significantly enhances daily life and addresses complex issues, such as the retrieval and archival of missing objects, a global problem causing substantial financial and emotional distress. Traditional methods, relying on manual searches and centralized databases, are often inefficient and time-consuming. The proposed Mobile-Based Archival and Retrieval of Missing Objects Application using Image Matching leverages advanced image recognition technology and the ubiquity of smartphones to provide a more effective solution. This application enables users to upload images of lost items and search a database of found objects using sophisticated image matching algorithms. By employing a microservice architecture, the system ensures high performance, scalability, and reliability, streamlining the recovery process and increasing the likelihood of finding lost items. This innovative approach offers a practical and efficient solution, reducing the burdens associated with lost belongings and harnessing modern technology to enhance recovery efforts.

## PROBLEM STATEMENT

Every year, countless personal items are lost or stolen, causing not only significant financial loss but also personal inconvenience and emotional distress to individuals.

Traditional methods of locating missing objects often rely on manual searches or centralized databases, leading to inefficiencies and delays in retrieval. There is a need for a more streamlined and accessible solution to aid in the swift recovery of lost items.

In the digital era, there is a critical need for a more dynamic and technologically advanced solution that can offer a higher rate of recovery with less effort and time. Existing digital solutions fail to fully exploit the potential of modern technologies like image recognition, which can significantly enhance the process of identifying and matching found items with their rightful owners.

The objective of this project is to design and implement a user-friendly mobile application that enables users to upload images of missing objects and search through a database of found items using image matching algorithms. This technology promises not only to enhance the probability of recovering lost items but also to streamline the process, making it quicker and more efficient.

PROJECT GOALS

* To create an easy-to-use mobile platform that allows users to upload images of lost or found objects.
* To use image matching technology to automate the comparison of newly uploaded images against existing entries in a robust database.
* To enable users to communicate securely and coordinate the return of objects.

This report explores the development and implementation of a Mobile-Based Archival and Retrieval of Missing Objects Application using Image Matching technology. It delves into the functionalities of the proposed application, examining how image matching algorithms can be used to efficiently locate lost items. Additionally, it investigates potential use cases across various sectors, such as transportation, hospitality, and personal belongings. Furthermore, the report assesses the effectiveness and feasibility of the proposed solution in comparison to traditional methods of object retrieval.

# **LITERATURE REVIEW**

Losing personal belongings can be a stressful and time-consuming experience. Fortunately, advancements in mobile technology and computer vision offer promising solutions for archiving and retrieving lost objects.

Extensive research have been conducted on various aspect of archival and retrieval of missing objects. Ghazal et al [1] proposed a multi-platform mobile application where users upload pictures of lost objects. The system employs a multi-feature approach combining image matching with color validation and text search to improve retrieval accuracy. Their experiments achieved a reported accuracy of 95% with a 100% precision rate.

Other studies suggest integrating additional information like location data or object descriptions with image matching for more robust retrieval [2]. This approach can potentially improve the search process by narrowing down potential matches, especially for generic objects.

Researchers have proposed many image matching algorithms at present. According to the different principles, these algorithms can be divided into five categories, that is, image matching algorithm based on the gray value [3], image matching algorithm based on the

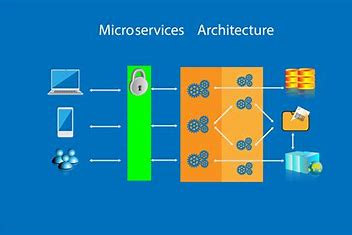
feature [4], image matching algorithm based on the frequency domain analysis [5], image matching algorithm based

on the neural network [6] and image matching algorithm based on the semantic recognition [7].

# **METHODOLOGY**

## Model of the Mobile Application Architecture used

1. Microservice Architecture



In this distributed architecture style, an app is built as an ecosystem of finer-grained, independently deployable microservices:

Key Characteristics:

 Functionalities split into autonomous services

 Services can be independently developed, tested, deployed

 Individual scaling capability per service

 Communication via API

Pros: Highly scalable. Enables continuous delivery.

Cons: Complex to create, orchestrate, test and monitor

Microservices shine for internet-scale apps needing extreme performance, availability and scaling needs. Examples include Netflix, Amazon, and Uber.

## REQUIREMENT GATHERING

For a successful design and implementation of a Mobile-Based Archival and Retrieval of Missing Objects Application using Image Matching, the requirement gathering process is important. This process will guide the development of an application that efficiently serves the needs of users wanting to find their missing objects using image matching technology. This phase is critical as it directly influences the scope, design, functionality, and overall success of the application.

Here is a detailed breakdown of the requirement gathering process for this project:

### 3.2.1 IDENTIFYING STAKEHOLDERS

This step involves listing all parties who might have an interest in or be affected by the mobile app.

* Users: Individuals who will use the app to report missing or found objects.
* Community Organizations: Local groups interested in property safety or community services.
* Legal and Compliance Experts: To address data privacy issues, especially related to uploading personal data and images.
* Developers and Technical Staff: Those who will develop and maintain the app.

### 3.2.2 REQUIREMENT GATHERING TECHNIQUES AND ACTIVITIES

INTERVIEWS:

We used a qualitative research method to gather in-depth information from potential users, stakeholders, experts, and other relevant parties. We conducted interviews with a series of questions. Here is a detailed look at how the interview questions were structured.

* How often do you misplace or lose important items?
* Which types of items do you commonly misplace or lose?
* Have you used any similar apps or solutions for finding lost items before? If yes,
* please specify.
* What features would you expect from an app designed to help you find missing
* objects using image matching?
* How comfortable are you with using technology to track and locate your
* belongings?
* What devices do you primarily use for such tasks (e.g., smartphone, tablet,
* computer)?
* Would you be interested in receiving notifications or alerts when an item is
* detected using image matching?
* How important is accuracy in locating missing items for you?
* Are you concerned about the privacy and security of your data while using such an

app?

* Would you prefer a free app with ads or a paid app without ads for this purpose?
* Do you have any suggestions or specific features you would like to see in this app?
* Personal Documents like National Identification Cards, Passports, Driver’s license, Credit Cards, Certificates.

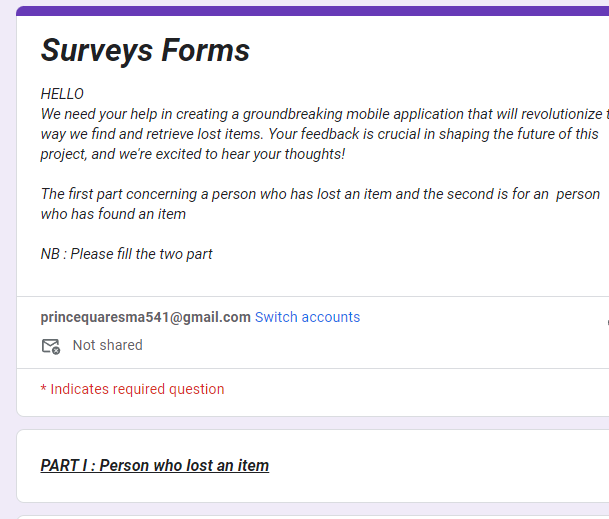
After performing this interview. We found out that the following objects are commomly lost in our community:

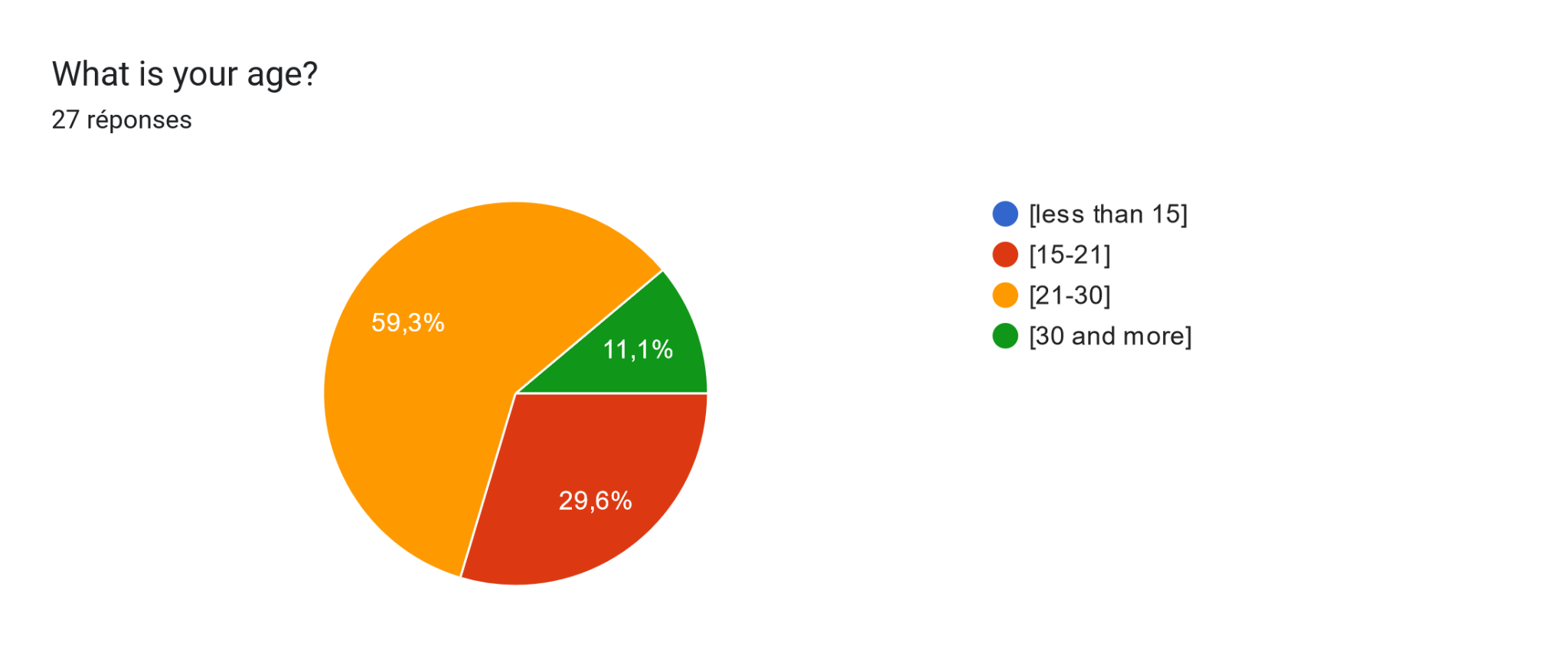
* Personal electronics like phones, laptops and tablets.
* School Items like textbooks and school books.
* Wallets and Purses.
* House and car keys.
* Jewelry and Watches.

The loss of these items often prompts the need for systems that can help in their quick retrieval to minimize inconvenience or the potential for theft.

ONLINE SURVEY:

We carried out an online survey using google forms to collect data from a large group of people about their needs, preferences, behaviors, and perceptions. This method is particularly valuable for gathering quantitative insights from a wide audience quickly and efficiently. We conducted the online survey with a series of question:





so the majority are the persons between [21-30]



Tableau des réponses au formulaire Forms. Titre de la question : How often do you misplace or lose important items?
. Nombre de réponses : 27 réponses.

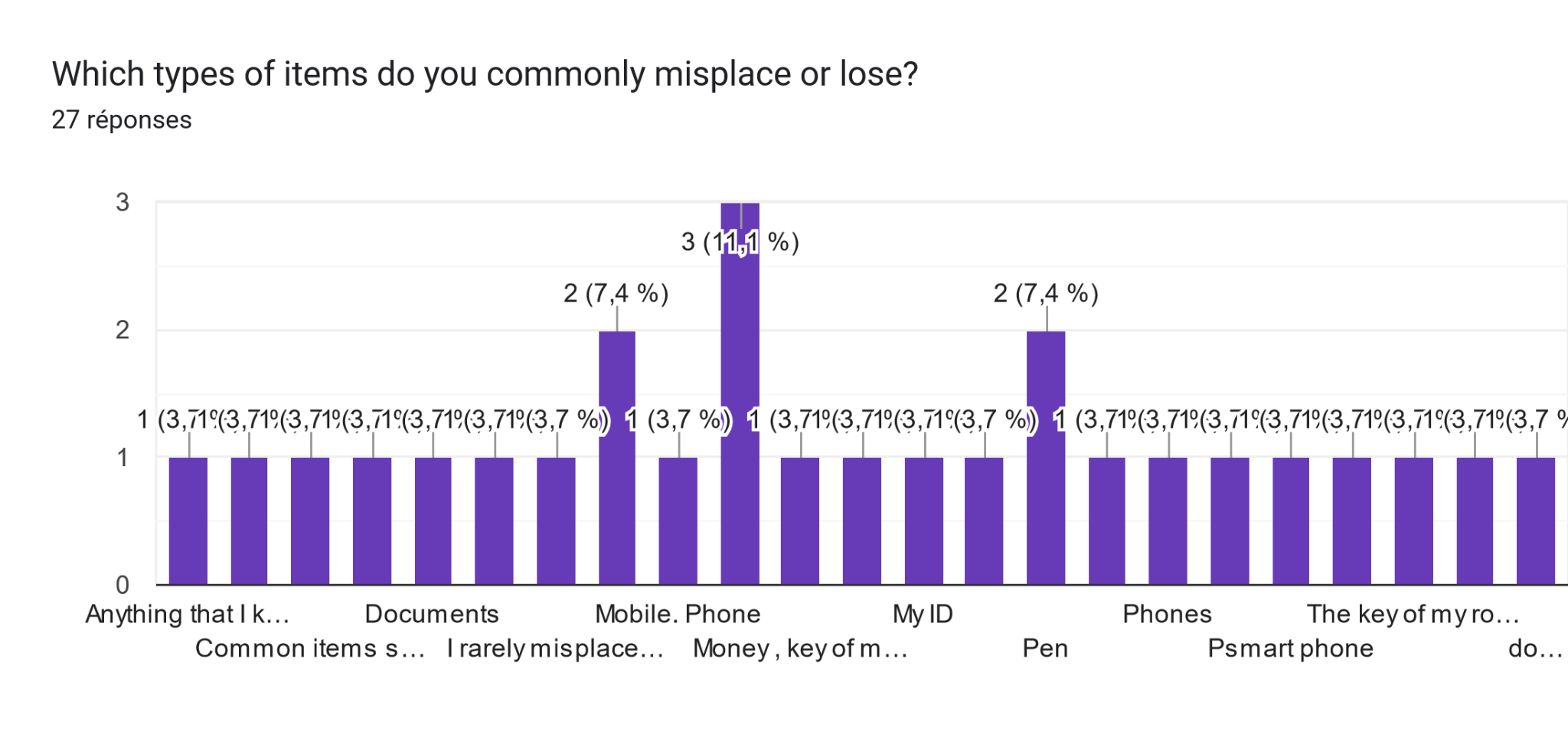


Tableau des réponses au formulaire Forms. Titre de la question : Which method have you  use to find this items ?
. Nombre de réponses : 27 réponses.

