# THE KELKAR EDUCATION TRUST’S VINAYAK GANESH VAZE COLLEGE OF ARTS, SCIENCE AND COMMERCE

**(Autonomous)**

**MULUND, MAHARASHTRA, 400081**

# DEPARTMENT OF INFORMATION TECHNOLOGY



**CERTIFICATE**

This is to certify that the project entitled," ", is bonafide work of bearing Roll No: submitted in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY.

**Internal Guide Head Of Department**

**External Guide**

**Date: College Seal**

DECLARATION

I here by declare that the project entitled, done at, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university. The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Name :

Signature :

The Kelkar Education Trust’s

V G Vaze College of Arts, Science and Commerce

(Autonomous)

**Find My Stuff**

**A Project Report**

Submitted in partial fulfilment of the Requirements for the award of the Degree of

**BACHELOR OF SCIENCE**

**(INFORMATION TECHNOLOGY)**

**By**

Aaditya Sheelkumar Pal

4045A029

**Under The Esteemed Guidance of**

**Mrs. Rakhee Rane**

**Assistant Professor**

****

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**V G VAZE COLLEGE OF ARTS, SCIENCE AND COMMERCE (AUTONOMOUS)**

***(Affiliated to University of Mumbai)***

**Mulund,** **400080 MAHARASHTRA**

**2023-2024**

FIND

MY STUFF

A Lost and Found Application

**Synopsis of the project**

**Title of the project**

Find My Stuff

**Statement about the problem**

To develop an android application that will help the college students and faculty find their lost goods. Also, they can report any lost items that are found. The various functions provided are reporting lost goods, posting and listing the found items, etc.

**Why this topic?**

Finding lost items proves to be very difficult and requires plenty of enquiring throughout the college premises, which requires a lot of hard work. There is no way or medium to contact someone who might potentially possess the lost item. Also, if someone does manage to find an item, they can’t report it or are unable to reach out to the owner. Hence, the aim is to connect the owner and the potential student or faculty possessing the item.

**Objective and Scope**

The scope of the project is Vaze College and the students and faculty currently a part of the college.

**Objectives: -**

1. To reduce the efforts required to enquire multiple people throughout the college.
2. To find and return the goods to the owner.
3. To provide a platform to list found items and to find its owner.
4. To reduce the time and effort required in the process.
5. To connect the owner and potential student or faculty possessing the item.
6. To increase the probability of finding valuable items.
7. To help the reported item listing to reach a large number of people

Methodology

For this system, I would use an iterative approach. The advantages of this approach are: -

1. Feedback from one interaction may improve the other iteration
2. Increments are delivered and developed.

Disadvantages are: -

1. Later, increments may require modifications to earlier increments.

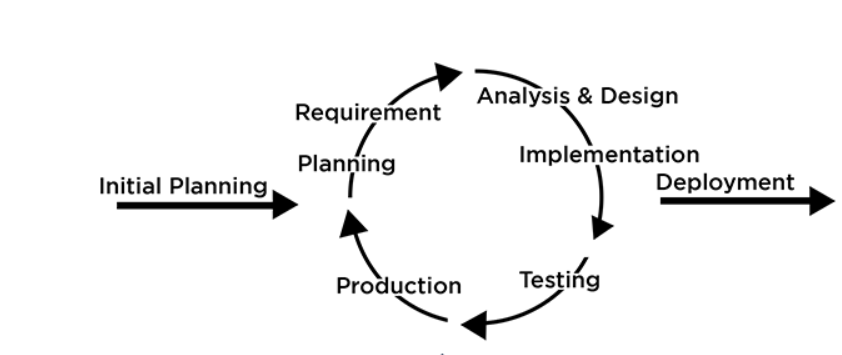
[](https://artoftesting.com/iterative-model)

Figure 0.1 Iterative development methodology

**Proposed Architecture**

The application would be developed using a client -server 2 tier architecture.

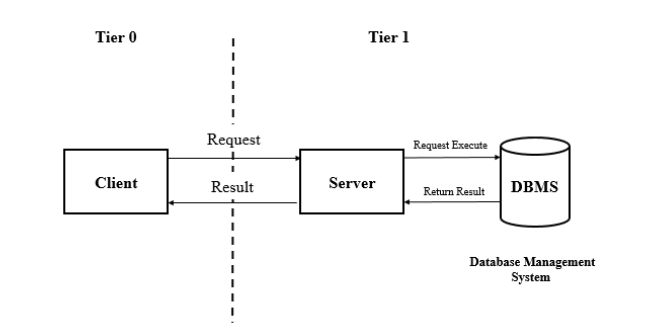
[](https://www.collegenote.net/curriculum/web-technology-csit/84/468/)

Figure 0.2 Client server 2 tier architecture

**Requirements**

1. Software Requirements
   1. Operating system: 64-bit Windows 8 or higher
   2. Database: MySQL
   3. PHP
   4. XML
   5. Java
2. Hardware Requirements
   1. Processor: Intel Core Duo 2.0 GHZ or AMD Athlon CPU or newer
   2. Ram: minimum 8GB
   3. Hard-Disk: 256 GB SSD, 8GB of available disk space
   4. Monitor: 1200 x 800 minimum screen resolution
3. Platform
   1. Android Studio

**Contribution**

Locating and finding items becomes very stressful and difficult as it involves enquiring a bunch of students and staff. These students and staff might not always be the right people to help in finding the items. This app helps simplify this stressful process. It would reach out to a large number of students and hence the chances of finding the items increases, saving time and efforts in doing so.

**Conclusion**

The application aims to streamline the difficulty of finding and reporting a lost item. By doing do, it saves a lot of time and efforts in locating the lost item by connecting and reaching out to a large number of students.

**Chapter 1**

**Introduction**

* 1. **Background**  
     Losing items around a college campus is a common phenomenon. Although, a physical lost and found department is in place which is helpful, but it presents a number of problems. The major issue is that there is no way of informing the owners that their items have been found. The students have to frequently visit the office and enquire about the lost items. Also, students circulate Whatsapp messages around groups enquiring about their items, but it is unable to reach a massive audience. If someone does manage to find a lost item, they have no means to find the owner and the lost and found department has no way to notify the owner about it. There is no system that facilitates a communication between all the parties. Facilitating a communication and a connection while making the process simpler and effective between the concerned students and the department, is the main goal of the system.
  2. **Objectives**

1. To provide a platform for students and faculty to report lost and found items.
2. To help students and faculty to locate and search for missing items within the college premises.
3. To reduce the time and efforts put to report and look for lost or found items.
4. To increase the chances of recovering the misplaced items by providing a common platform for listing lost or found items.
5. To make it easier to connect the owner and the potential person who might have the item in his possession.
6. To provide a reliable system to manage lost and found items.  
   1. **Scope Purpose and Applicability**
      1. **Purpose**The major purpose that the system servers is that it provides a common platform for all the students and faculty to report lost or found items and to streamline the process by making it convenient and easy for owners to recover their lost items. The need for the system arises as there is no such system in place and the process to find a lost item is very time-consuming and tiresome.
      2. **Scope**The system is developed for students and faculty members currently a part of the V.G. Vaze College. The app will deal with lost items such as personal belongings, clothing, jewellery, books, gadgets and would exclude basic stationary items such pens, pencils.

Some limitations that might occur could be the type of images uploaded and data privacy of faculty members.

* + 1. **Applicability**

The application though specific to the college, it can be used in various other places as like: -

* Hotels – Guests can find their personal belongings that have been misplaced.
* Airports – Can be used to find luggage and other belongings such as electronic gadgets, passports, etc.
* Malls – People can use it to find their bags, etc.
* Offices – Electronics such as laptops, mobiles and various other important files are quite oftenly misplaced. The app can be used to find these items.
* Public places – Hospitals, Parks, Societies, etc can also use this app.

**Chapter 2**

**Survey of Technologies**

1. **IDEs**
2. **Android Studio**

It is the official integrated development environment (IDE) for developing android apps built on JetBrains’s IntelliJ IDE. It uses a Gradle-based build system, Android Emulator, code templates and GitHub integration. It is a one-stop place for development of android apps. It supports various programming languages such as Java, Kotlin or C++, while also supporting various frameworks. It receives the latest updates directly from Google keeping it up-to-date with the trends. It also has support for a large array of third-party plugins. Superior android development, project structure, code completion and refactoring, emulation, etc are some of the many advantages and features of Android Studio.

1. **Eclipse**

It is an IDE used primarily for Java development. However, it supports various other languages like C, C++, JavaScript to name a few. For several years, a version of Eclipse with an Android plug-in was recommended, But Google ceased support for this plug-in, causing developers to shift to Android Studio for Android development.

1. **NetBeans**

It was originally developed by a student of Prague University. It is similar to Eclipse and is primarily known as Java IDE. Android development is supported using plug-ins which are not supported to a large extent today.

1. **Visual Studio Code**

Very commonly referred to as VS Code, it is an open-source code editor made by Microsoft with the Electron Framework for Windows, Linux and MaxOS. It includes vast number of features such as support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring and embedded Git. Users can change the theme, customize keyboard shortcuts and preferences and install extensions that add functionality. A notable feature is the ability to create extensions that add support for new languages, themes, debuggers, etc via plugins such as the Dart plugin which VS Code utilizes when used to develop Dart apps for which the Flutter framework is used. Good and extensive plugin library is also a very prominent feature of VS Code.

**Why have I selected Android Studio?**

Currently, I chose Android Studio as it the official IDE, hence receiving continuous recent updates to the IDE as well as the plugins. It provides all the necessary features while being highly feature rich and offers an extensive way of debugging and optimizing the application.

1. **Frameworks**
2. **React Native**

It is an open-source cross-platform development framework developed and maintained by Facebook. It uses React, a flexible JavaScript Library to build apps for both Android and iOS. It makes app development much easier and faster by having a single codebase and uses native platform specific built-in components and APIs that give great performance. Having a single codebase reduces development time and cost while having elegant interface across platforms.

1. **Flutter**

Developed by Google, it is also a free, open-source mobile framework for developing applications. It simplifies the multi-platform development process to craft excellent native interfaces. Flutter is written in Dart language and is a method to implement hybrid app development. Using a single codebase, it uses Google’s rendering engine called Skia to develop visuals. It boasts a hot reload functionality which allows for continuous testing without having to restart applications. It is used to develop fast, high- quality applications for iOS and Android in record time from a single codebase.

1. **Xamarin**

It is also an open-source platform for building modern applications for iOS, Android and Windows applications with .NET. It features a friendly development environment with an abstraction layer that manages communication of shared code with underlying platform code. It allows developers to write all the business logic in a single language while achieving native performance, look and feel on each platform. Its major unique feature is that it uses .NET and C# for cross-platform applications.

1. **Ionic**

It is designed for developers who are familiar with web development and want to develop hybrid and interactive mobile apps. Its rich and complete set of elements, gestures, animations and software tools enable the developers to develop high quality mobile, desktop and Progressive Web Apps from a single codebase. It is easily integrable with Angular JS and utilizes functionalities like Bluetooth, fingerprint authentication, etc and also uses emulators, live reload and logging to offer amazing performance. It also uses Cordova plugins to access camera, GPS, etc in an easy manner. It is the most developer-friendly framework and uses HTML, CSS and JavaScript.

1. **Apache Cordova**

It makes use of HTML, CSS and JavaScript for building mobile applications. The open-source platform consists of a set of pre-defined plugins that provide access to the device’s camera, GPS, file system, etc. The developers can develop apps that are compatible for more than one platform without re-implementing it with the language of each platform. The resulting applications are hybrid, ie they are neither truly native nor purely web-based. This causes the applications to run slower than native applications. They are not Web apps because they are not packaged as apps for distribution. The slow or reduced performance is a major drawback however it allows developers to add more functionality using JavaScript.

1. **Databases**
2. **MySQL**

It is the most popular open-source relational database management system. Its relational nature helps to organize the data into one or more data tables. The structured data can be inserted, extracted and modified. Its major features are its ability to manage users, allow for network access, facilitating testing and creation of backups. Query caching, Unicode support, multiple storage engines, SQL support are some of its prominent features.