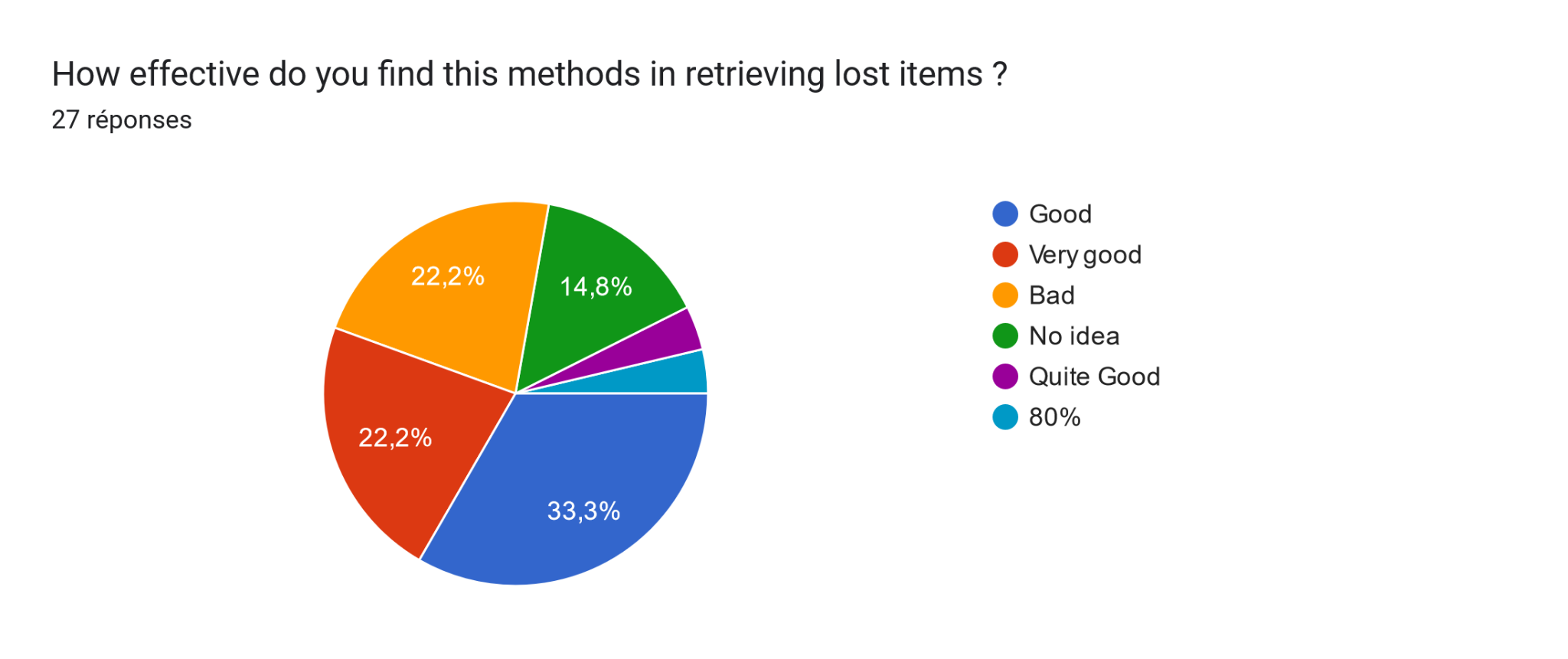
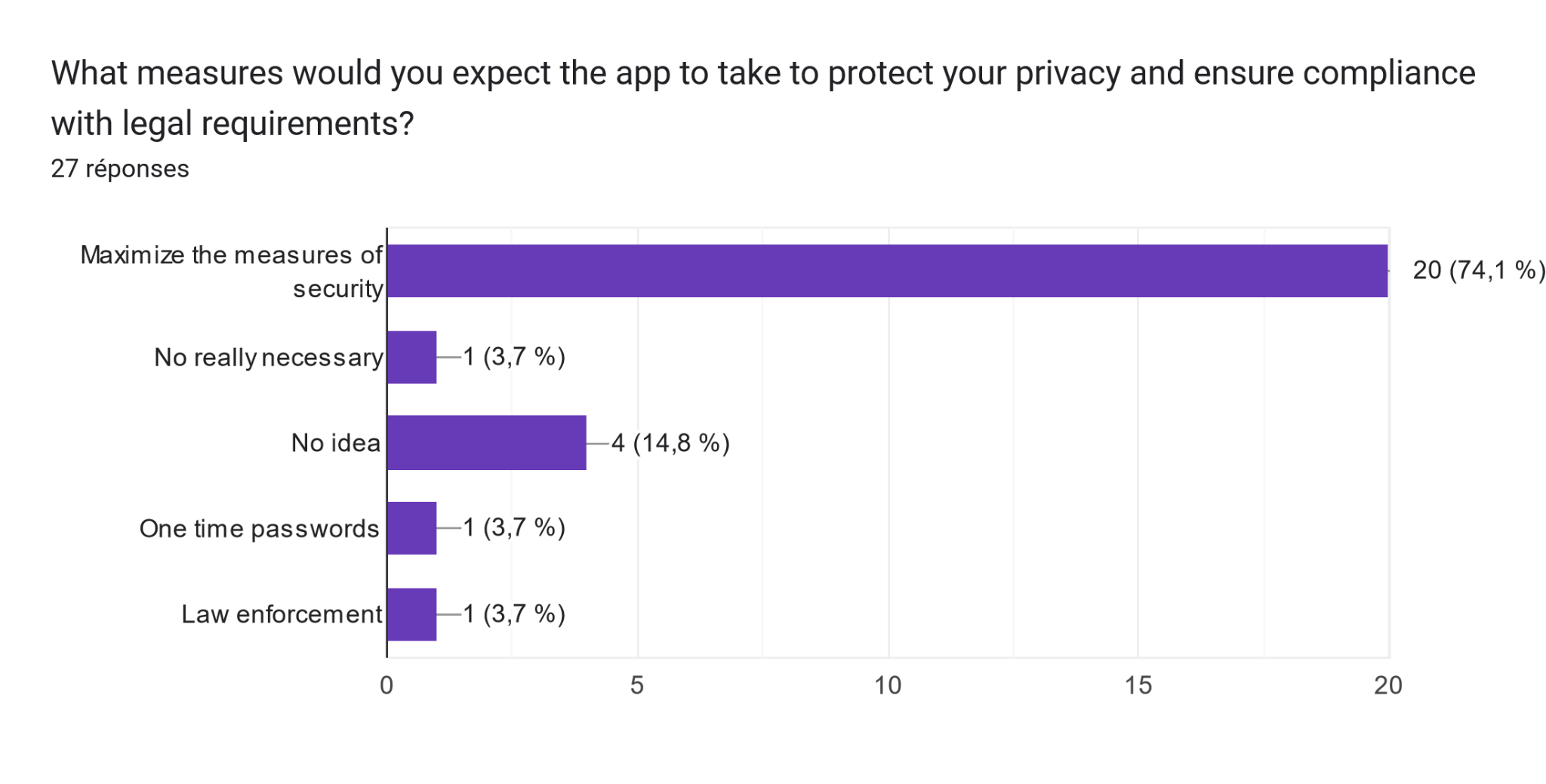


Tableau des réponses au formulaire Forms. Titre de la question : Do you want to have a mobile App to found your items ?
. Nombre de réponses : 24 réponses.

Tableau des réponses au formulaire Forms. Titre de la question : Are you  concerned about the privacy and security of your data while using a such App ?
. Nombre de réponses : 27 réponses.



* Concerning the person who found an items

Tableau des réponses au formulaire Forms. Titre de la question : Have you ever found a lost item and successfully returned it to its owner?
. Nombre de réponses : 27 réponses.

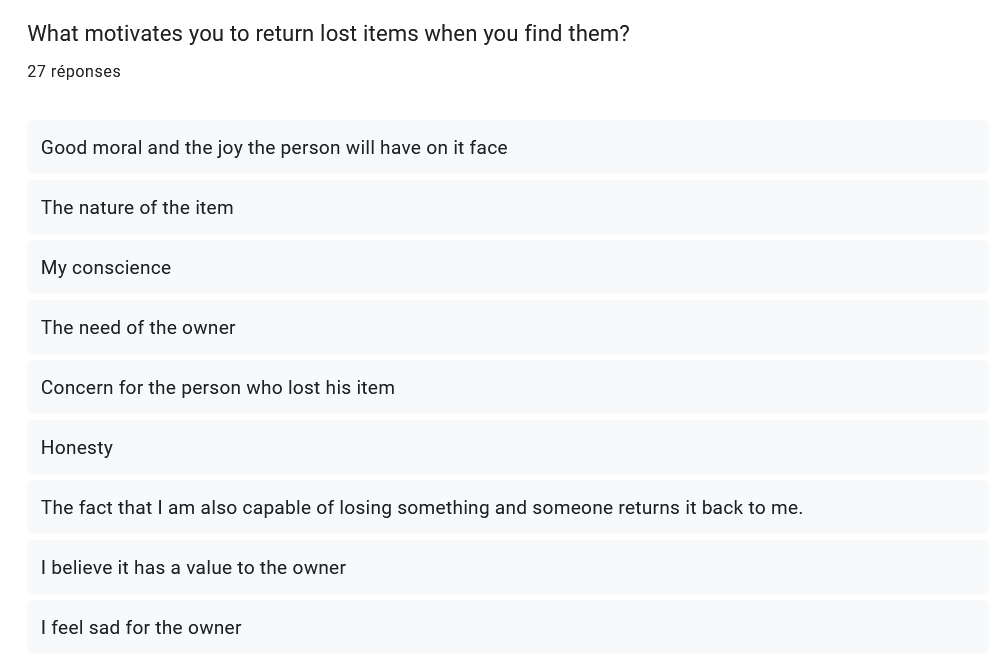


Tableau des réponses au formulaire Forms. Titre de la question : How often do you come across lost items in your daily life?
. Nombre de réponses : 27 réponses.

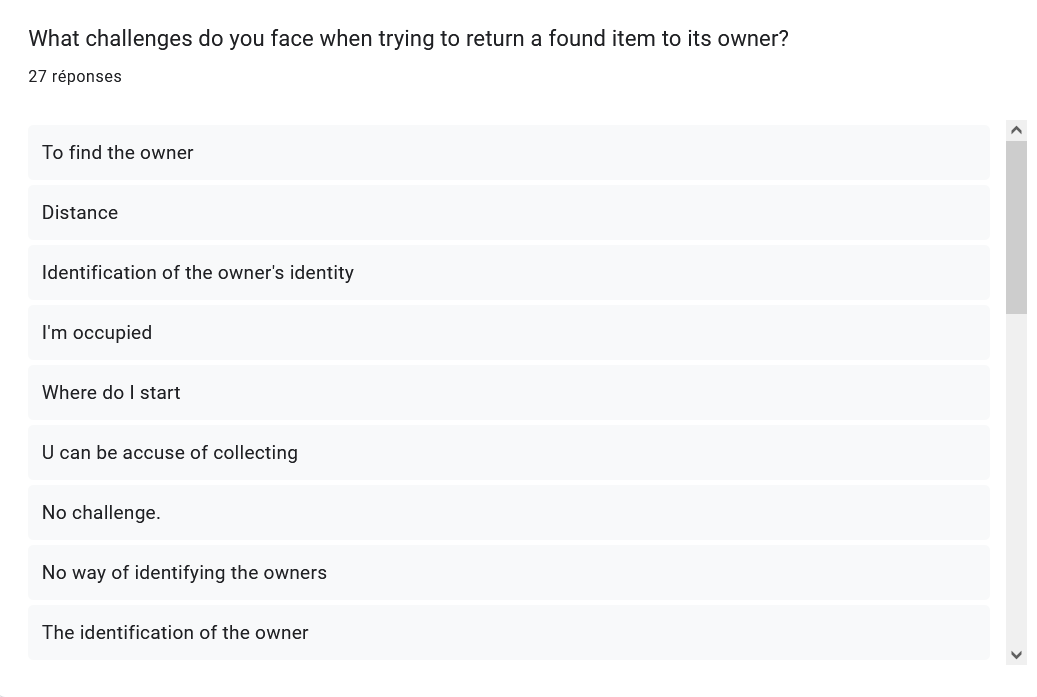
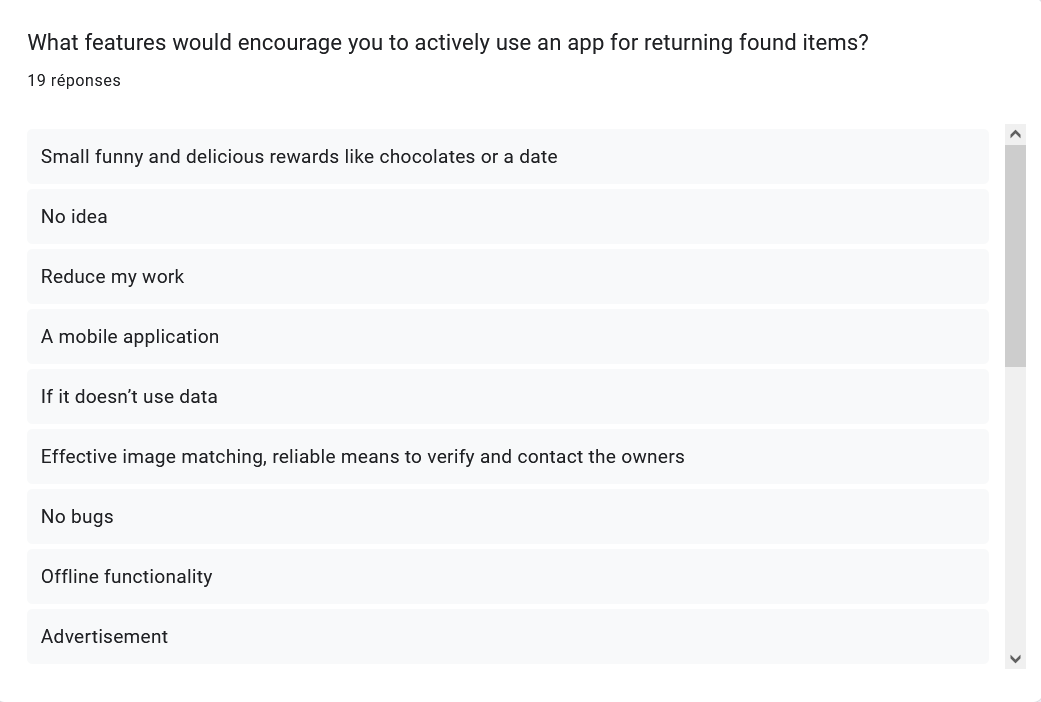