1. **PostgreSQL**

It is an advanced, enterprise class open-source relational database that supports both SQL (relational) and JSON (non-relational) querying. It is highly stable, backed by more than 20 years of community development which has contributed to its high levels of resilience, integrity and correction. It also has a rich support for advanced data types and performance optimization. However, it is more power-hungry and performs slower when compared to MySQL.

1. **SQLite**It is an embedded, file-based RDMS. The application does not run under a separate server process. The serverless architecture enables the database to be cross-platform compatible. The SQLite adheres to ACID properties to safeguard transactions against memory allocation failures. Its compact libraries and small footprint make it fit for applications that do not require a heavy database and because it is stores locally, it can cache data quickly and easily without delaying.
2. **Firebase**It is a backend-as-a-service. It is backed by Google and is an application development software that enables developers to develop iOS, Android and Web Apps. The cloud computing services include hosting databases, services, authentication, and integration of applications, including Android, iOS, JavaScript, PHP, C++, etc. It uses NoSQL (not only SQL) as a real-time database. Its blazing speed, cloud-based nature and its ability to handle large sets of data along with numerous features make it a viable option.
3. **AWS Dynamo DB**

As part of a larger ecosystem of Amazon Web Services (AWS), this database is a NoSQL database that is known for its speed and efficiency when it comes to retrieval of information and data from the system. It is a key-value database, so there are variety of data types that can be stored within this system. It is a highly scalable and a complex database system for developers that are working with the applications that need to manage big user data and constant engagement.

1. **MongoDB**

It is the most popular NoSQL database and offers many features geared towards the development of mobile applications. As a document-based database, it is proficient with the JSON data-interchange format, making the storage of web pages and other documents like chat logs and messages. It is also highly scalable and has the capacity to grow with the changing needs of the application.

**Why I am using MySQL for my application?**

I will be using MySQL database for its widespread popularity, its speed and its relational nature. It can be accessed via a server and can be stored or hosted remotely easily which ensures the availability of the app at any given time. The data is stored in a relational manner, ie it is easier to understand as it is stored in a structured format.

1. **Other**
2. **PHP**

PHP stands for Hypertext Pre-processor and is a widely used open-source general purpose scripting language. The server-side scripting language is used to perform operations and access databases with the most popular one being MySQL. Using PHP, the data can be inserted and also fetched from the databases making the applications dynamic.

1. **Room Library**

It is a persistent library that allows fluent access to the database by creating an abstract layer over SQLite. It simplifies the process of adding a structured SQL database to the application.

1. **Firebase**

This backend-as-a- service includes many features like authentication of email, phone numbers by sending OTP to the users. Also, firebase offers services to send messages to other users as well as send push notifications when required. It offers a vast library of devices on which the application can be simulated and hence perform testing of the applications.

**Why will I be using PHP?**

I will be using PHP as a server-side scripting language, due to its platform independent nature. It also supports major web servers and performs its operations with greater speed and is also commonly used along with MySQL database, hence making it easier to troubleshoot any issues related to the availability of forums due to its popularity.

**Why Firebase?**I would Firebase Services to send push notifications, authenticating users via email and phone number, etc. Firebase is developed by Google and hence integrates smoothly with Android Studio IDE.

**Chapter 3**

**Requirements and its Analysis**

* 1. **Problem Definition**

The process of locating and trying to reunite a lost item with its rightful owner is quite frustrating, tiresome and time-consuming. To overcome this, the lost and found application will be put to use. The users of the system mainly include students and the faculty, both of whom can report lost items and post, if they happen to lose an item. The system will revolutionize the process by simplifying and speeding up the process. The users can contact the owner of the item, keep track of the progress of their lost / reported item, view the lost items in an organized manner, etc. The administrator, for eg the lost and found department, would be the authority responsible for the application’s management. The communication feature among two users of the application would be an important step in verifying the owner of the item. All these functionalities would allow a large number of students to connect and create a community, increasing the chances of locating an item effortlessly.

**Sub-Systems**

1. Login/Registration:
   1. The users would have to register for the first time.
   2. Once the users are registered successfully and authenticated, they can access the rest of the app.
   3. Once logged in, the users don’t need to log in every time they wish to use the app, i.e they remain logged in unless they explicitly log out.
2. Posting/Reporting a lost item:
   1. This sub-system is concerned with users being able to post or report a lost item by filling out and providing the item’s description and the contact information along with the item’s image as well.
3. Item viewing:
   1. This sub-system displays all the lost items as well the reported items to the user in an organized manner.
   2. It filters the items according to various categories, date, etc.
   3. The uses can also view items that they have posted and also track its status.
4. Profile management:
   1. The users can view their information and also edit their contact details.
5. Communication:
   1. The users, for eg: the owner of a lost item and its finder can communicate with each other through the in-app messaging.
   2. The users will be notified about important updates, matches or messages via in-app notifications and emails wherever necessary.
6. Verification and matching:
   1. Once a new item is reported as lost, it is checked if any user has already found that item and the user will be alerted accordingly.
   2. If a user claims to be the owner of an item or claims to have possession of an item, they need to answer a few questions regarding the item in order to verify their ownership.
7. Administration
   1. The admin can manage users, monitor reported/ lost items and answer queries.
8. User Assistance
   1. This sub-system would allow the users to contact the administrator to overcome any difficulties or issues that they are encountering.
   2. **Requirement Specification**
      1. **Requirement Gathering**

The various ways to gather requirements are: -

1. **Survey/Questionnaire:**

Questionnaires and surveys are basically a set of questions that are distributed among the stakeholders in order to get an understanding of the system. The questionnaires can be distributed to multiple stakeholders at the same time hence saving time and reducing the efforts required to gather requirements. One disadvantage of this method is the lack of flexibility to change the questions or add follow-up questions based on the response received.

1. **Interviews:**

Interviews are usually conducted one-on-one. Interviews provide the flexibility of asking follow-up questions which can help gather more specific and detailed information. But the interviews can be time-consuming and the interviewee must have in-depth knowledge of the current system.

1. **Brainstorming:**

In this method, the system is approached from every point of view and all the possible scenarios and their outcomes are thought of which include what-if scenarios and blue-sky ideas. The general idea is to break away from the existing convention and to figure out the various situations that take place in the systems to gather detailed requirements of the system. Role-play is an extension of brainstorming where various roles are enacted to replicate the situations and scenarios that take place which can help to develop a better understanding of the system.

1. **User Observation:**

User Observation is one of the best to fully understand and discover how people and technology in the current system operate and behave. It gives us a realistic idea of how things actually work. It can be categorized into active and passive. Active observation takes place when the people being observed are questioned about the actions that they carry out to gain a better understanding of the process. Passive observations are better at getting feedback without any communication involved.

1. **Prototyping:**

This method gives the users a chance to try out how their system would look and feel. This results in continuous feedback and based on the feedback, continuous changes and improvements are made, making it an iterative process. The prototypes are reverse engineered to discover the requirements along with the various other features and functionalities that would satisfy the stakeholders.

**Method I used and why?**

I have used surveys /questionnaires to gather the requirements from the students of the college who constitute the majority of the user base. Apart from that brainstorming was used to figure out other possible scenarios and to think of their possible outcomes. I used Google forms to distribute the questionnaire. I also used interviews, asking questions about the current system in place and how things in the college operate. These interviews were answered by my guide and an administrative staff.

The interview questions included:

* How does the current lost and found department operate?
* Who can be the admin for the app?
* How AI can be integrated in the app?

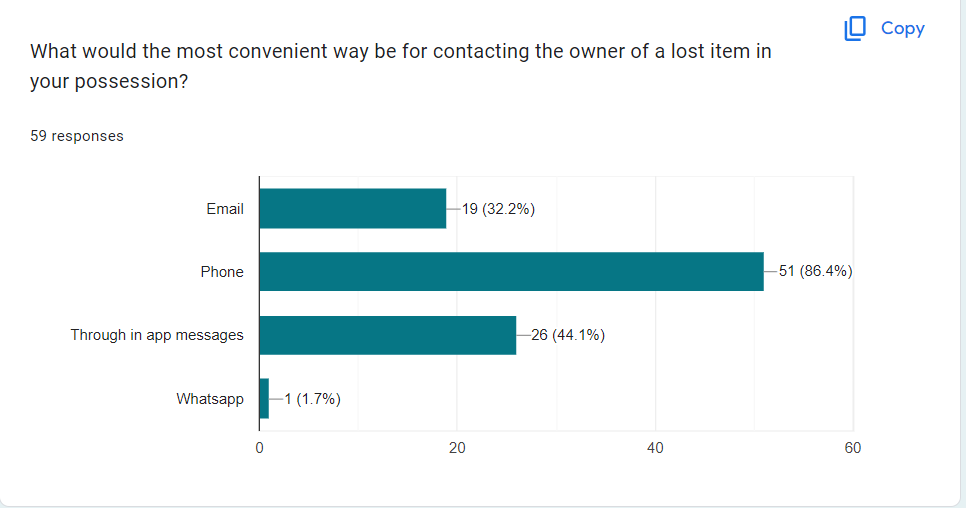
The link for the google is as follows:

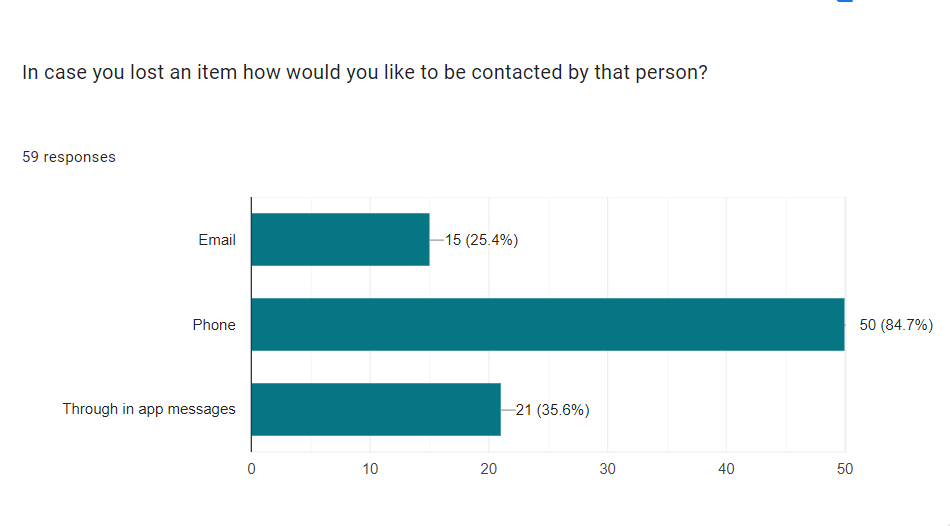
<https://docs.google.com/forms/d/e/1FAIpQLSc6Pw-xDYexQH5Y9NslgVcCAO9crRhwNjof6EDWv-gym2JtuQ/viewform?usp=sf_link>

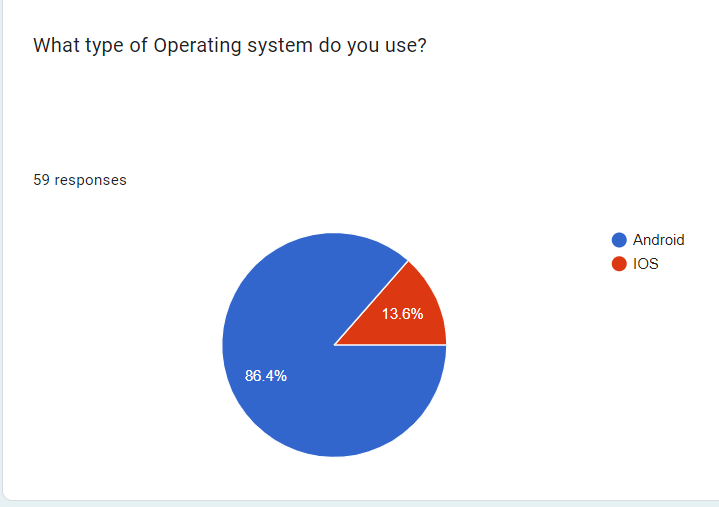
The link to the spreadsheet is as follows:

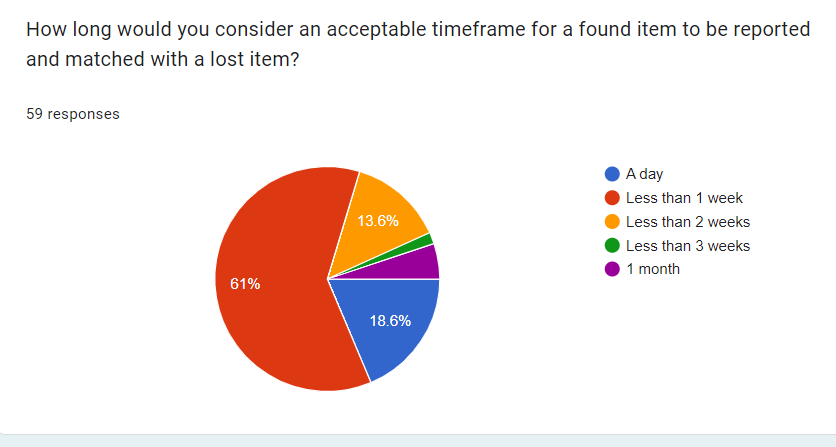
<https://docs.google.com/spreadsheets/d/13A8RAn2stw_Lg6NtZAOh6Z26_QbrinDOh-Adczxg9gQ/edit?usp=sharing>

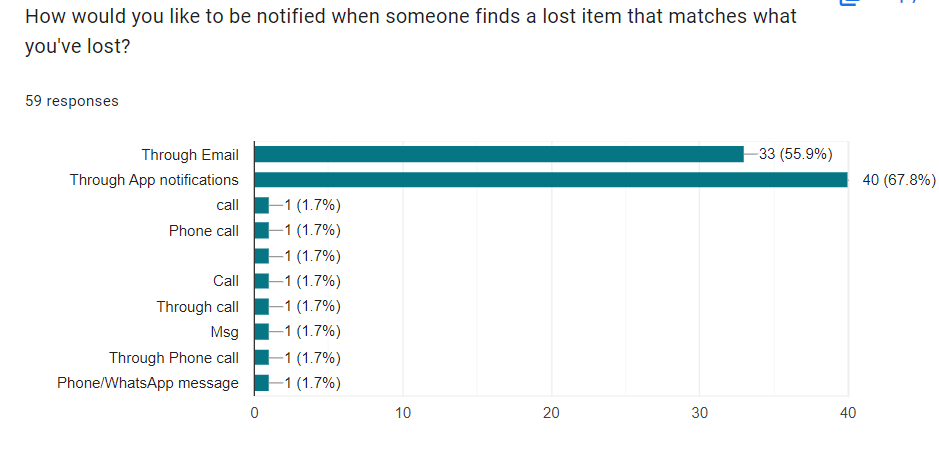
The following are the results:

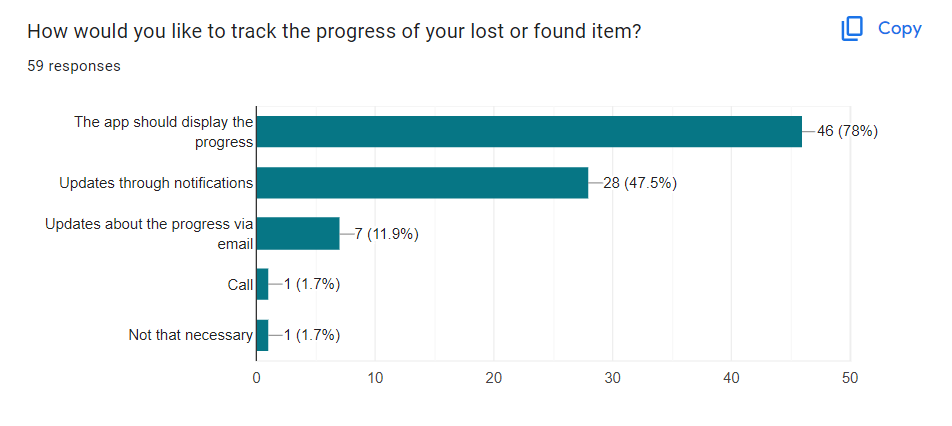


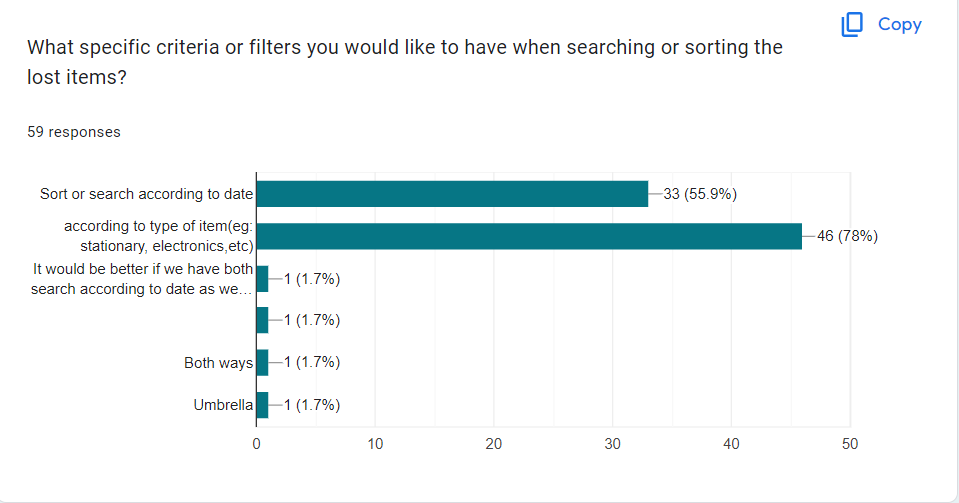


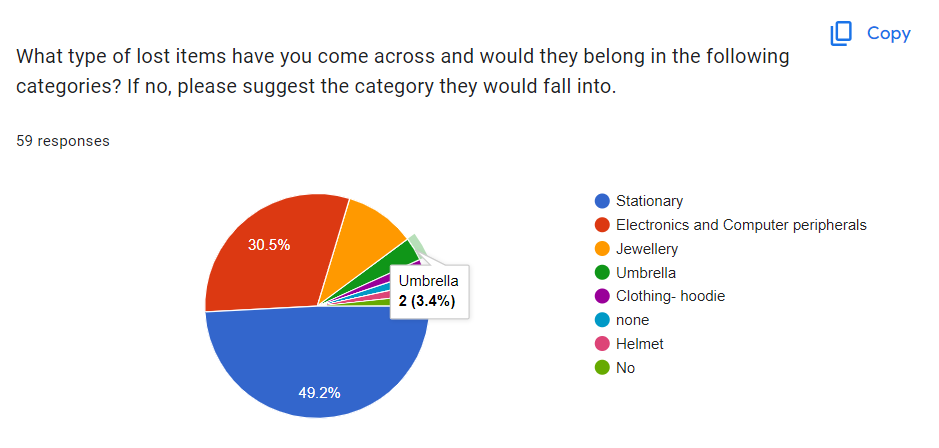


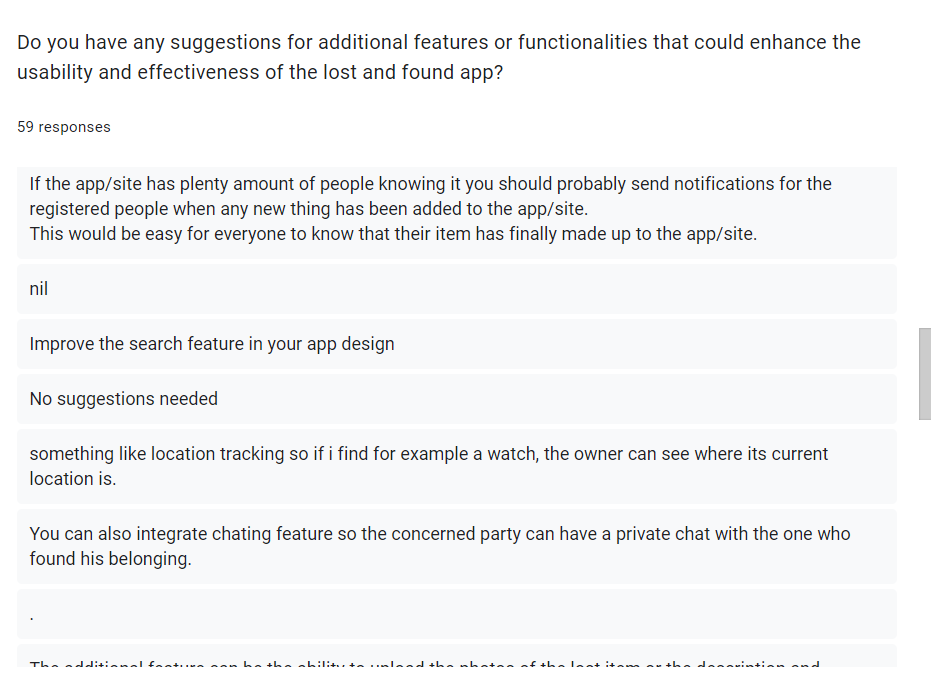


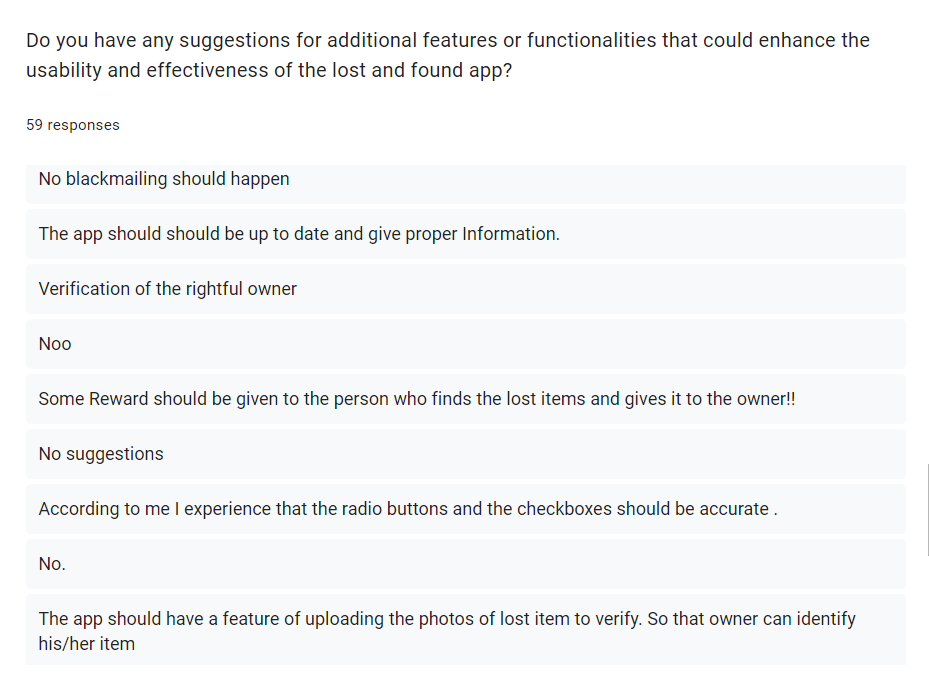


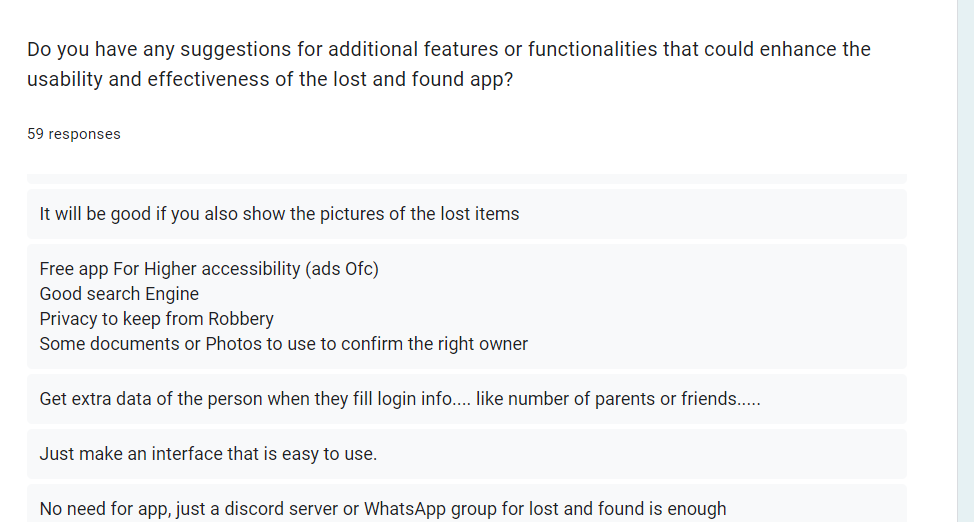












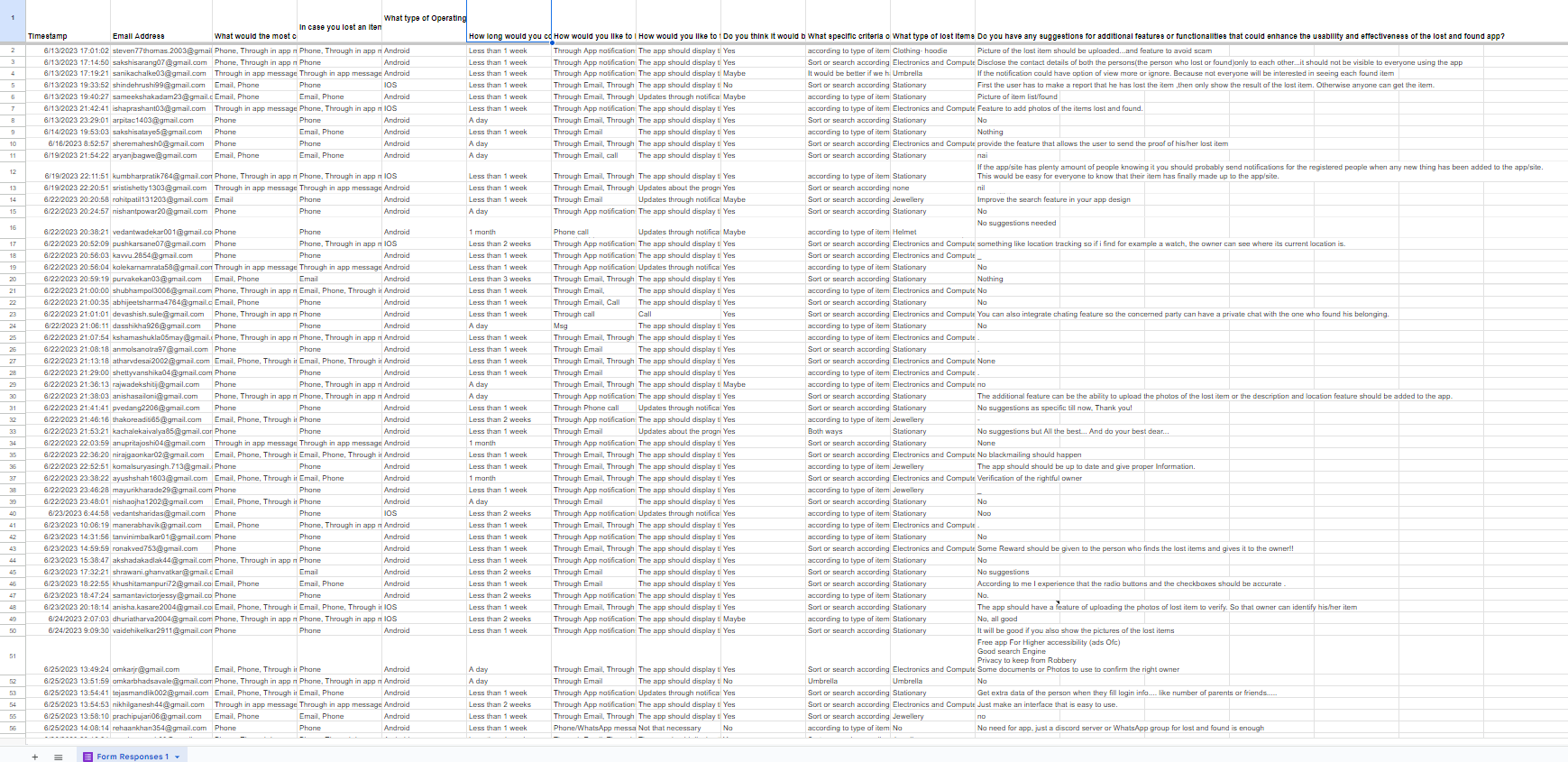
[](https://docs.google.com/spreadsheets/d/13A8RAn2stw_Lg6NtZAOh6Z26_QbrinDOh-Adczxg9gQ/edit?usp=sharing)

Figure 3.1 Requirement Gathering through Google forms

**Requirement Analysis**

All the requirements gathered are stated in simple English statements:

1. The users of the system should include the students who are currently pursuing 11th, 12th or a degree from the college or a faculty of the college. One user from the office would be the admin for the application.
2. User Registration
   1. The users if they are students should be able to register using their control id.
   2. Administrator would not need to register as they will be managed from the admin section
   3. The faculty members would have to register using their email which is unique and contains ‘@vazecollege.net’.
   4. The registration page should provide a way to distinguish between the student registration and faculty registration.
   5. The password should be minimum 8 characters containing at least one digit and one character.
   6. The username of the students would be optional and control id would be used to identify them uniquely, but email would be used as the username for the faculty.
3. User Login
   1. The user will have to enter their credentials like registered email or control id depending whether there are a faculty or a student respectively.
   2. The user should be displayed an appropriate message when the authentication fails.
   3. The user need not login every time they visit the app until they explicitly log out.
   4. The user should also have the option of “forget password”.
4. Posting/Reporting a lost Item
   1. When reporting a lost item, the users should be able to upload a photo of the same.
   2. The following details of the items should be collected: - date of loss, time of loss, location, unique identifiers like lost items name, description, color, category, etc.
   3. Category should include electronics, clothing, jewellery, stationary, riding gear and other.
5. Item viewing
   1. The items should be divided into two - items reported as lost and items reported as found.
   2. The item posts should be arranged from latest and recent posts to oldest posts.
   3. A lost item should remain in the list for a maximum period of one month.
   4. The user should be able to filter the items according to category, date, time and being able to search the item.
   5. The users should be able to see the items they have reported and also track its progress.
6. Communication
   1. The finder of an item or the owner of the item should be able to communicate through in-app messaging.
   2. Notifications should be sent to the users if an item of the same category is reported.
   3. The in-app messaging should only take place between the two users only when they are verified by answering the questions correctly and are the owner and the finder of the item.
7. Verification and matching
   1. When reporting an item as lost, the user posting it should also include a few security questions and answers. These questions will be asked to the user who claims to be the owner/ finder. Only after successful verification, the users should be able to chat.
   2. If an item being reported as found is already reported as a lost item, the owners should be notified about this match via in-app notifications and email.
   3. The details of the users are only revealed after verification.
8. Administration
   1. The admins can suspend, delete, add a user profile.
   2. The admins should be able to delete lost item postings.
   3. The admins should be able to add faculty that may not be able to register via the normal procedure.
   4. The admins should be able to update the status of the item saying that it is found or with the lost and found department.
9. Others
   1. After successfully verifying that the user is an owner/finder, the application should provide a way to owner and finder to call the next person directly from the app.
   2. For the faculty and students, emails will be verified via OTP sent to their entered emails.
      * 1. **Functional Requirements**
10. **User registration**
    1. The students need to register using their control id and the faculty needs to register using their” @vazecollege.net” email id.
    2. The registration page should be separate for faculty and students.
    3. During registration, the password should be of minimum 8 characters in length containing at least one digit and one character.
    4. The username will be email.
    5. Every student and faculty should be only having one account
    6. The student or faculty should be actively be a part of the college.
11. **User Login**
    1. During login, the “forget password” option should be present.
    2. If authentication fails, the appropriate message needs to be displayed.
    3. The users need not log in every time they visit the app, until they explicitly log out.
12. **Posting/ reporting lost items**
    1. The users should be able to upload a photo of the lost item.
    2. The following details should be collected from the users when they are reporting a lost item: -
       1. Date
       2. Time
       3. Location
       4. Description
       5. Unique identifiers like model number, etc
       6. Category
    3. The user who is posting will also have to provide minimum 2 questions and answers which will be asked to the users during verification.
    4. The reported items should fall in these categories:
       1. Personal Belongings
       2. Clothing and footwear
       3. Electronics
       4. Riding gear (Helmets)
       5. Sports equipment
       6. Bags and backpacks
       7. Miscellaneous
    5. Date range should be between one month from the current date of the current year.
13. **Item Viewing** 
    1. Items that are listed should be categorized into these items – ‘reported as lost’ and ‘reported as found’.
    2. The reported items should be arranged in the descending order of their date posted i.e the most recent ones appear first or on top.
    3. A lost item remains in the list for a maximum of one month i.e 30 days from the date it was posted.
    4. The users should be able to filter the items according to the date of loss, posted date, location, category and also able to search a specific item.
    5. The users should be able to delete and mark the lost item as ‘found’, for the items that they have reported.
    6. The users should also be able to see the progress of their reported item.
    7. The users should be able to update the status of their posts/ reports if the item was found.
    8. They should be able to delete the report as well.
14. **Communication**
    1. The owner and the finder of the item should be able to communicate via in-app messaging.
    2. The users should receive push notifications when the status of their item is updated, a match is found, a new item in the same category is posted and other important updates.
    3. The communication between the two users should take place after successful verification.
    4. The contact details of the users should be revealed only after the successful verification.
    5. The finder or owner should be able to call the other concerned user and faculty.
15. **Verification and matching**
    1. The users should be notified when a match for their lost reported item is found via notification and email.
16. **Profile**
    1. For students and faculty, they should be able to edit their phone number.
    2. Users should be able to update their password.
17. **Administration**
    1. The admins should be able to suspend, delete or add a new user.
    2. The admins should be able to delete, and update the status of the reported item if the lost and found department handles the handing over of items to the owner.
    3. The admins should be from the lost and found department.
    4. The admins should be able to manage other admins and be able to add faculty that can’t be registered through normal procedure.
18. **User Assistance**
    1. The users should be able to report any harmful, unethical activity to the admin via the email.
    2. The users should be able to contact the admin incase they need any help.
       * 1. **Non-functional Requirements**
19. **Portability and Compatibility**

The application would be compatible with older android versions starting from devices with Android Oreo. If the need arises, the app can be made compatible with version older than Android Oreo (Android 8).

1. **User friendliness (Usability)**

The app would implement proper navigation to access various activities, The title of the activities would be self-explanatory and easily understandable to the user. The UI would be catchy, attractive and would use animation. The app would also contain a help section to assist the user in case of any difficulties.

1. **Security**

The passwords stored in the database would be encrypted.

In the application, the contact details of any user are only revealed when a successful verification takes place between two users by answering the questions related to the reported item.

1. **Maintainability**

The application would be well-documented.

The application code would contain consistent nomenclature for variables.

The code would be readable and comments would be used to inform the reader about why that code was implemented.

* + - 1. **System Requirements**

1. Login

* **Function** – To authenticate the user and allow access to the rest of the application
* **Description** – Authenticates the user.
* **Inputs** – Password, control id, email
* **Source** – The form with appropriate fields
* **Outputs** – Message informing the user whether they are authenticated or not.
* **Destination** – None
* **Action** – The email and password provided by the user are matched with the respective fields from the database. A message is displayed based on the result of this match.
* **Pre-condition** – User is not logged in.
* **Post-Condition** – The login state in the local storage is changed to logged in. Device id is generated and stored in local storage along with the login state. Device id, device token is stored in database after successful login.
* **Requires** – Internet Connection.