Tableau des réponses au formulaire Forms. Titre de la question : Would you be willing to use a mobile app to help identify the owner of a lost item through image matching?
. Nombre de réponses : 26 réponses.



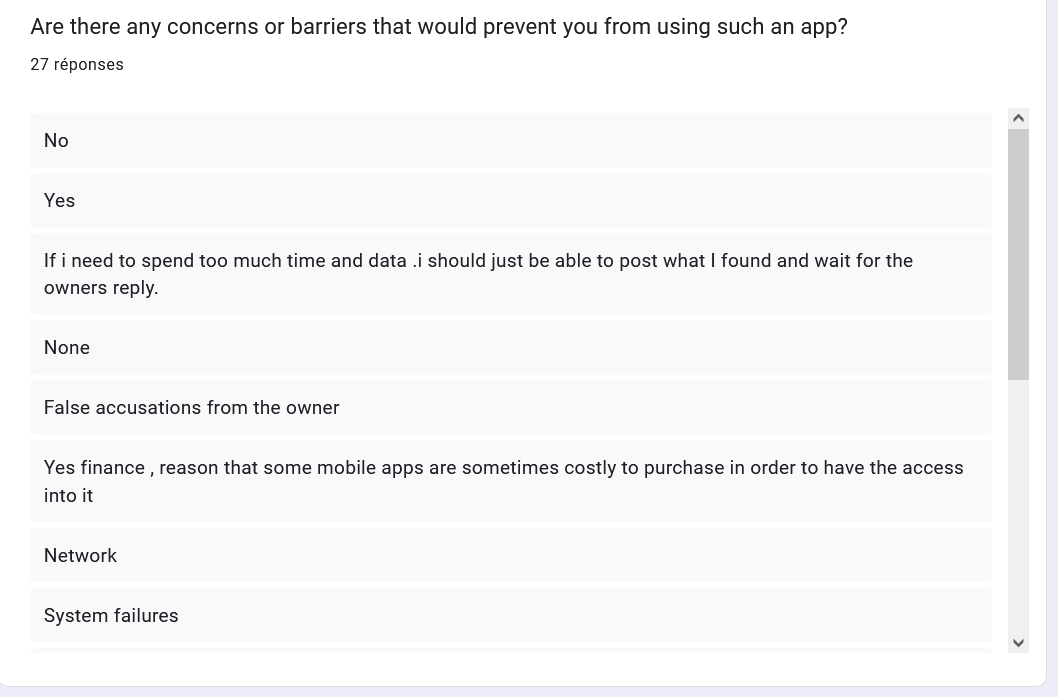
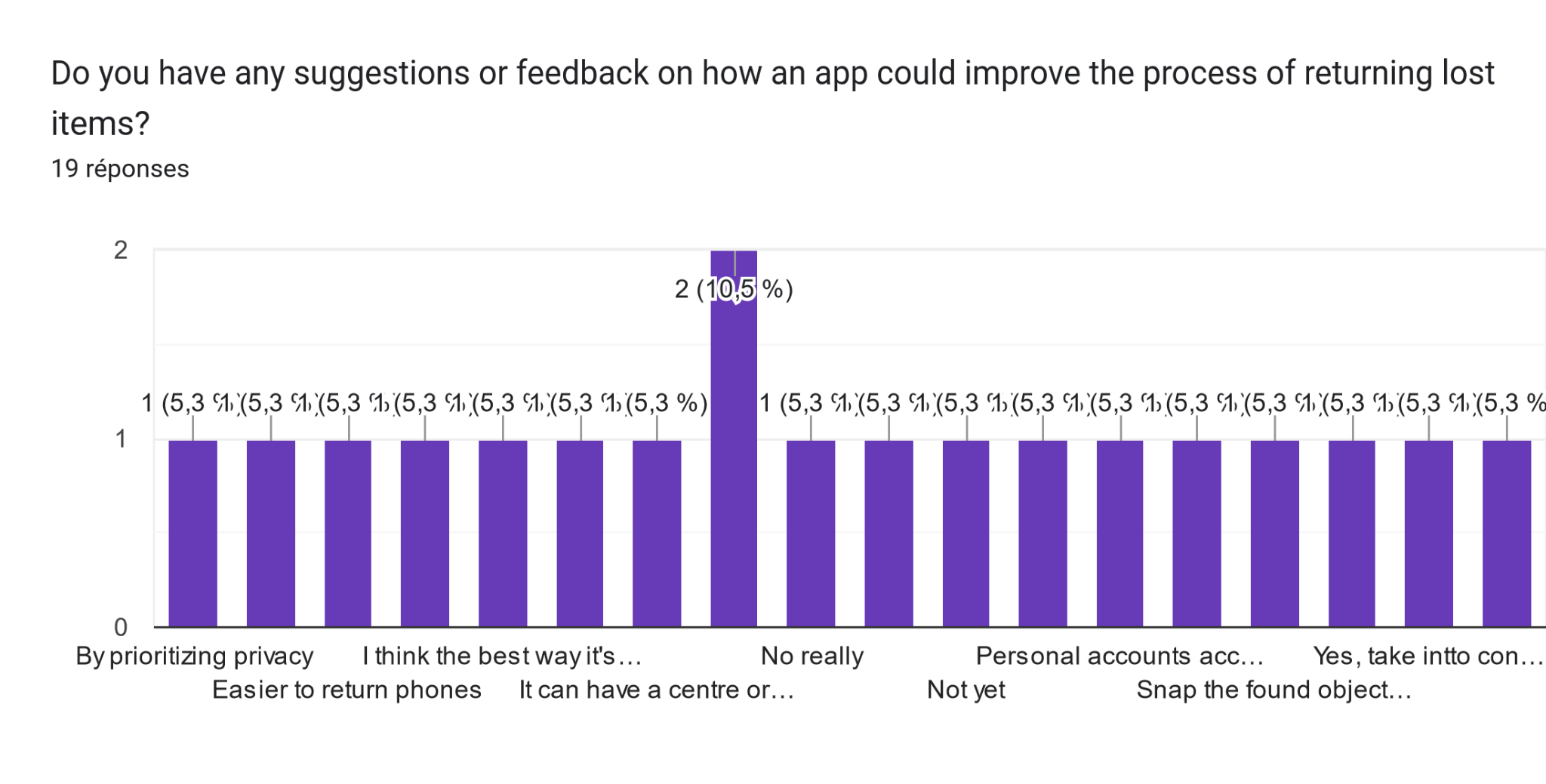
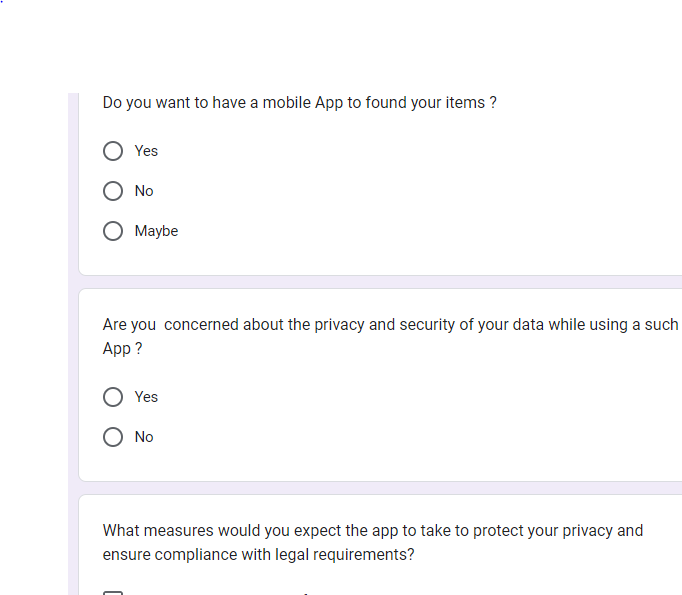


Tableau des réponses au formulaire Forms. Titre de la question : How important is it for you to have a reliable and secure method to connect with the owner of a lost item?
. Nombre de réponses : 26 réponses.

Tableau des réponses au formulaire Forms. Titre de la question : Would you prefer to remain anonymous when returning a found item, or are you open to communication with the owner?
. Nombre de réponses : 26 réponses.





### 3.2.3. CONTEXTUAL ANALYSIS:

The contextual inquiry was conducted in several places across Buea. The inquiry consisted of direct observation of individuals in public spaces, such as markets, transportation hubs, and commercial districts, as well as semi-structured interviews with participants who had experienced losing or finding objects.

Observations:

* Initial Reactions to Losing Objects:

Upon realizing they had lost an object, individuals displayed visible signs of distress, including expressions of frustration and anxiety.

* Immediate search efforts were observed, with individuals scanning their surroundings and retracing their steps to locate the missing item.

Some individuals sought assistance from nearby individuals or authorities, while others relied on personal search efforts.

* Reactions to Finding Objects:

Individuals who found lost objects initially displayed surprise or curiosity upon discovery.

Decision-making processes regarding the found objects varied, influenced by factors such as the perceived value of the item and the likelihood of finding the owner.

Many individuals expressed a desire to return found items to their rightful owners, motivated by a sense of moral obligation and community spirit.

Interview Findings:

* Emotional Impact of Losing Objects:

Participants described feelings of frustration, anxiety, and helplessness upon losing objects, particularly if the items held sentimental or practical value.

The loss of essential items such as identification documents or money was particularly distressing and often led to immediate search efforts.

* Motivations for Returning Found Objects:

Participants cited cultural norms emphasizing honesty and communal responsibility as primary motivations for returning found objects.

Some individuals recounted personal experiences of losing items and expressed a desire to be compensated when they find an item

DOCUMENT ANALYSIS

Conducting a document analysis for requirement gathering in the development of a mobile-based application for archival and retrieval of missing objects using image matching involves systematically reviewing existing documentation and sources relevant to our project. This method helps to understand current capabilities, identify gaps, and define the scope of new system requirements based on learned information.

SHAREHOLDERS NEEDS AND EXPECTATIONS

* User-Friendly Interface: Users expect a simple, intuitive interface that makes uploading images, searching for items, and navigating the app straightforward.
* Accuracy of Image Matching: High accuracy in matching found objects with lost items to ensure users can reliably recover their belongings.
* Speed: Quick responses and fast loading times, especially when uploading images and receiving match results.
* Privacy and Security: Robust protection of personal data, including images and contact information. Users want assurance that their information is secure and privacy is respected.
* Accessibility: The app should be accessible to users of all abilities, including provisions for those with visual or motor impairments.
* Notifications and Updates: Timely notifications about potential matches and updates on their lost or found items.
* Support and Help: Easy access to customer support in case of issues or questions regarding the app’s use.

SYSTEM LIMITATIONS

* Manual Matching: Users must manually browse and compare lost and found listings, which can be time-consuming and inefficient.
* Low Recovery Rates: The effectiveness of these apps often depends on the user base size and engagement, leading to potentially low recovery rates.
* Limited Search Capabilities: Search functions are often basic, relying on text descriptions without utilizing more advanced filtering or matching technologies.

Requirement Prioritization

Requirement prioritization is a critical process in the development of any software project. It involves evaluating and ranking each requirement based on its importance to the project's objectives, stakeholders' needs, and the overall value it adds to the product. This report focuses on the prioritization of requirements for the proposed mobile application designed to help users find missing objects using image matching technology.

Prioritization Methodology:

For this project, we will use the MoSCoW method of prioritization, which categorizes requirements into four groups:

* Must have (M): Essential for the project's success and form the minimum usable subset.
* Should have (S): Important but not necessary for launch; can be included in future releases.
* Could have (C): Desirable but not necessary and could improve user experience or customer satisfaction if included.
* Won’t have (W): Least critical, lowest payback items, or not appropriate at this time.

Must Have (Critical for Launch):

1. **User Authentication**:

* Secure login/logout functionality.
* Password recovery and account verification mechanisms.

1. **Image Upload and Database Management:**

* Basic image upload functionality with metadata (description, location, timestamp).
* A centralized database infrastructure for storing and retrieving images efficiently.

1. **Core Image Matching Functionality:**

* Robust image recognition and matching algorithms.
* Accurate and fast search results for matching missing objects.

1. **Real-Time Notifications:**

* Immediate alerts for users when potential matches are found.
* Push notification service integration.

1. **Data Privacy Compliance:**

* Adherence to GDPR and other relevant data protection regulations.
* Secure handling and storage of user data.

Should Have (Important but Not Critical for Initial Launch):

1. **Profile Management:**

* Ability for users to manage their profiles and posted items.
* Options for users to edit and delete their posts.

1. **Advanced Search Filters:**

* Additional search parameters such as date, location, and category.
* Enhanced sorting and filtering capabilities for search results.

1. **User Feedback Mechanism:**

* Features for users to report success stories or provide app feedback.
* Rating system for community engagement and trust-building.

1. **Social Media Integration:**

* Sharing of posts to social platforms to increase reach.

Social login options for ease of access

Could Have (Desirable Features):

1. **User Community Features:**

* Forums or chat functions for user interaction.
* Gamification elements to encourage community participation.

1. **Multilingual Support:**

* Translation features to cater to non-English speaking users.
* Localization of content based on the user's geographic location.

Won’t Have (Lowest Priority):

1. **Augmented Reality (AR) Integration:**

* AR features for visualizing lost items in the environment.
* Due to budget and time constraints, this feature will not be included in the initial release.

1. **Offline Access:**

* Ability to use certain features of the app without an internet connection.
* Considered a luxury feature that can be explored in future updates.

The prioritization of requirements ensures that the development team can focus on delivering the most valuable features first, aligning with the project's strategic goals and constraints. This prioritized list will guide the development process and help stakeholders understand the project's focus areas for the initial launch and future updates.

### System Requirements Specification

The System Requirements Specification (SRS) document outlines the system requirements for the mobile application designed to archive and retrieve missing objects using image matching technology. This includes functional nonfunctional and both software and hardware requirements needed to support the functionality of the application.

Purpose:

The purpose of this SRS is to provide a detailed overview of the functional and non-functional requirements that will guide the development and ensure the application meets the needs of its users and stakeholders.

Scope:

The application will allow users to upload images of missing objects, search for objects using image matching, and receive notifications for potential matches. It will support both iOS and Android platforms.

Non-Functional Requirement

Non-functional requirements specify how the system performs certain tasks and under what constraints. And then include:

Performance:

* The system shall return search results within an acceptable time frame.
* The system shall handle a high number of concurrent users without degradation of performance.

Usability:

* The system shall offer an intuitive and user-friendly interface.
* The system shall provide help documentation for user support.

Reliability:

* The system shall have an uptime of 99%.
* The system shall provide accurate image matching results.

Security:

* The system shall implement industry-standard encryption for data transmission.
* The system shall have measures to prevent unauthorized access and data breaches.

Scalability:

* The system shall be designed to easily accommodate an increasing number of users.
* The system shall allow for future expansion of features and functionalities.

Hardware Requirements:

Mobile Devices:

* The system shall be compatible with iOS devices running iOS 13.0 or later.
* The system shall be compatible with Android devices running Android 8.0 (Oreo) or later.

Server:

* The system shall be hosted on a server with a minimum of 8 GB RAM and quad-core CPU.
* The system shall ensure data redundancy and backup solutions are in place.

Software Requirements:

1. Operating Systems:

* The system shall support the latest two major versions of iOS and Android operating systems.

Development Frameworks:

* The system shall be developed using cross-platform frameworks such as React Native or Flutter for mobile development.
* The system shall use a reliable database management system like PostgreSQL or MongoDB.

1. APIs and Libraries:

* The system shall integrate with push notification services like Firebase Cloud Messaging (FCM) or Apple Push Notification service (APNs).
* The system shall utilize image processing libraries or APIs that support image matching functionalities.

The system requirements outlined in this SRS are intended to provide a comprehensive understanding of the application's specifications. These requirements will drive the development process and ensure that the end product aligns with user expectations and project objectives.

Technical requirement:

Technical requirements for a mobile-based archival and retrieval system for missing objects using images encompass the specific technologies, frameworks, and components needed to develop and deploy the system effectively. Here's a detailed explanation of key technical requirements:

Mobile App Development:

* Cross-Platform Compatibility: Develop the mobile app for multiple platforms, such as iOS and Android, to reach a wider audience. Consider using frameworks like React Native or Flutter to build cross-platform apps with a single codebase.
* User Interface Design: Design intuitive and user-friendly interfaces that facilitate easy reporting of missing objects, searching for matches, and receiving notifications. Implement responsive design principles to ensure compatibility with various screen sizes and resolutions.
* Image Capture and Processing: Integrate functionalities for capturing images using the device's camera and processing them to enhance quality and reduce noise. Implement features like cropping, rotating, and adjusting image brightness and contrast.
* Offline Support: Implement offline support to allow users to report missing objects and search for matches even when they're not connected to the internet. Use local storage to cache data and synchronize it with the server when connectivity is restored.

Backend Development:

* Server Infrastructure: Set up a robust server infrastructure to handle image processing, storage, and matching algorithms. Use cloud services like AWS, Google Cloud, or Azure for scalability and reliability.
* API Development: Develop RESTful APIs to enable communication between the mobile app and the server. Define endpoints for user authentication, object reporting, search queries, and notifications.
* Image Matching Algorithm: Implement image recognition and matching algorithms to compare reported missing objects with found objects. Consider using computer vision libraries like OpenCV for feature extraction and matching.
* Database Management: Use a relational or NoSQL database to store user data, reported objects, matching results, and other relevant information. Ensure data integrity, security, and scalability by optimizing database design and indexing.

1. Integration of Third-Party Services**:**

* Authentication Services: Integrate authentication services like Firebase Authentication or OAuth to enable secure user authentication and authorization.
* Cloud Storage: Use cloud storage services like Amazon S3 or Google Cloud Storage to store uploaded images securely and cost-effectively.
* Push Notifications: Integrate push notification services like Firebase Cloud Messaging (FCM) or Apple Push Notification Service (APNs) to send timely notifications to users about potential matches for their reported missing objects.

Security and Privacy:

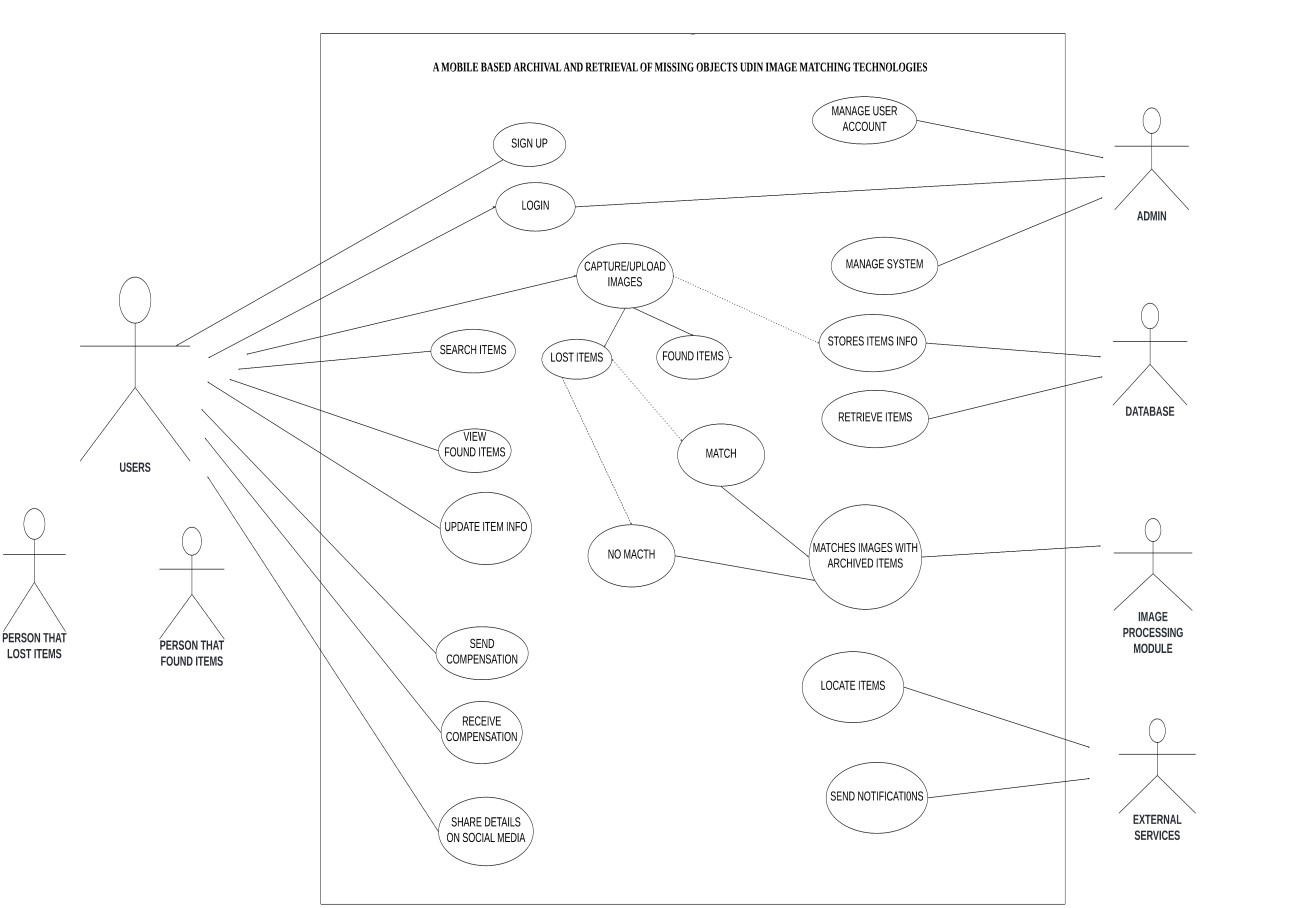
* Data Encryption: Encrypt sensitive data, including user information and uploaded images, both in transit and at rest, using industry-standard encryption algorithms.
* Secure Communication: Use HTTPS protocol for communication between the mobile app and the server to ensure data integrity and prevent eavesdropping.
* Access Control: Implement access control mechanisms to restrict unauthorized access to sensitive features and data. Use role-based access control (RBAC) or permissions-based access control to define user roles and privileges.

1. Testing and Quality Assurance**:**

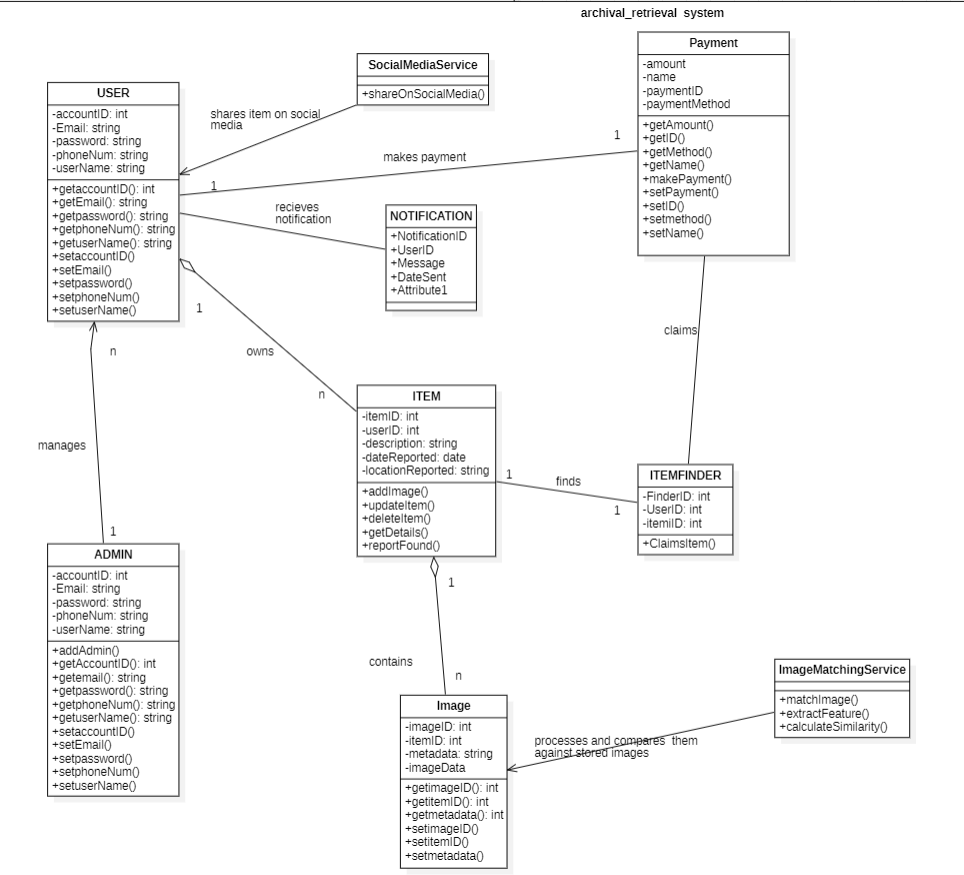
* Unit Testing: Write unit tests to verify the functionality of individual components and ensure code quality and reliability.
* Integration Testing: Conduct integration testing to validate the interactions between different modules and components of the system.
* User Acceptance Testing: Involve real users in user acceptance testing to gather feedback on the usability, performance, and functionality of the system.