1. Registration

* **Function** – To enable new users to register themselves
* **Description** – Provides a form through which the details of the users are taken and upon successfully determining that the user fulfils the constraints, the details are stored and the user can log in.
* **Inputs** – Password, control id, email
* **Source** – The form with appropriate fields
* **Outputs** – Message informing the user whether they are registered or not.
* **Destination** – Data is stored in the database.
* **Action** – Once the user fills up the required fields as per the constraints, the system checks whether the user has already registered. If the user is registering for the first time, their details are stored in the database. Appropriate message is displayed to the user.
* **Pre-condition** – User is not logged in.
* **Post-Condition** – The account status is set to ‘registered’ if successfully registered else it remains ‘not registered’.
* **Requires** – Internet Connection.

1. Report lost item

* **Function** – To report an item that may be lost or found by the user.
* **Description** – The details of the item are provided by the user which would be stored in database.
* **Inputs** – Date of loss, time of loss, location, category, image(optional), item description, security question and answers.
* **Source** – The form with appropriate fields
* **Outputs** – Message informing the user whether the item was successfully reported or not.
* **Destination** – Data is stored in the database.
* **Action** – The item data filled by the user should match certain constraints. If they satisfy these constraints, then the data is stored. Appropriate message is displayed to the user.
* **Pre-condition** – User should be logged in.
* **Post-Condition** – ‘reported\_as’ will be set to ‘lost’ or ‘found’ based on whether the item is reported as lost or found. The date when the item was reported is also stored. This data is assigned by the system and is the system’s server time when the item is reported.
* **Requires** – Internet Connection.

1. Viewing reported items posted by user and manage it

* **Function** – To enable the user to manage the items they have reported.
* **Description** –The user can edit the item description and other details. They can also delete the reported items and also update the status of reported item. They can also just simply view the details of the item.
* **Inputs** – updated details when editing the item details. No inputs for deleting.
* **Source** – The form with appropriate fields. None for deleting.
* **Outputs** – Message informing the user whether the item was successfully updated /deleted or not. All items reported by the user.
* **Destination** – The activity responsible for displaying the items, data is updated in database.
* **Action** – When the user is editing the details of the reported item, the updated data should match certain constraints. If they satisfy the constraints, the data is updated in database.

The user can also update the status of the reported item which is updated in database.

The user can delete a reported item, which deletes the item from the database. The list of all the reported items is fetched from the database posted by user and displayed to the user with the items arranged in the descending order of their date reported.

* **Pre-condition** – User should be logged in.
* **Post-Condition** – None.
* **Requires** – Internet Connection. User’s control id or email for displaying the items reported by them.

1. Viewing all reported items

* **Function** – To display all the reported items to the user.
* **Description** – All the reported items are displayed to the user after fetching from the database.
* **Inputs** – None
* **Source** – None
* **Outputs** – All the reported items that were posted on or the previous days from the current date.
* **Destination** – Activity meant to display all reported items.
* **Action** – This system fetches all the reported items and sorts them and only displays the reported item that have been posted on the current date or the dates prior to the current date.
* **Pre-condition** – User should be logged in.
* **Post-Condition** – None
* **Requires** – Internet Connection, Server’s current date.

1. Notifying the users about important updates.

* **Function** – The system should notify the users regarding important updates.
* **Description** –The system sends notifications (push notifications) to the user when the status of their item is updated, a matching item to the lost item is found, a in-app message is received, etc.
* **Inputs** – None
* **Source** – None
* **Outputs** – The notification with the appropriate message content.
* **Destination** – The user’s mobile device
* **Action** –When the event that triggers a notification occurs, for instance when the status of the reported item changes, a in-app message is received sent by other user, a similar matching item is reported,etc the notification is sent by the server to the respective devices using the push notification service.
* **Pre-condition** – The user should be logged in.
* **Post-Condition** – None
* **Requires** – Internet Connection. Push notification service provider.

1. The system should sort and filter reported items as specified by the users.

* **Function** – To sort and filter the reported items as specified by the users.
* **Description** –Filters the reported items as per the filter options set by the user.
* **Inputs** – Filter options based on which the reported items need to be filtered.
* **Source** – Filter options selected by the users.
* **Outputs** – Filtered list of reported items.
* **Destination** – Activity(screen) meant to display the reported items.
* **Action** – If the reported items are to be filtered according to date, then the date is received from the user and the items reported ie date of loss matches the received date are displayed.

Similarly, the reported items are filtered according to ‘date\_reported’.

If ‘location’ is selected as a filter, then the location is fetched from the user and items reported with the location that matches the received location is displayed.

If ‘category’ is selected as a filter, then the reported items belonging to that category are displayed.

When multiple filters are selected, the reported items that match the filters provided are displayed.

* **Pre-condition** – User should be logged in.
* **Post-Condition** – None.
* **Requires** – Internet Connection. List of reported items.

1. The system should provide the users a way to search reported items.

* **Function** – To return a list of reported items that satisfy the search query.
* **Description** – Filter the reported items that contain the name of the item as entered by the user.
* **Inputs** – Name of item to look for.
* **Source** – A field to take input (search query) from the user.
* **Outputs** – The filtered list of reported items that contain the name of the item as entered by the user.
* **Destination** – Activity meant to display the reported items.
* **Action** – After receiving the name of reported item to look for, the system looks for the reported items that contain the name of item entered by the user. It displays all the reported items that contain the name of the item entered by the user.
* **Pre-condition** – User should be logged in.
* **Post-Condition** – None
* **Requires** – Internet Connection, list of reported items.

1. The system should provide the users a way to communicate by exchanging messages with each other within the application.

* **Function** – To send and receive in-app messages to and from other users.
* **Description** – Sending and receiving messages from other users after successful verification that either user (owner or finder) has answered the verification questions correctly.
* **Inputs** – The device tokens of the users, message body.
* **Source** – Device token from database, input field for message body.
* **Outputs** – None.
* **Destination** – Message body is stored in database.
* **Action** – The application would send the message body to the server. The server will also get the users device token from the database after determining the users which may be provided by the application. Then the server would make of these details to send the notification about the message and the message body is stored in database.
* **Pre-condition** – User should be logged in.
* **Post-Condition** – None.
* **Requires** – Internet Connection. Users should be verified after answering the verification questions. Both users should be registered.

1. The system should allow the owner and finder to place call/ contact each other via call directly from the application.

* **Function** – To call the respective user.
* **Description** – Calling the respective user without having to open the dialer app manually by clicking on a call icon present in the app.
* **Inputs** – Phone number of the user to be contacted.
* **Source** – The activity meant to display the user’s detail who is to be contacted.
* **Outputs** – None.
* **Destination** – None.
* **Action** – The application will use the default phone/call application to pace a phone call to the provided number. It starts an intent with the phone as its data.
* **Pre-condition** –
* **Post-Condition** – The user should be logged in.
* **Requires** – Users should be registered, respective users should have answered the verification questions correctly.

1. The system should provide a way to verify whether the user is the rightful owner or claims to possess the correct item.

* **Function** – To verify the users by asking them questions.
* **Description** – The system would ask the users who claims to be the rightful owner or claims to be in possession of a reported item certain questions. Based on the answers to these questions, the users are verified.
* **Inputs** – Answers to the questions.
* **Source** – The form with appropriate fields.
* **Outputs** – Message informing the user whether they are verified or otherwise.
* **Destination** – None.
* **Action** – The answers provided by the user are checked with the answers present in the database. Based on the results of this check, a message is displayed.
* **Pre-condition** – User should be logged in.
* **Post-Condition** – The ‘verification\_status’ between the two users and for a particular item is changed to ‘verified’ or ‘not verified’ accordingly.
* **Requires** – Internet Connection. Question and answers stored in the database.

1. The system should allow the user to edit their contact information

* **Function** – To edit phone number and email.
* **Description** –Update the phone number and email.
* **Inputs** – Phone number, email
* **Source** – The form with appropriate fields
* **Outputs** – Message informing the user whether the details are updated or otherwise.
* **Destination** – Updated details are stored in database.
* **Action** – The received inputs are validated to match the general format of phone number and email. If email is to be updated, then it needs to be verified, by entering the OTP sent to the email address. Then the details are sent to the database to be updated and the appropriate message is displayed informing the user whether the updating was successful or not.
* **Pre-condition** – User should be logged in.
* **Post-Condition** – None.
* **Requires** – Internet Connection.

1. The system should be able to send OTP via email for email verification.

* **Function** – Sending email containing the generated OTP.
* **Description** – The OTP is generated and sent to the email address provided by SMTP.
* **Inputs** – Email address.
* **Source** – The form with appropriate fields
* **Outputs** – None.
* **Destination** – None.
* **Action** – The 6-digit OTP is generate and sent to the provided email address via SMTP. SMTP service used can be either Firebase SMTP or Java Mail API.
* **Pre-condition** – None.
* **Post-Condition** – None.
* **Requires** – Internet Connection.

1. The system should provide a help section which allows the user to contact the admin and raise their concern as well as view FAQs during signup.

* **Function** – Contact the admin and view FAQs.
* **Description** – Display the FAQ’s and open the email application to contact the admin.
* **Inputs** – None.
* **Source** – None.
* **Outputs** – List of FAQs.
* **Destination** – The activity concerned with displaying the FAQs.
* **Action** – The FAQs are fetched from the database and displayed to the users. When trying to contact the admin, the default email application is opened with the ‘to:’ field being set to the admin’s email address.
* **Pre-condition** – None.
* **Post-Condition** – None.
* **Requires** – Internet Connection.

1. The system should provide the user a way to delete their profile.

* **Function** – To delete the user’s profile.
* **Description** – Delete the user’s profile i.e details from the database permanently.
* **Inputs** – Control id for students and email for faculty.
* **Source** – Database.
* **Outputs** – Message informing the user whether their profile is deleted or not.
* **Destination** – Database.
* **Action** – The user is deleted from the database using the control id or the email provided.
* **Pre-condition** – User is logged in.
* **Post-Condition** – User is logged out.
* **Requires** – Internet Connection.

1. The system should allow the users to log out

* **Function** – To log out the user.
* **Description** – The user logs out of the application.
* **Inputs** – None.
* **Source** – None.
* **Outputs** – Message informing the user whether they are logged out or not.
* **Destination** – None
* **Action** – The user is directed to the login page.

The device token is deleted from the database.

Login state is changed to false.

* **Pre-condition** – User is logged in.
* **Post-Condition** – User is logged out.

The login state is changed to ‘false’ in local storage.

The stored username is set to ‘null’.

Device details are deleted from database.

* **Requires** – Internet Connection.

1. The system should provide the admin a way to manage reported items.

* **Function** – To manage the reported items.
* **Description** – Delete or update the progress of the reported item.
* **Inputs** – The reported item, updated status of that item.
* **Source** – The reported item is fetched from the database, the updated status is received from the application (activity).
* **Outputs** – Message informing the admin whether the deleting or updating was successful or otherwise.
* **Destination** – Data is deleted or updated in the database.
* **Action** – When the reported item is requested to be deleted, the details of the item are deleted from the database and the appropriate message is displayed.

When the status of the reported item is updated, the changes are reflected in the database and the appropriate message is displayed.

* **Pre-condition** – The admin is logged in.
* **Post-Condition** – None.
* **Requires** – Internet Connection.

1. The system should provide the admin a way to manage the users.

* **Function** – To manage the users.
* **Description** – The admin adds a news user, suspends, deletes a user.
* **Inputs** – User’s control id or email depending whether they are a student or faculty respectively, user’s details.
* **Source** – The control id or email is received from the application (activity). The details of the new user to be added is received from the form with the appropriate fields.
* **Outputs** – Message informing the admin the result of the performed operation.
* **Destination** – Database.
* **Action** – When the admin adds the details of the new user, the details are checked with the database. If the user being added is a faculty, then the email given by the college is checked.