# 3.3 UML DIAGRAMS

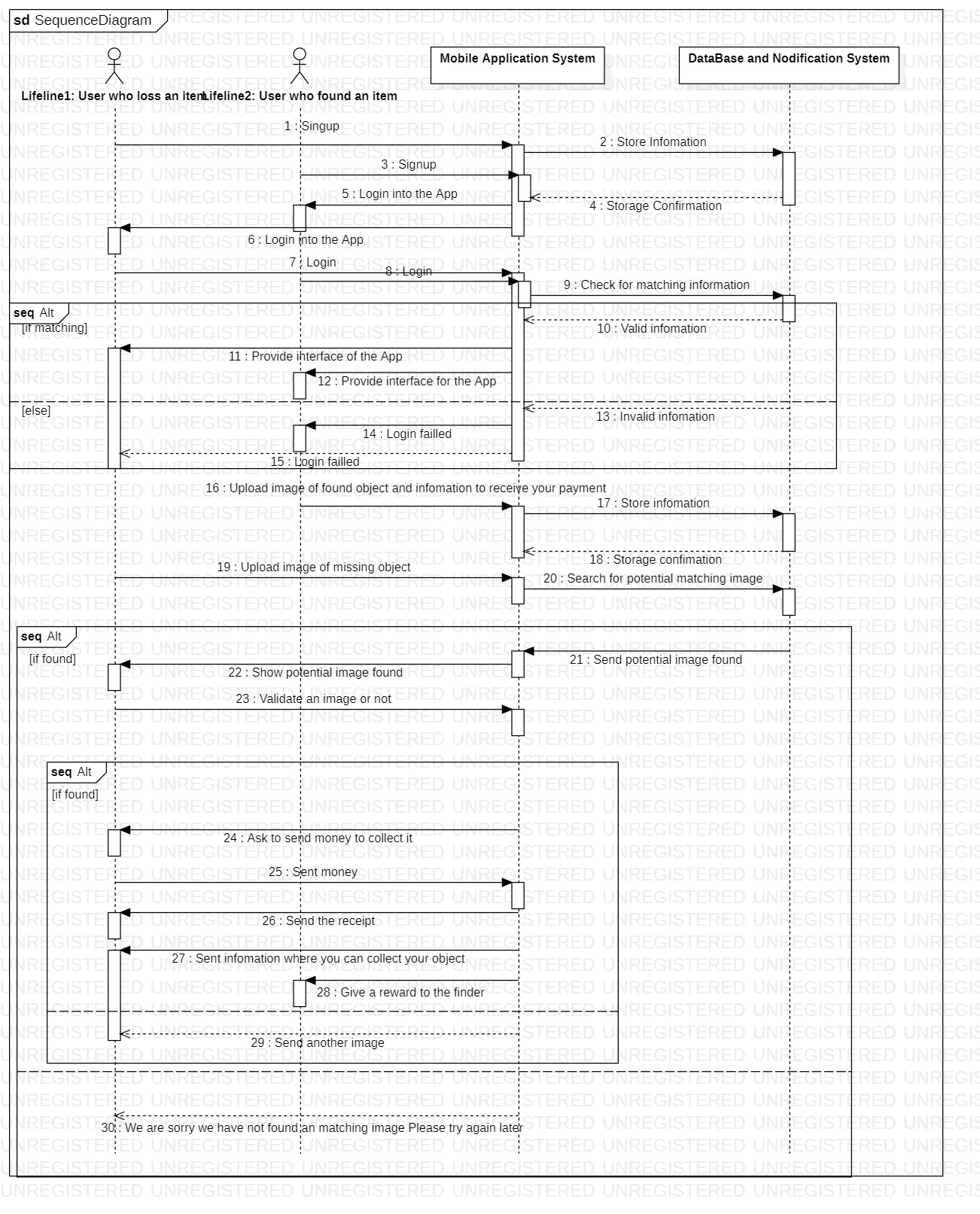
## 3.3.1 USE CASE DIAGRAM



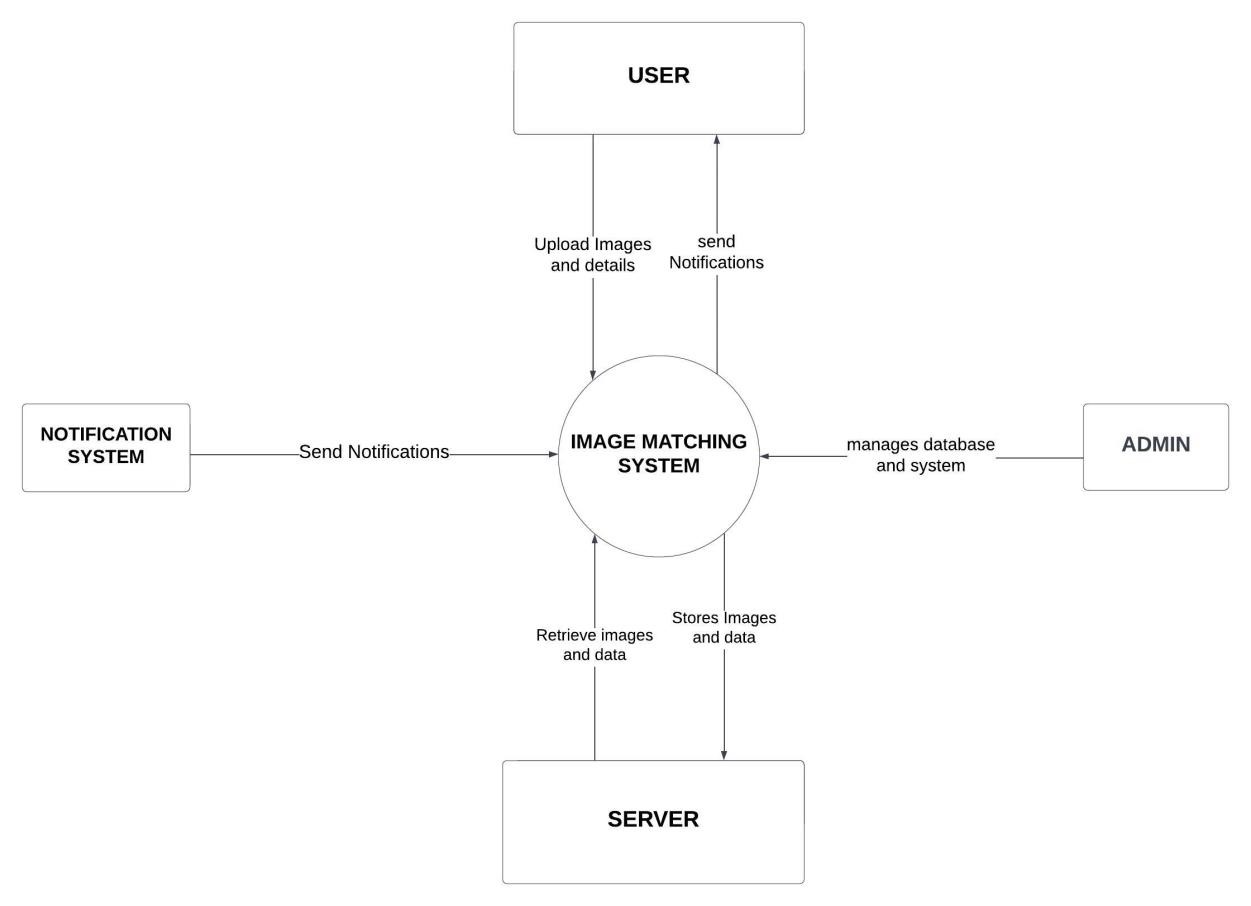
## 3.3.2 CLASS DIAGRAM



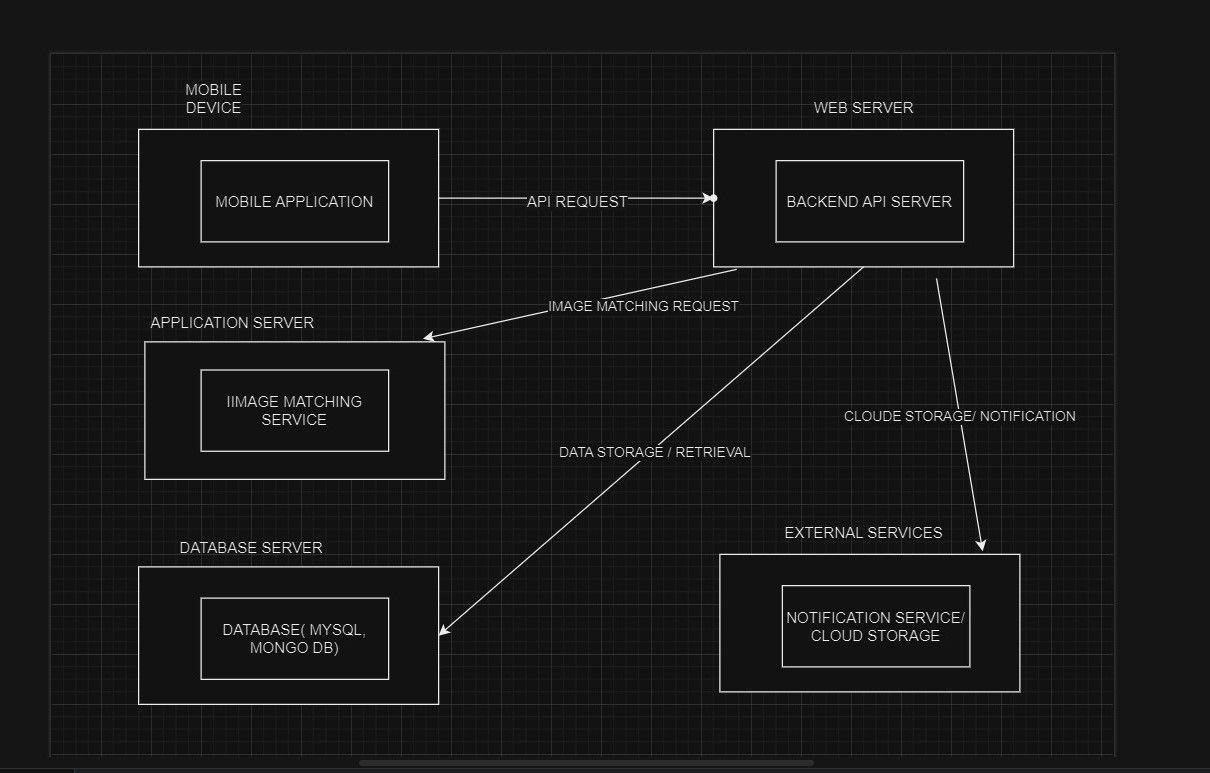
## 3.3.3 SEQUENCE DIAGRAM



## 3.3.4 CONTEXT DIAGRAM



## 3.3.5 DEPLOYMEMT DIAGRAM



# UI DESIGN

The UI design of this "Lost to Found" app emphasizes simplicity, intuitive navigation, and visual appeal. It features a clean and minimalist layout with easy access to key functionalities such as searching for lost items, reporting found items, and browsing through a gallery of found items. The design prioritizes user-friendliness and efficiency, with a focus on guiding users through the process of locating lost items using image matching technology. Overall, the UI design aims to provide a seamless and engaging experience for users searching for and reporting lost items.

Understanding the target audience is crucial for designing an effective UI. Extensive user research was conducted to gather insights into user preferences, behaviors, and pain points related to lost item retrieval. Surveys and interviews were conducted to identify common challenges faced by users when trying to locate missing items.

Design Principles

* Simplicity: The UI of this app is designed to simple using likable and universal colors like blue and white, which complicates the application less for the users
* Unity: Crating a sense of harmony between all the elements in a page. This creates a sense of unity when all the elements are conceptually arranged together. This ensures that the concepts being communicated are clear and cohesive. We did however introduce some variety to create a balance between a dynamic and chaotic design
* Hierarchy: There is a perceived order of the various elements included in each page which is aimed at highlighting the most important elements
* Rhythm: All the elements of this UI design were consistent in shape, size and color which intuitively guides the user.
* Balance: Coherent rules applied to fonts, color, size, gives a more beautiful and balanced interface.

Color Scheme:

Primary color: Blue

Secondary colors: white and Sky Blue

The "Lost 2 Found" app utilizes a color scheme comprising blue, white and sky blue. This choice is aimed at enhancing user experience and ensuring the app's effectiveness in helping users archive and retrieve missing objects.

### Reason for Color Choice:

* Trust and Reliability:

Blue is universally associated with trust, reliability, and security. It instills confidence in users, making them feel their information and activities on the app are safe and secure.

* Calmness and Clarity:

Sky Blue evokes feelings of calmness and serenity. It helps reduce user stress, which is particularly important when dealing with the anxiety of lost items.

* Visual Hierarchy and Contrast:

The combination of blue and sky blue provides a subtle yet effective contrast, ensuring important elements stand out without overwhelming the user. This visual hierarchy helps users navigate the app efficiently.

* Consistency and Professionalism:

Maintaining a consistent color palette throughout the app enhances the overall aesthetic and user experience. It presents a professional and polished interface that aligns with the app's purpose.

* Focus and Readability:

Blue tones are easy on the eyes, reducing strain during prolonged use. This is crucial for an app where users might spend considerable time searching for and reporting items.

2.3. Typography:

Typography Selection: Clear Fonts with Minimal Text

The typography used in this app is designed to enhance readability, user engagement, and overall user experience. The chosen fonts are clear and straightforward, complemented by the strategic use of minimal text and a catchy slogan: “You lossam, we findam.”

Rationale for Typography Choice:

* Clarity and Readability
* Minimal Text for Maximum Impact
* Engaging Slogan:

The slang slogan, “You lossam, we findam.” is prominently featured. It is concise, memorable, and adds a touch of personality to the app. This informal phrase resonates with users, making the app feel approachable and user-friendly.