If the user being added is a student, then the user’s control id is checked.

Based on the result, the appropriate message is displayed.

When the user is being suspended, the account status of the user is updated to ‘suspended’. The appropriate message is displayed to the admin based on the result of this operation.

When a certain user is deleted, the details of the user are deleted from the database and appropriate message is displayed based on the result of this operation.

* **Pre-condition** – The admin should be logged in.
* Post-Condition – None.
* **Requires** – Internet Connection.

1. The admin should be able to unsuspend the suspended users.

* **Function** – To unsuspend the suspended users.
* **Description** – The admin unsuspends the suspended users.
* **Inputs** – User’s control id or email depending whether they are a student or faculty respectively.
* **Source** – The application provides the control id or email of the user depending whether they are a student or faculty respectively.
* **Outputs** – Message informing the admin the result of the performed operation.
* **Destination** – None.
* **Action** – The admin unsuspends the user from the application and the account status of the user changes to ‘registered’. Email of the same is also sent to the user.
* **Pre-condition** – The admin should be logged in.
* Post-Condition –Account status changes from ‘suspended’ too ‘registered’.
* **Requires** – Internet Connection. List of suspended users.

**Chapter 4**

**System Design**

* 1. **Diagram 1**
  2. **Diagram 2**
  3. **Entity Relationship Diagram**

An Entity-Relationship Diagram is a visual representation used to model the structure of a database. It illustrates the relationships between different entities, which represent real-world objects, concepts, or information within the database system. ER diagrams consist of entities, attributes which are the properties of those entities, relationships which are the connections between those entities, and cardinality among those entities. ER diagrams are essential tools for database design, helping developers and stakeholders understand the logical organization of data, identifying key relationships, and ensuring the proper implementation of a database schema. By providing a clear and intuitive overview, ER diagrams play a crucial role in improving communication and ensuring the accuracy of database systems in various domains, such as business, software development, and data management. ER model becomes an abstract data model that defines a data or information structure which can be implemented typically in a relational database.

**Diagram Notations:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Symbol** | **Description** |
| Rectangle |  | Entity set |
| Double Rectangle |  | Weak Entity set |
| Diamond |  | Relationship set |
| Double Diamond |  | Identifying relationship set for weak entity set |
| Eclipse |  | Attribute |
| Double Eclipse |  | Multivalued attribute |
| Dotted Eclipse |  | Derived attribute |
| Primary key | A | Primary key |
| Discriminator | A | Discriminating attribute of weak entity set |
| Many-to-many |  | Many-to-many relationship |
| One-to-one |  | One-to-one relationship |
| Many-to-one |  | Many-to-one relationship |
| Double line |  | Total participation of entity set in a relationship |
| Line |  | Links attribute to entity set or represents Partial participation of entity set in a relationship. |
| Mapping cardinality | 1..h | Cardinality limits |
| Role indicator | Role-name | Role indicator |
| ISA | ISAA | ISA (specialization or generalization) |
| Total generalization | ISAA | Total generalization |
| Disjoint generalization | disjoint  ISAA | Disjoint generalization |

Table 4.1 Entity Relationship Diagram Symbols

* + 1. **Entity Sets**

1. **Users**

It represents all the users (students and faculty) that have registered.

Attributes:

1. control\_faculty\_id – single valued, simple attribute

* It is the control id of the student which is assigned to the student by the college or a random id generated for the faculty.

1. user\_email - single valued, simple attribute

* Email of the user. It is optional for students, but mandatory for faculty.

1. user\_phone - single valued, simple attribute

* Phone of the user.

1. user\_fullname - single valued, simple attribute

* It is the full name of the user.

1. user\_password - single valued, simple attribute

* It is the password of the user used to login to the application.

1. user\_account\_status - single valued, simple attribute

* It represents whether the account is suspended or active.

1. user\_type - single valued, simple attribute

* It is used to distinguish between the student users and the faculty users. It can take the following values: ‘student’, ‘faculty’.

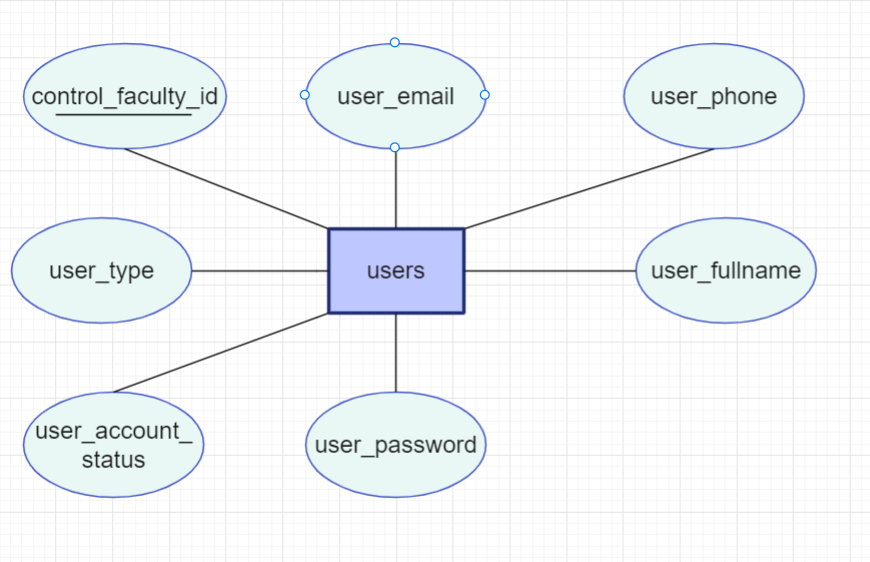
****

Figure 4. Users entity set

1. **Admin\_users**

It represents all the admins of the application.

Attributes:

1. admin\_id – single valued, simple

* It is an id given to the admin to uniquely identify each admin.

1. admin\_email- single valued, simple attribute

* Email of the admin.

1. admin\_fullname - single valued, simple attribute

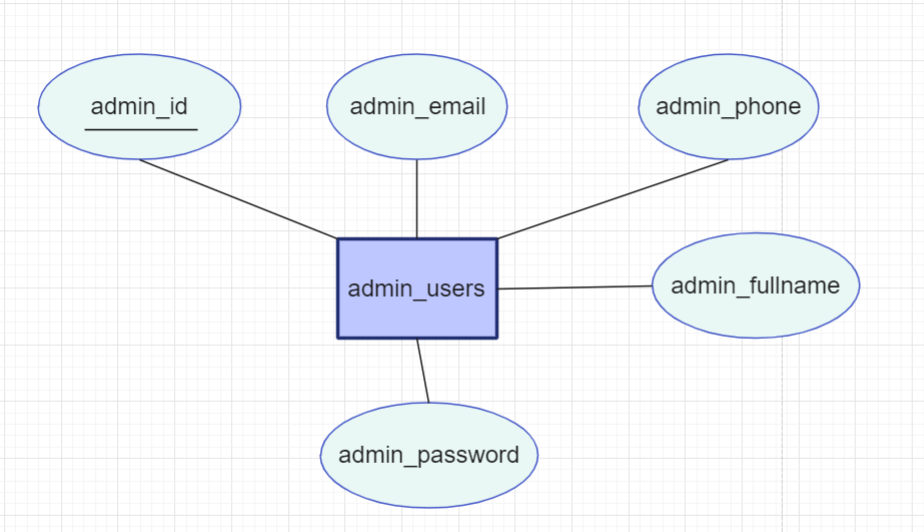
* Full name of the admin.

1. admin\_password - single valued, simple attribute

* Password of the admin’s account used to login to the application.

1. admin\_phone - single valued, simple attribute

* Phone of the admin.

****

*Figure 4. admin\_users entity set*

1. **User\_device\_details**

It stores the user’s device information such as the device id and the device token in order to send push notifications.

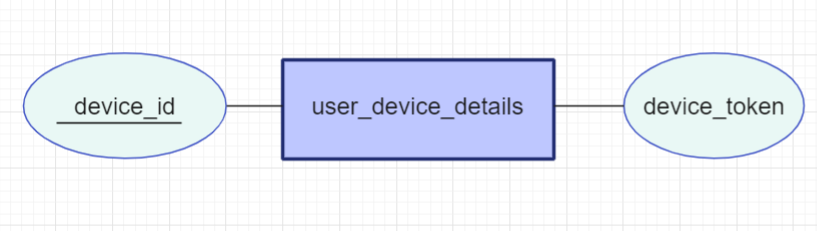
Attributes:

1. device\_id – single valued, simple

* It uniquely identifies each device which has an account logged on to it.

1. device\_token - single valued, simple

* It is used to store tokens by Firebase when the user logs in. This token is used by Firebase to send the push notification.

****

*Figure 4. user\_device\_details entity set*

1. **FAQ**

It stores the FAQ’s to be displayed to the user. FAQ stands for frequently asked questions.

Attributes:

1. faq\_id – single valued, simple

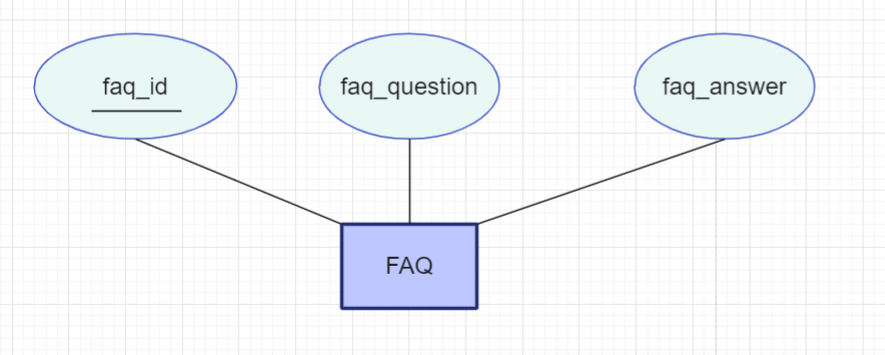
* It uniquely identifies each FAQ.

1. faq\_question - single valued, simple

* A FAQ question.

1. faq\_answer - single valued, simple

* The answer to the FAQ.

****

*Figure 4. FAQ entity set*

1. **Reported\_items**

It stores information related to the items that are reported by the users. These items can either be the ones that are lost or the ones that have been found by others.

Attributes:

1. r\_id- single valued, simple

* It is an id assigned to each reported item that distinguishes it from the rest.

1. r\_name - single valued, simple

* The name of the item.

1. r\_location- single valued, simple

* The location where the item was found or misplaced.

1. r\_time\_of\_loss - single valued, simple

* The time when the item was found or lost.

1. r\_date\_of\_loss - single valued, simple

* The date when the item was found or lost.

1. r\_progress - single valued, simple

* Tells the progress of the item such as an item is still, or was it found, or is it with the lost and found department.

1. r\_reported\_as - single valued, simple

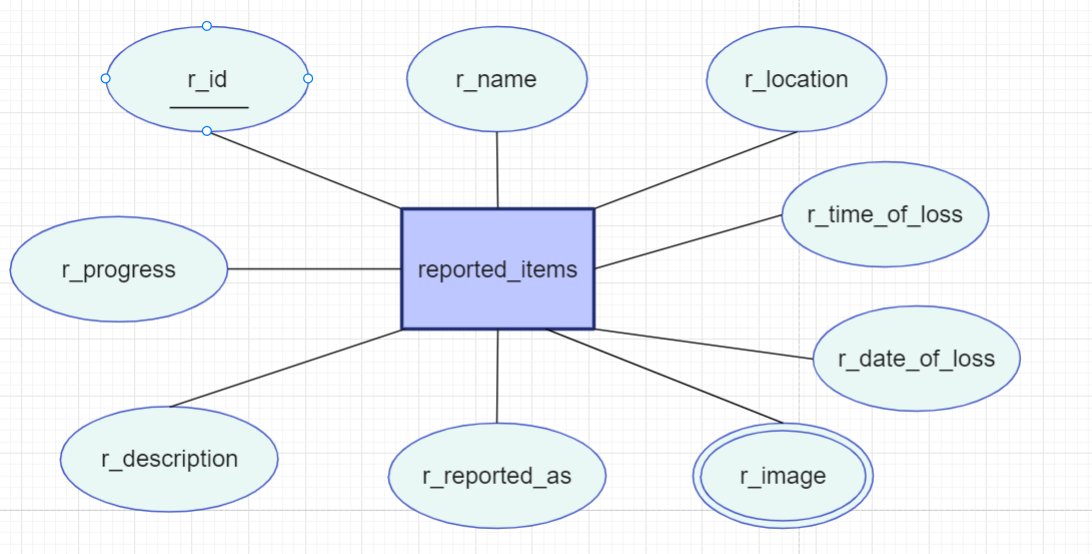
* signifies whether the item is reported as lost or found.