* Consistency:

Consistent use of typography throughout the app ensures a cohesive look and feel. This consistency helps users become familiar with the interface quickly, improving overall usability.

* Aesthetic Appeal:

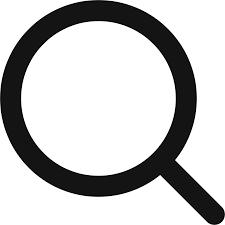
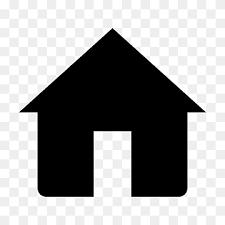
The clean fonts and minimalistic text contribute to a modern, professional aesthetic. This enhances the visual appeal of the app and aligns with the clean design ethics.

Icons and imagery

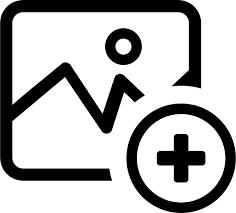
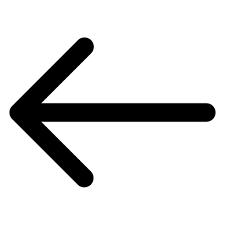
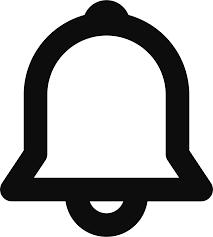
The icons and imagery used are designed to be simple and easily recognizable. This ensures quick identifiable functions and easy to navigate the app without confusion. Dis engaging icons and images also contribute to a more interactive and enjoyable user experience.

In the "Lost to Found" app, various icons are used to enhance usability and provide clear visual cues for different functions. Here are some of the key icons used:

**ICONS**



Home screen Search User profile



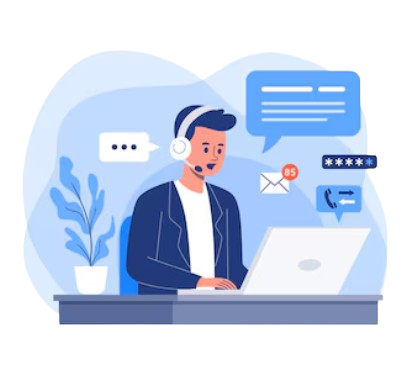
Notifications Input image (lost/ found) Back icon

**IMAGES**



App Logo Visualizes return of lost item Contact us





Search of missing items Represent support and customer service

PAGE DESCRIPTION:

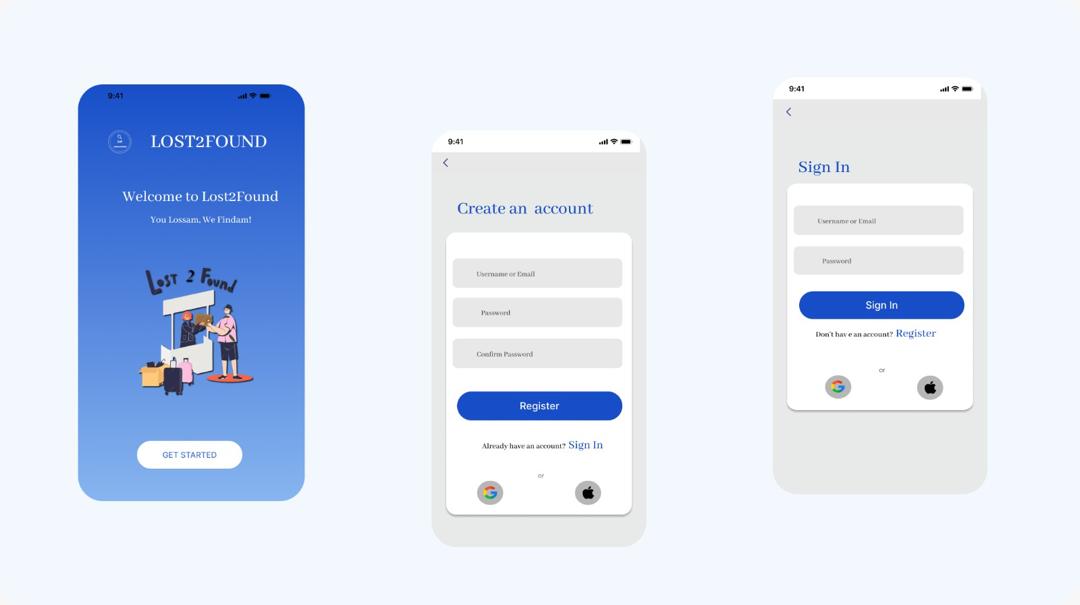
Providing a detailed description of all pages of the application. They are as follows:

* Welcome page

The first page on the application including the app’s logo, welcome message and the apps slogan “you lossam, we findam!” which is meant to capture the attention of the user and retain them..

* Sign in /sign up page

Including simple and clear instructions to help the user create an account or gain access to their already existing account. With minimal but clear words for simplicity



* Home page

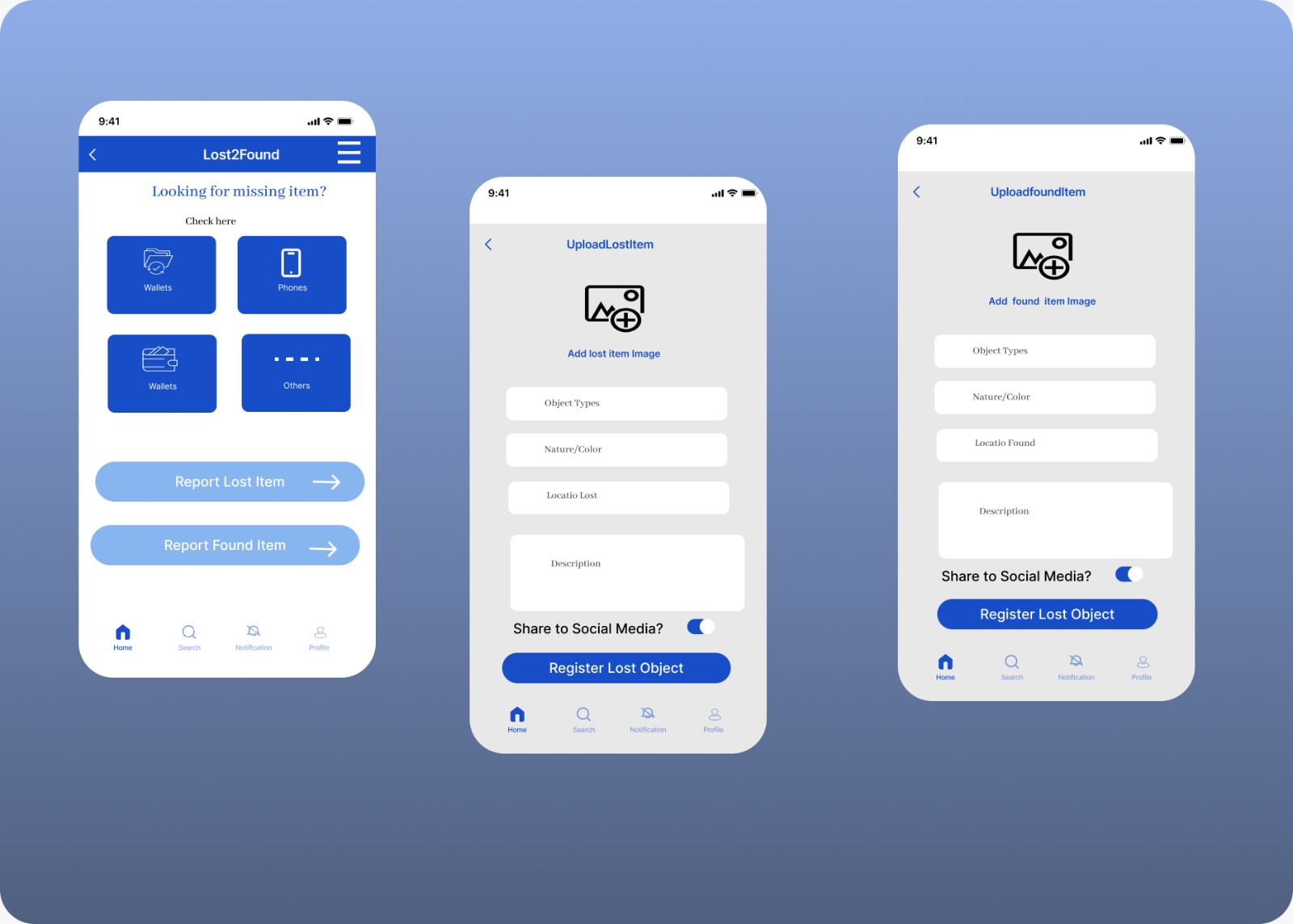
Evokes the user’s eyes to lost item images already recorded in the application, engaging them and also call to action buttons of either recording a lost or found item with icons on the bottom of the page for a seamless navigation.

* Upload lost object page

This page includes clear instructions on how to register a missing object (*add lost item image)* and fields for object description including *object type,* *nature/type, location lost.* This page also offers options like *share to social media?*

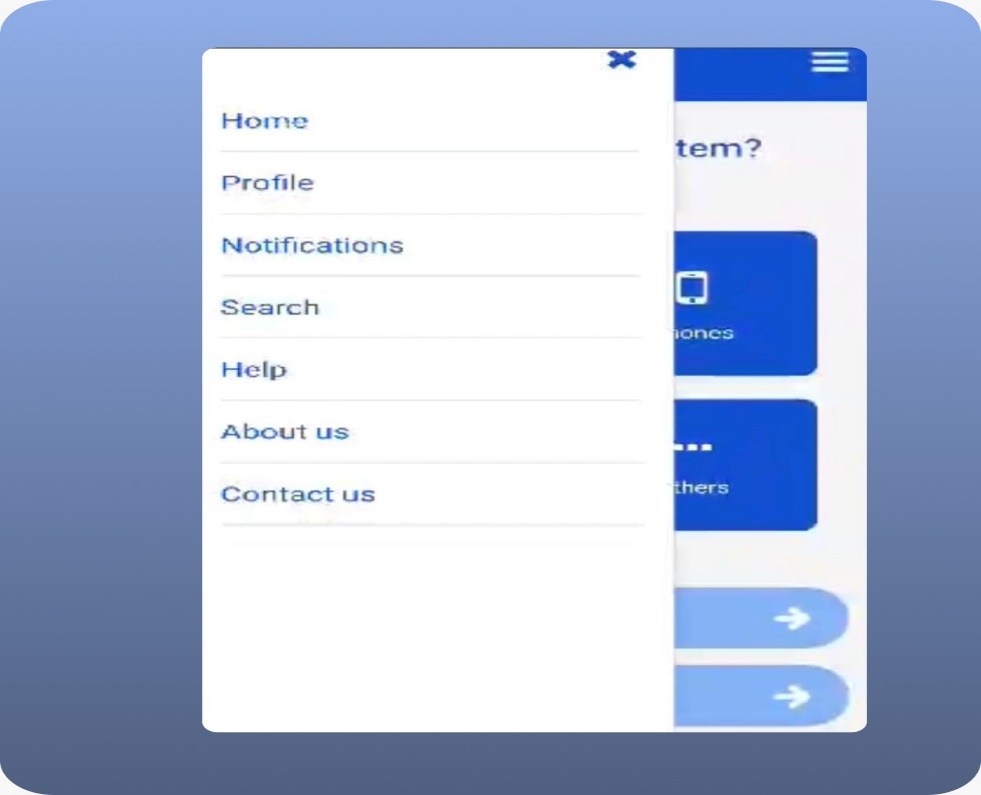
* Upload found object page

This page includes clear instructions on how to register a found object (*add found item image)* and fields for object description including *object type,* *nature/type, location found.* This page also offers options like *share to social media*



* Menu bar

With the menu button found on the top right corner of the home page for easy visibility without disruptions to the app includes buttons leading to the home, profile, notification, search, help, about us and contact us pages. With minimal words and colors for easy understanding.



* Profile page

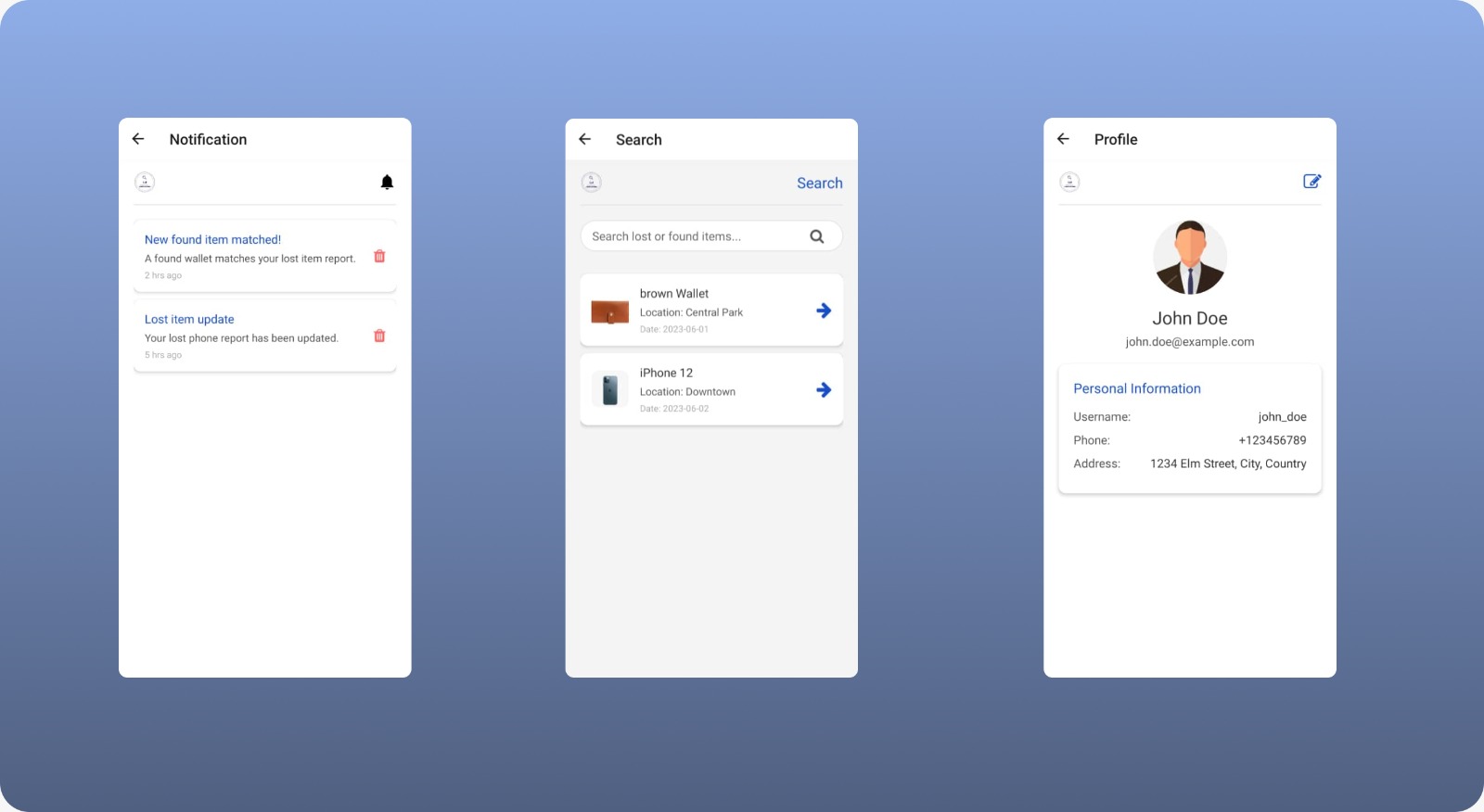
This page user’s personal and public information including a profile picture or nil, email address, username, phone number and address with minimal colors

* Search page

With clear instructions on the search bar *search for lost or found item ,* this page is meant for the search and display of items already registered on the database.

* Notification page

This page where all message sent from the application to the user are found which will include the image and description of a potential found item which might bgelong to said user



* Help page

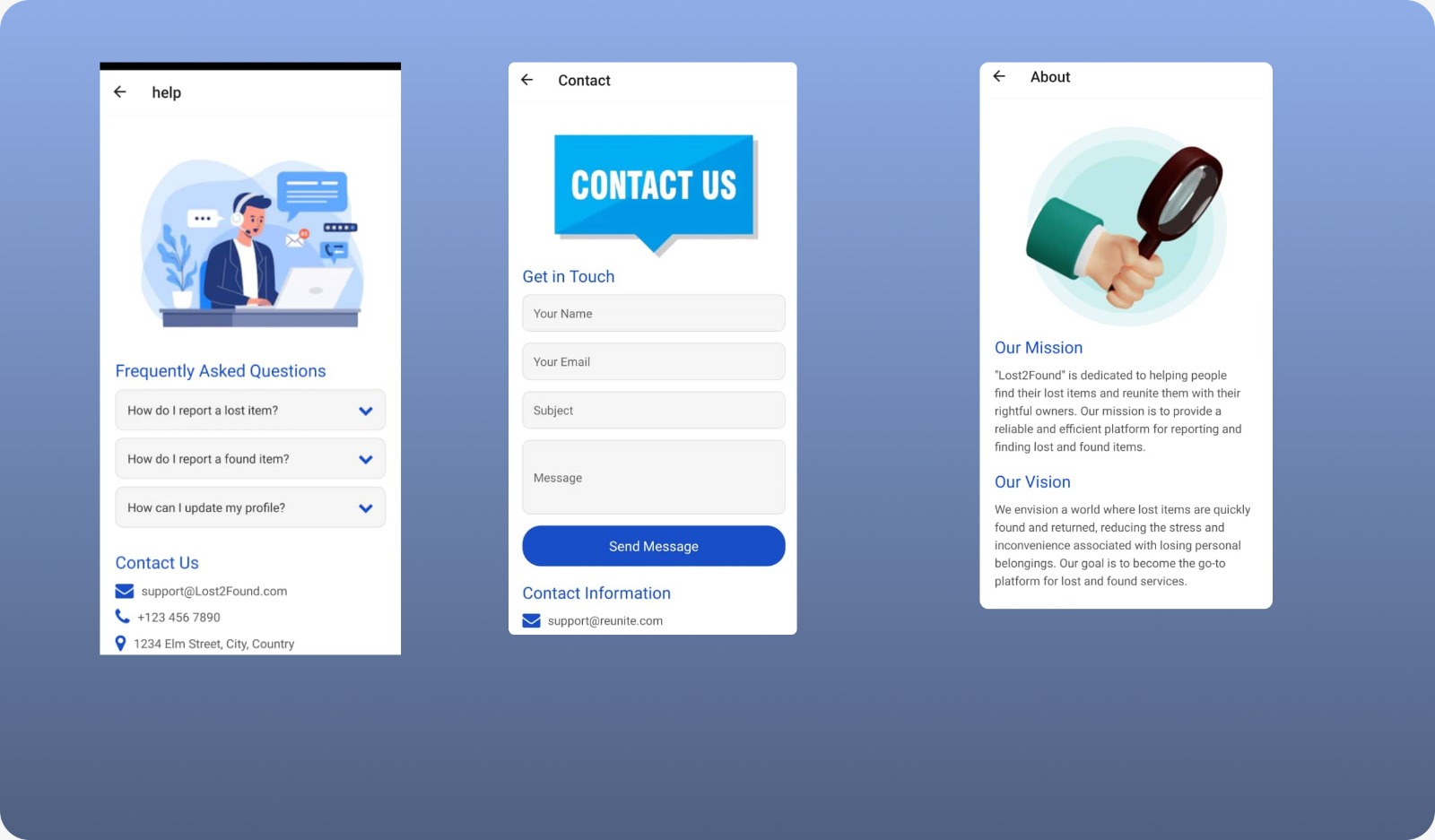
This page includes a user-friendly image representing customer support/communication, also *frequently asked questions* to make the user more comfortable in asking their own questions. This page also includes the app’s contact information to render more help to the user if needed.

* About page

With an image representing “search” reminding users of the app’s purpose, this page also includes more and detailed information about the app’s purpose

* Contact us page

Being simple and clear, this page simple offers the user the app’s contact information (email) to the user and give them the opportunity to send their message/feedback/complaint directly.



# UI IMPLEMENTATION

## Tools and Technologies

In this project, we utilized a variety of tools and technologies to build a cross-platform mobile application for archiving and retrieving missing objects.

Key tools and technologies used include:

React Native: A popular framework for building native mobile applications using JavaScript and React.

Expo: A set of tools and services built around React Native, which makes development, building, and deploying React Native applications easier.

React Navigation: A library for routing and navigation in React Native applications.

Vector Icons: We used react-native-vector-icons to include various icons, such as FontAwesome icons, in the app.

Expo ImagePicker: A library for selecting images from the device's library or camera.

Expo Sharing: A library for sharing content with other applications.

Navigation

Navigation in this project is handled using the React Navigation library. The setup allows seamless navigation between different screens in the application. The AppNavigator.js file configures the navigation stack:

Example from the AppNavigator.js page

Form Handling

Form handling in this project involves capturing user input through various form fields and processing the input upon form submission. We use React's useState hook to manage form data and handle submission logic within each relevant component

Example for the signInSignUp.js page,

Image Uploading

Image uploading is implemented using the Expo ImagePicker library. This allows users to select images from their device's gallery or take new photos using the camera. The selected image is then displayed in the image picker component, and users can change the image by clicking on the image picker again.

Example from UploadLostImagePage.js:

Notification System

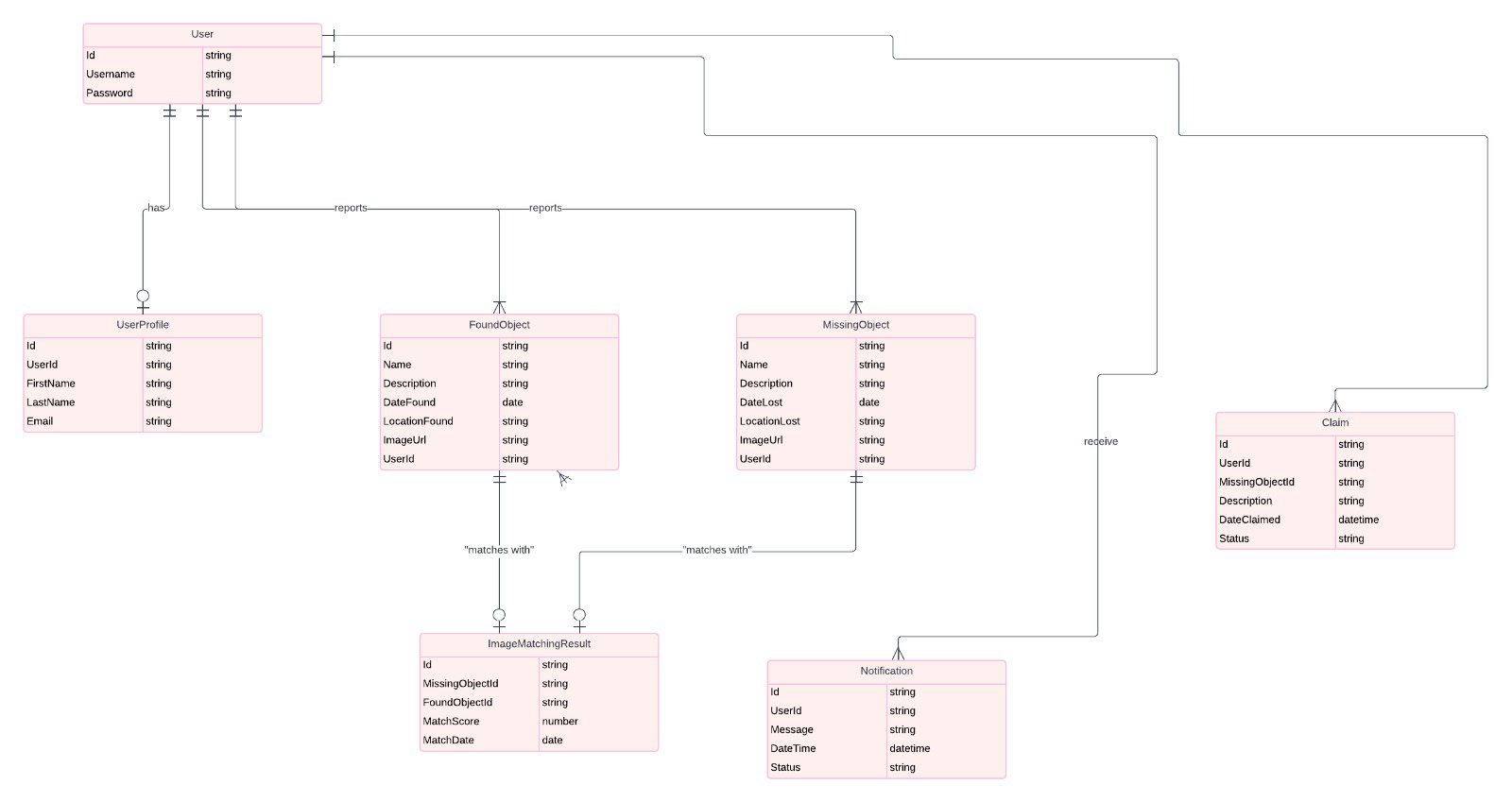
For the notification system, we used Expo's Notifications API. This API allows us to handle both local and push notifications. Here’s how we set up the notifications:

expo install expo-notifications

# Database Design

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

### Entity-Relationship Diagram (ERD)



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

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')

5. 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)

5. 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.

## DATABASE IMPLEMENTATION

For our backend implementation, we implemented the following functionalities

* 1. Firebase integration

Our app uses Firebase Authentication for user management and Firebase Firestore for data storage. Images are stored in Firebase Storage, and image metadata is stored in Firestore.

1. Firebase authentication:

Firebase Authentication is a service provided by Firebase that allows you to authenticate users to your app with minimal hassle. It supports multiple authentication methods including email and password, phone numbers, and popular federated identity providers like Google, Facebook, and Twitter.

How It Works:

* User Management: Firebase Authentication handles the backend processes involved in authenticating users. This includes creating user accounts, signing in, signing out, and managing user sessions.
* Secure Authentication: Firebase ensures that authentication processes are secure by using modern security practices and protocols. For instance, passwords are hashed and stored securely.
* Simplified Integration**:** Firebase Authentication provides SDKs for various platforms (iOS, Android, Web) and makes it easy to integrate authentication into your app with just a few lines of code.
* User Identification: Once authenticated, users receive a unique identifier (UID) which can be used to manage and associate their data within your app's backend.