1. r\_description- single valued, simple

* The items description.

1. r\_image – multivalued, simple

* The images of the lost item.

****

*Figure 4. reported\_items entity set*

1. **Security\_QA**

It is a weak entity set of the ‘reported\_items’ entity set that stores the security questions and answers for that reported item.

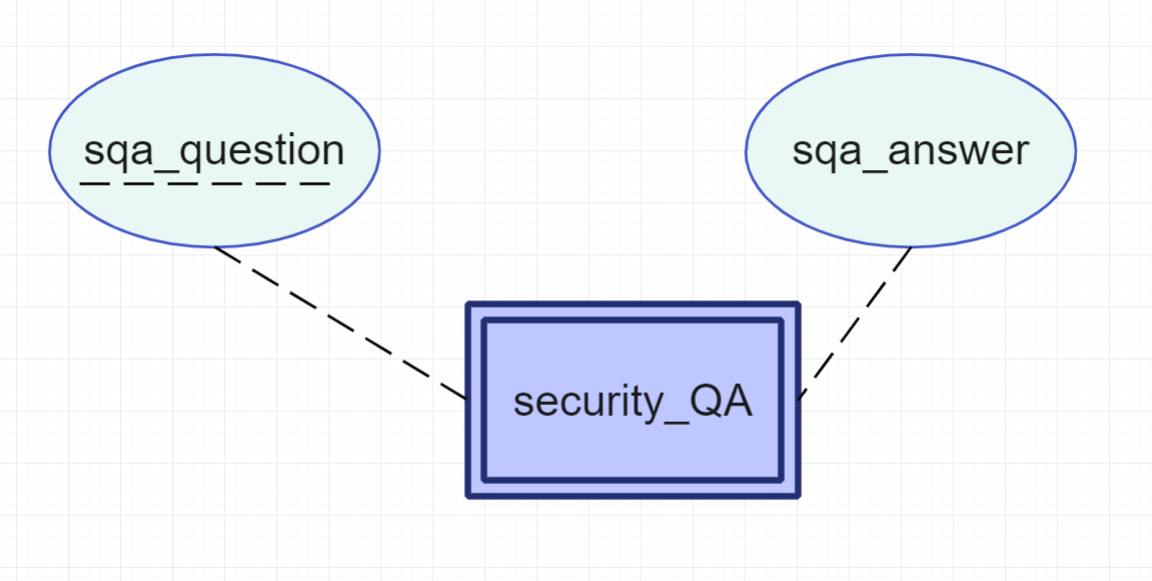
Attributes:

1. sqa\_question - single valued, simple

* It is the discriminator and represents the question asked for verification of a reported item.

1. sqa\_answer - single valued, simple

* It is the correct answer to the respective question.

****

*Figure 4. security\_QA entity set*

1. **Category**

It represents the categories to which a reported item may fall under.

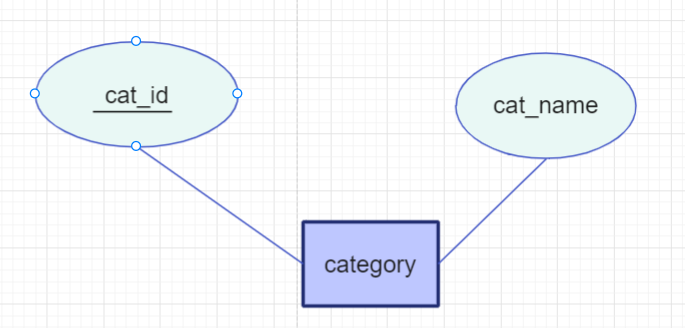
Attributes:

1. cat\_id - single valued, simple

* uniquely identifies each category.

1. cat\_name - single valued, simple

* name of the category.

****

*Figure 4. category entity set*

1. **Messages**

It stores the messages sent by the users and admins.

Attributes:

1. m\_id - single valued, simple

* uniquely identifies each message.

1. m\_content - single valued, simple

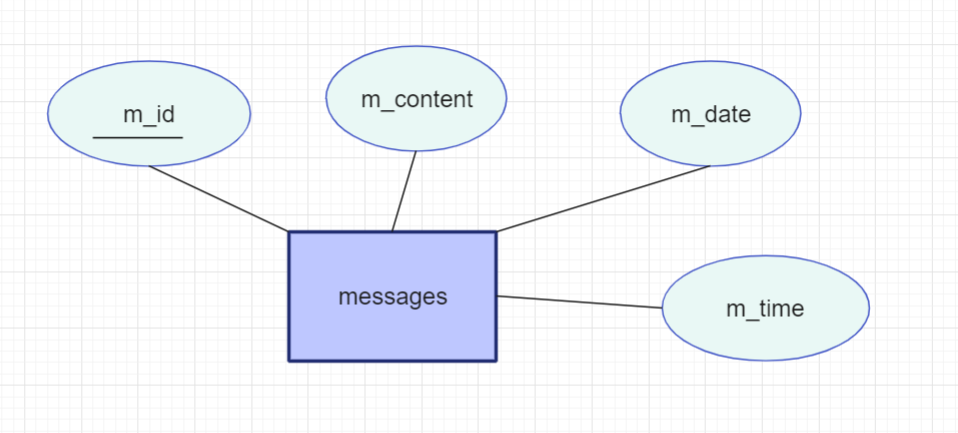
* the content or body of the message.

1. m\_date - single valued, simple

* The date when the message was sent.

1. m\_time - single valued, simple

* The time when the message was sent.

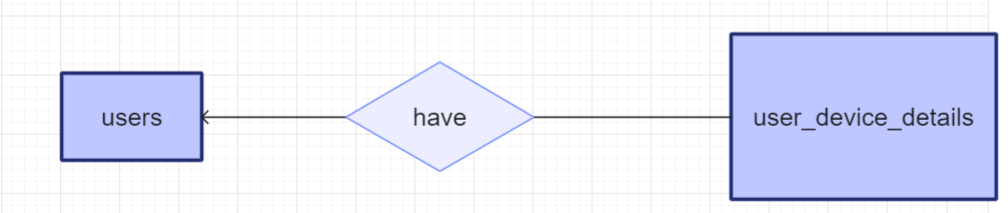
****

*Figure 4. messages entity set*

* + 1. **Relationship Sets**

1. users -> have -> user\_device\_details

* It determines which user’s device details are stored.
* One to many
* Partial participation throughout.

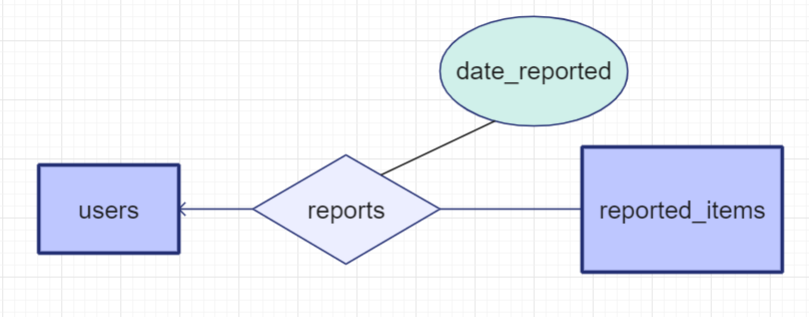


*Figure 4. have relationship set*

1. users -> reports -> reported\_items

* It specifies which user has reported a specific item.
* One to many
* Partial participation throughout.
* Descriptive Attributes:

1. date\_reported – single valued, simple
   * + - * It is the date when the item was reported.

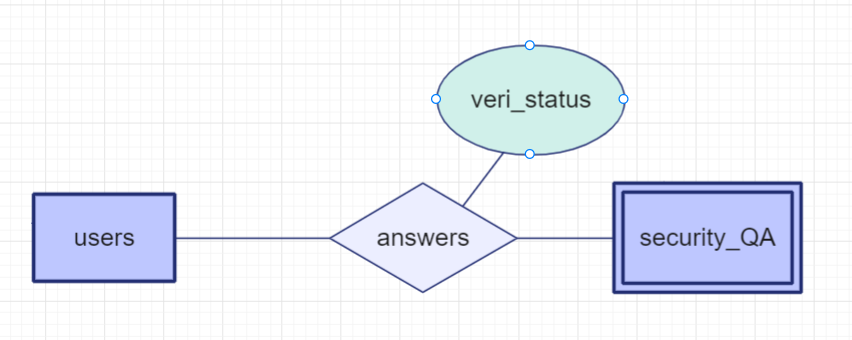
****

*Figure 4. reports relationship set*

1. users -> answers -> security\_QA

* It represents that users answer security questions for verification purposes.
* Many to many
* Partial participation from users
* Total participation from security\_QA
* Descriptive Attributes:

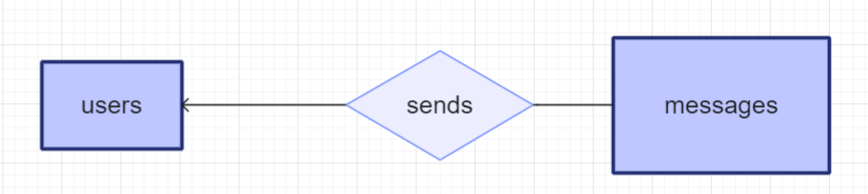
1. veri\_status – single valued, simple
   * + - * It represents the verification status when the users answer the questions.

****

*Figure 4. answers relationship set*

1. users -> sends -> messages

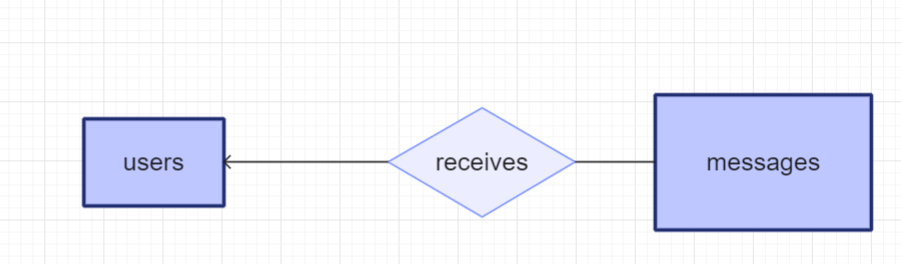
* It signifies which user sent which particular message.
* One to many
* Partial participation throughout.

****

*Figure 4. sends relationship set*

1. users -> receives -> messages

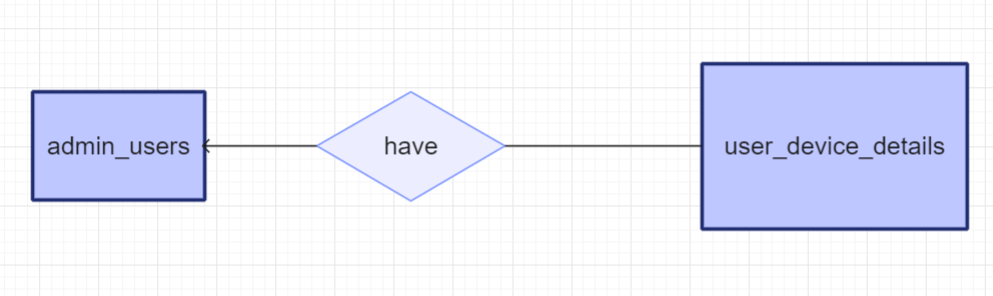
* It signifies to which user was intended to receive the message.
* One to many
* Partial participation throughout.

****

*Figure 4. receives relationship set*

1. admin\_users -> have -> user\_device\_details

* It determines which user’s device details are stored.
* One to many
* Partial participation throughout.

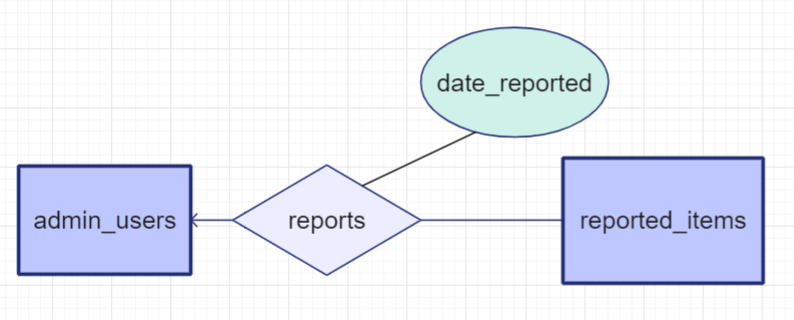
****

*Figure 4. have relationship set*

1. admin\_users -> reportss -> reported\_items

* It specifies which admin has reported a specific item.
* One to many
* Partial participation throughout.
* Descriptive Attributes:

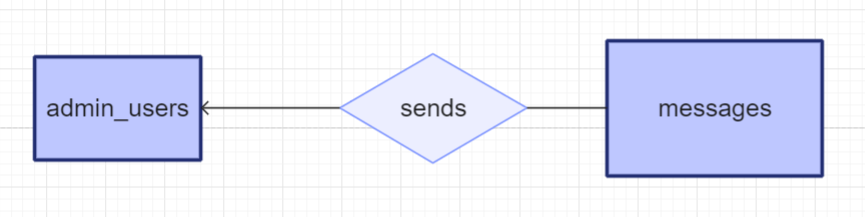
1. date\_reported – single valued, simple
   * + - * It is the date when the item was reported.

****

*Figure 4. reports relationship set*

1. admin\_users -> sends -> messages

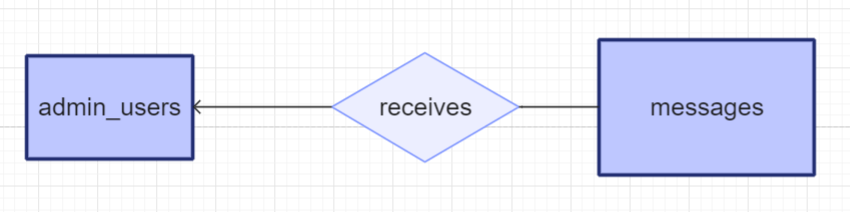
* It signifies which admin sent which particular message.
* One to many
* Partial participation throughout.

****

*Figure 4. sends relationship set*

1. admin\_users -> receives -> messages

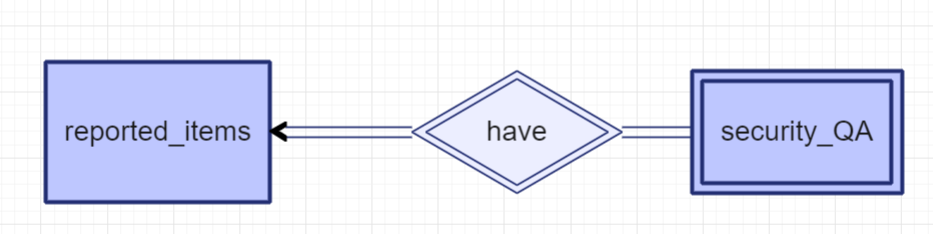
* It signifies to which user was intended to receive the message.
* One to many
* Partial participation throughout.

****

*Figure 4. receives relationship set*

1. reported\_items -> have -> security\_QA

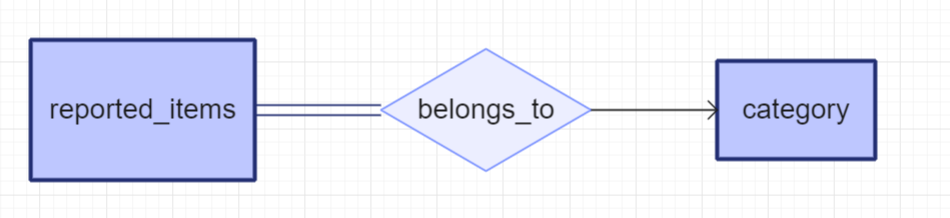
* It is an identifying relationship between the strong entity ‘reported\_item’ and the weak entity set ‘security\_QA’ and this represents that the reported item has security question and answers.
* One to many
* Total Participation throughout.

****

*Figure 4. have identifying-relationship set*

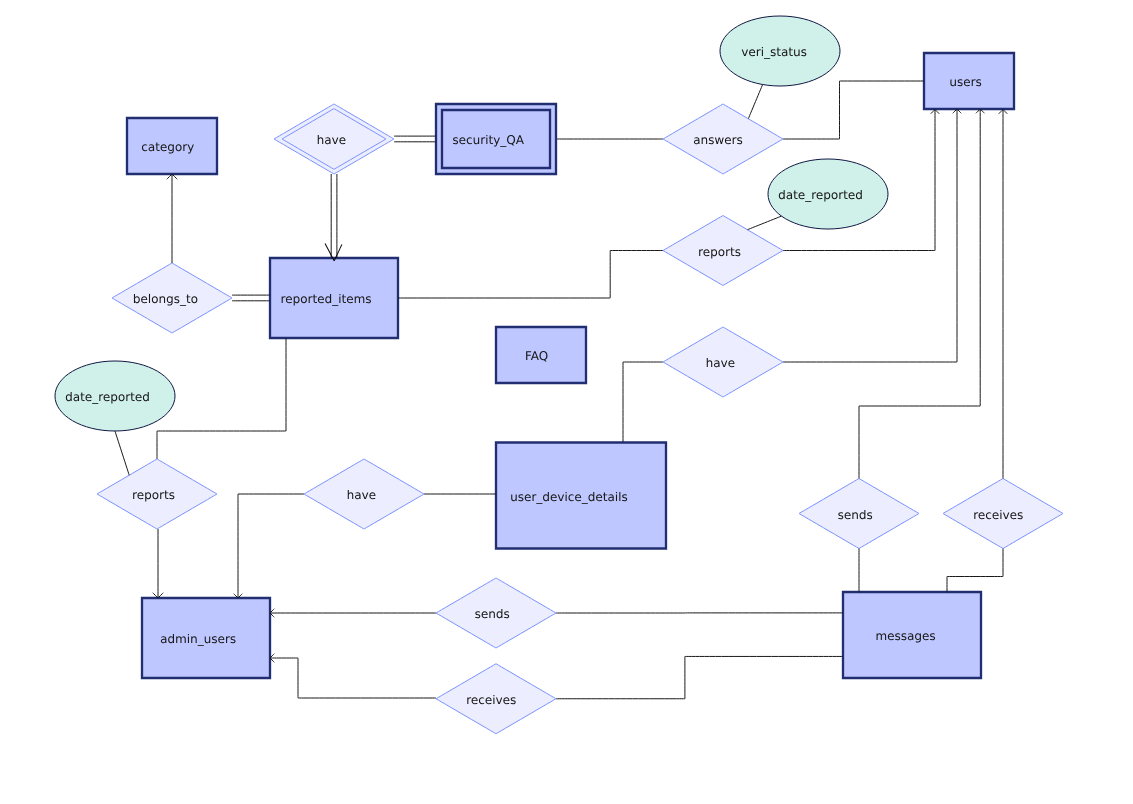
1. reported\_item -> belongs\_to -> category

* It depicts that a reported item belongs to some particular type of category.
* Many to one
* Total participation from reported\_items.
* Partial participation from category.

****

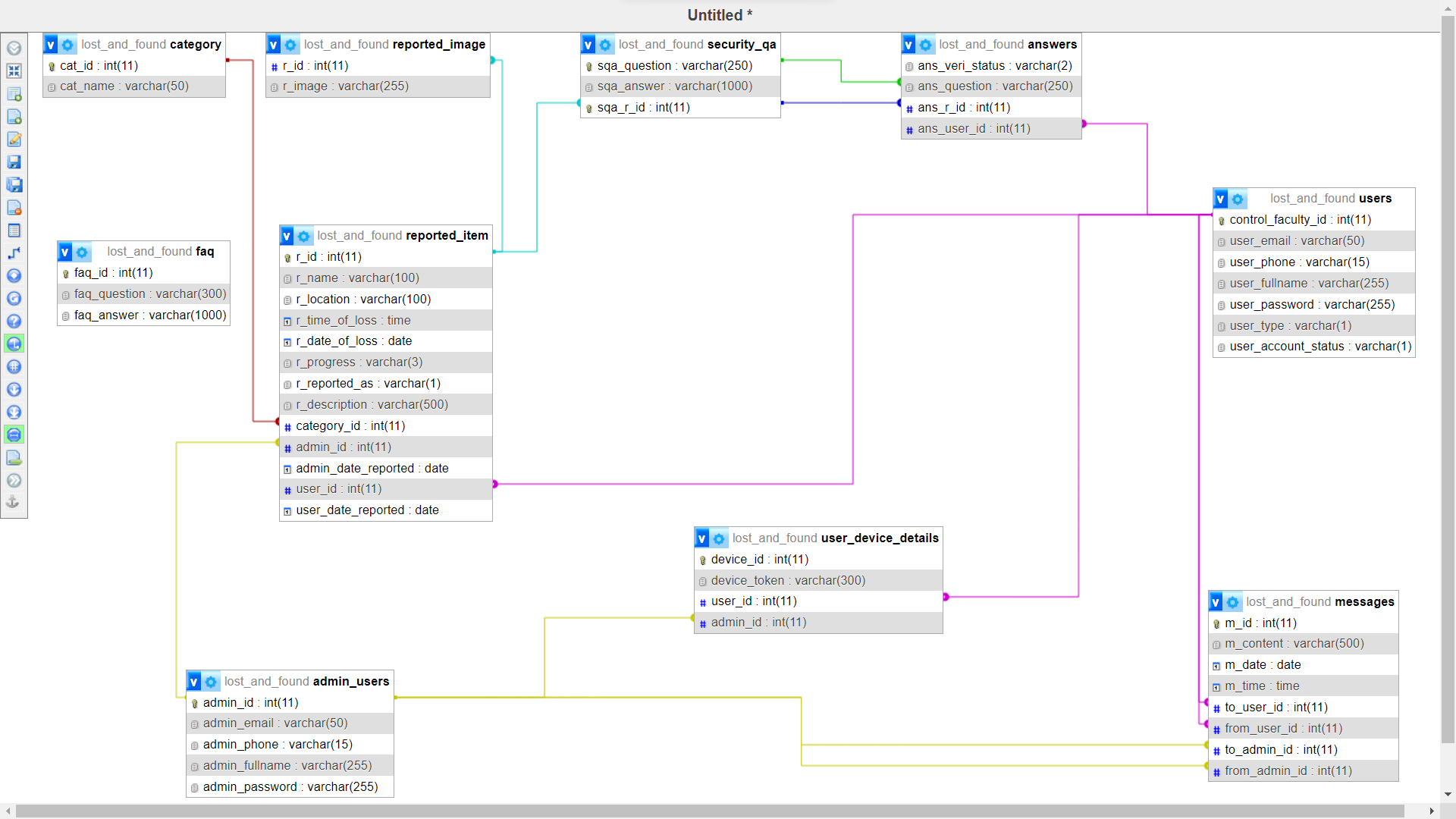
*Figure 4. belongs\_to relationship set*

* + 1. **ER diagram**



*Figure 4. ER Diagram*

* + 1. **Schema diagram**



*Figure 4. Schema Diagram*