**Code of user sign-in and sign-up**



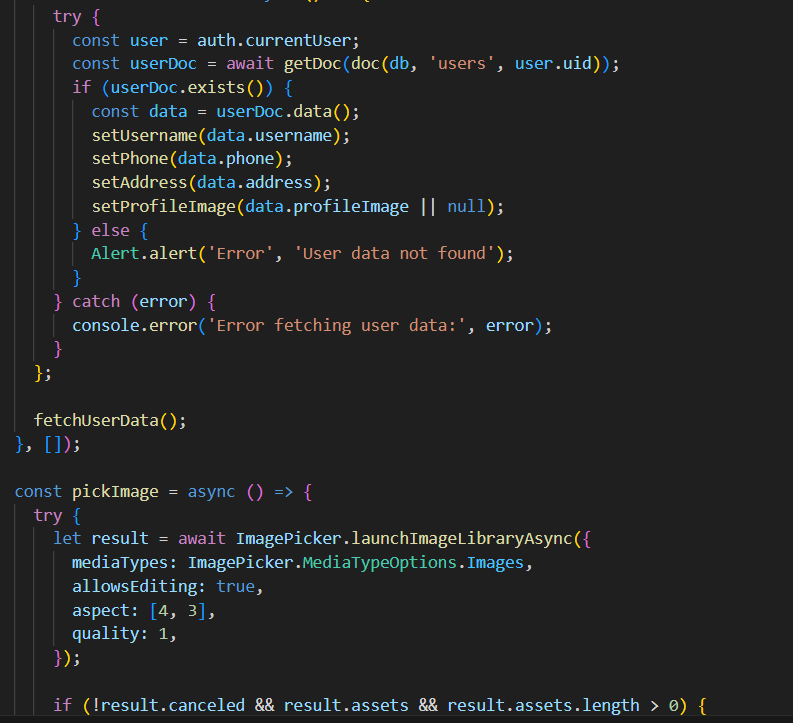
1. Firebase firestore**:** Firestore is used to store metadata about users, objects, images, and matches.

Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. It is a NoSQL database that stores data in documents, which are organized into collections.

How It Works**:**

* Documents and Collections: Data in Firestore is stored as documents, which are JSON-like objects. These documents are organized into collections. For example, a collection named "users" might contain user documents, each with fields like name, email, and profile picture.
* Real-time Synchronization: Firestore provides real-time updates. When data changes, Firestore syncs those changes to all clients listening to those updates, providing real-time synchronization across devices.
* Queries: Firestore supports powerful querying capabilities. You can query data in a collection with multiple conditions, order the results, and paginate them.
* Offline Support: Firestore has built-in offline support. Your app can read and write data to Firestore even when the device is offline. Once the device is back online, Firestore syncs any local changes back to the cloud.
* Security Rules: You can define security rules to control access to your Firestore data, ensuring that users can only read or write data they are authorized to access.

**Code for user profile edit**



1. Firebase storage**:** Firebase Storage is used to store image files.

Firebase Storage is a service for storing user-generated content such as photos and videos. It is built on Google Cloud Storage and is designed to scale from prototype to production.

How It Works:

* File Storage: Firebase Storage allows you to upload and store files such as images, videos, and other large binaries. These files are stored in Google Cloud Storage, which offers high scalability, redundancy, and security.
* Easy Integration: Firebase Storage SDKs for iOS, Android, and web make it easy to integrate file storage capabilities into your app. You can upload files directly from the user's device or from your server.
* Access Control: Firebase Storage integrates with Firebase Authentication, allowing you to control access to files based on the user's authentication state. You can define security rules to restrict who can upload or download files.
* Metadata and URLs: Each file stored in Firebase Storage can have associated metadata, such as MIME type and custom key-value pairs. Files are accessible via unique URLs, which can be shared or used within your app to display the content.
* Automatic Handling of Large Files: Firebase Storage handles file uploads and downloads efficiently, automatically resuming interrupted transfers and splitting large files into manageable chunks.

**Code of lost item image upload**



**Code of found item image upload**

****

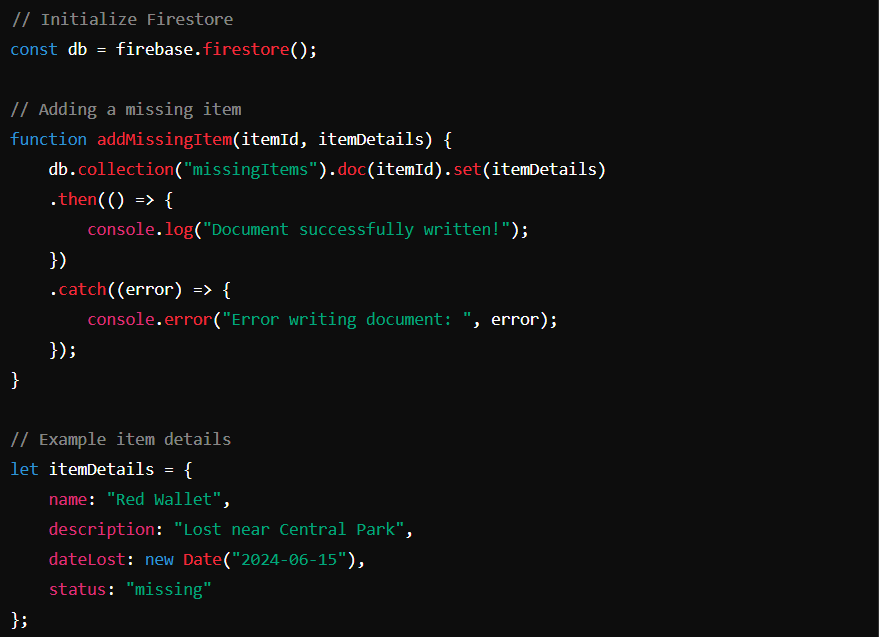
Data Management

* Data Insertion

Data insertion is a fundamental operation for the mobile application, which allows users to report its registration information and missing items. This process involves collecting data from the user through the mobile interface and sending it to the server, which then inserts the data into the database.

Firebase Implementation**:**

* User Action: The user inputs the item details in the mobile app.
* Firebase Interaction: The app sends this data to Firebase Realtime Database or Firestore using Firebase SDK methods.
* Data Structure: In Firebase, data is stored in JSON format. Each missing item is represented as a document in a collection (Firestore) or a node in the database (Realtime Database).

**Example: Inserting a Missing Item  
  
**

* Data Retrieval

Data retrieval involves fetching data from the database based on specific user queries, such as searching for items lost in a particular location or within a certain date range.

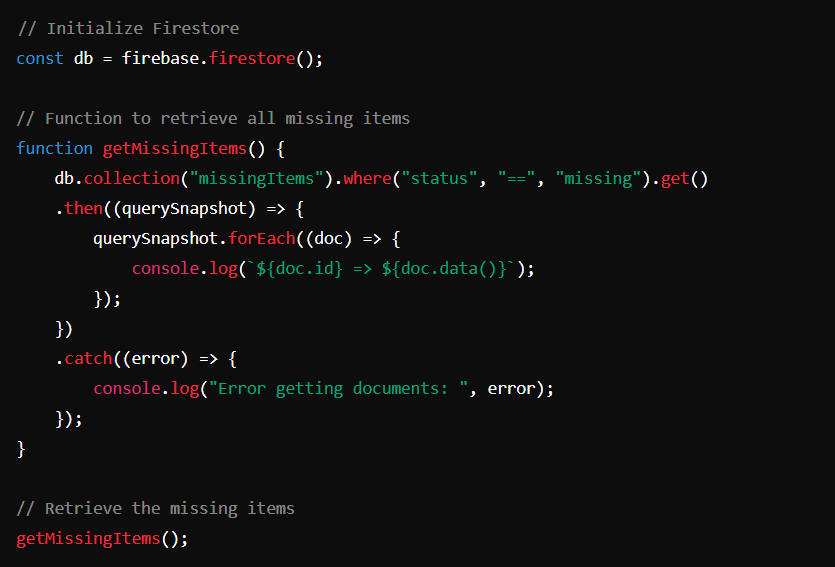
Firebase Implementation:

* User Action: The user enters the search criteria in the mobile app.
* Firebase Query**:** The app uses Firebase SDK methods to query the database. Firebase supports various query operations such as filtering by field, ordering, and combining multiple filters.

Example Description:

* The app constructs a query to retrieve documents or nodes that match the search criteria.
* Firebase executes the query and returns the relevant data to the app.

**Query**

****

**Filtering Mechanisms**

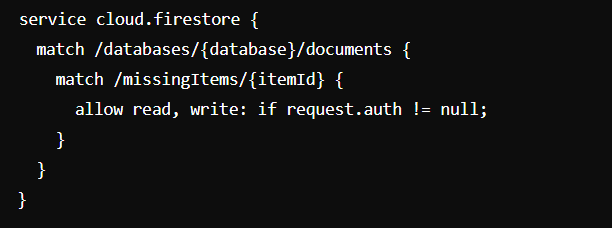
Firestore allows for powerful and flexible filtering of data using various query operators. Filtering helps the mobile application fetch only the relevant missing items based on specific attributes or conditions. This filtering mechanism ensures that only missing items are retrieved, ignoring items with other statuses.  
in our case : The .where("status", "==", "missing") clause filters documents where the status field is equal to "missing".

* **Data Security**

Data security outlines measures implemented to protect data stored in Firestore:

* **User Authentication and Authorization:** Firebase Authentication ensures only authenticated users can access or modify data, enforced through Firestore security rules.

Example Security Rule:



* **Data Encryption:** Data is encrypted both in transit (via SSL/TLS) and at rest (using 256-bit AES encryption), ensuring data remains secure during transmission and storage.
* Backup and recovery strategies : Firestore automatically handles backups and provides built-in recovery options to restore data in case of accidental deletions or corruption.

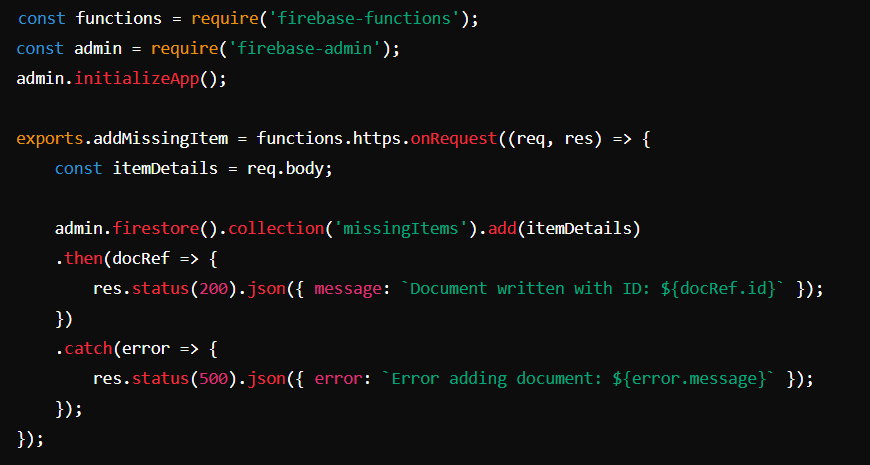
Application Integration

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.

#### 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

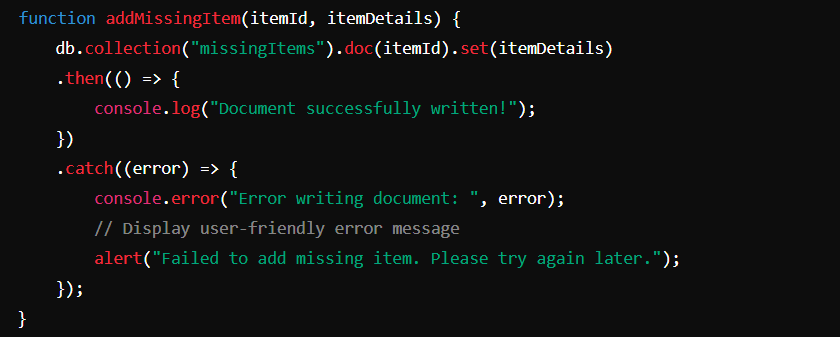
Describes the flow of data between the mobile application and Firestore database. It covers how user-generated data (such as missing item reports) is transmitted to Firestore and subsequently retrieved and displayed within the application.

* 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.

Technology Stack

List the technologies used in the project, including:

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

# **CONCLUSION**

The development of a Mobile-Based Archival and Retrieval of Missing Objects Application using Image Matching represents a significant advancement in the effort to locate and return lost items. By harnessing the power of image recognition technology and the widespread availability of smartphones, this application offers a practical and efficient solution to a common problem. Through careful requirement gathering, prioritization, and the implementation of robust system specifications, the project addresses the limitations of traditional methods and existing digital solutions. The proposed application not only enhances the probability of recovering lost items but also provides a user-friendly interface, ensuring accessibility and ease of use. As a result, this innovative approach has the potential to alleviate the financial and emotional burdens associated with lost belongings, fostering a greater sense of security and community trust.

# **REFERENCES**

[1] M. Ghazal, Y. Alkhalil, S. S. Ali, F. Haneefa, and E. Rashed , Archival and Retrieval of Lost Objects using Multi-feature Image Matching in Mobile Applications, International Journal of Computing and Digital Systems (2016).

[2] Ghazal, H., Haneefa, G., & Al-Faouri, A. M. Mobile-Based Archival and Retrieval of Missing Objects Using Image Matching. International Journal of Computer Science and Information Security (IJCSIS), (2015): 92-100.

[3] YG Yang, QQ Zhao, SJ Sun, “Novel quantum gray-scale image matching,” OPTIK, 126(22), 3340-3343(2015).

[4] J Ma, H Zhou, J Zhao, et al, “Robust feature matching for remote sensing image registration via locally linear transforming” IEEE Transactions on Geoscience and Remote Sensing, 53(12), 6469-6481(2015).

[5] X Qu, Y Sun, Y Gu, et al, “A high-precision registration algorithm for heterologous image based on effective sub-graph extraction and feature points bidirectional matching,” Changsha, China: IEEE, 1835-1839(2016).

[6] Y Lecun, “Stereo matching by training a convolutional neural network to compare image patches,” Journal of Machine Learning Research, 17(2016).

[7] C Jin, J Guo, “Image semantic annotation approach based on the feature matching,” Kunming, China: Advances

in Intelligent Systems and Computing, 250, 281-288(2014).