REPUBLIC OF CAMEROON

Peace-Work-Fatherland

University of Buea

REPUBLIC DU CAMEROUN

Paix-Travail-Patrie

Universite de Buea

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

CEF440: Internet Programming and Mobile Programming

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

Academic year 2023/2024

SUPERVISED BY

Dr Nkemeni Valery

PRESENTED BY MEMBERS OF GROUP 9

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

ACKNOWLEDMENTS

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

TABLE OF CONTENT

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

INTRODUCTION

The design process follows a structured approach, including various diagrams to model the system comprehensively:

1. **Context Diagram:** Establishes the system foundation, detaling data flow and interactions between the user, the application, and backend services.
2. **Use Case Diagram:** Based on the context , it details the functionalities and interactions between the user and the application.
3. **Sequence Diagram:** Breaks down specific use cases into sequences of actions and messages between components.
4. **Class Diagram:** Defines the system's internal building blocks, their attributes, and methods.
5. **Deployment Diagram:** Illustrates the physical layout of the system components and their interactions. (can be created later).

These models provide a clear and concise blueprint for developing a reliable, efficient, and user-friendly application that integrates seamlessly with various external services to deliver real-time information to drivers.

1. CONTEXT DIAGRAM

What are Context Diagrams?
Context Diagrams are high-level visual representations that show the interactions between a system being developed and its external entities, such as users, other systems, or processes. They provide a big-picture view of how the system fits into its environment without diving into the internal details of the system itself. Typically, they consist of a central system surrounded by external entities, with arrows representing data flow or interactions between them. They're useful for understanding system boundaries and dependencies.

Below are the benefits of using Context Diagram:

- **Clarity**: Context diagrams offer a straightforward and clear snapshot of how the system interacts with its surroundings, aiding stakeholders in grasping its scope and context with ease.
- **Communication**: Serving as effective communication aids, these diagrams streamline discussions and decision-making among stakeholders, fostering collaboration between business users, developers, and project managers.

- **Risk Identification**: Context diagrams help pinpoint potential risks stemming from the system's interactions with external elements, empowering stakeholders to identify, assess, and mitigate risks effectively.
- **Scope Definition:** By delineating the system's boundaries and external entities, context diagrams assist in defining the project's scope, ensuring that efforts are focused on pertinent components and processes.
- **Dependency Management**: Understanding the system's relationships with external entities facilitates the management of dependencies, empowering stakeholders to anticipate and prepare for potential impacts of external changes.

Components of Context Diagrams

1. System/Product:

This is the primary focus of the diagram, representing the system being analyzed or designed.

2. External Entities

These are entities outside the system boundary that interact with the system. They could be users, other systems, or processes that provide input to or receive output from the system

3. Data Flow

These are arrows representing the flow of data or information between the system and external entities. They illustrate the exchange of data between the system and its environment.

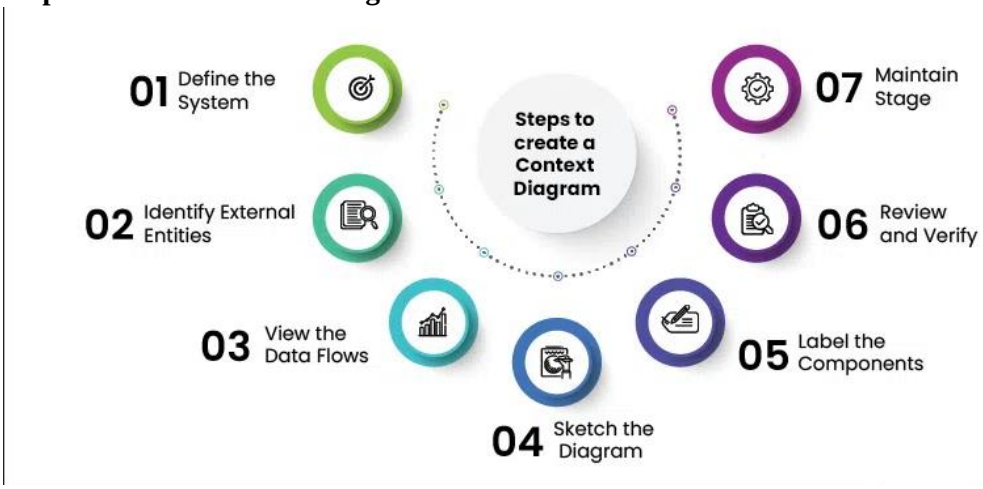
- ***Unidirectional Data Flow***

Unidirectional data flow indicates that data moves in only one direction between two components. For example, data flows from an external entity to a process, indicated by a single headed arrow showing the direction of data flow

- ***Bidirectional Data Flow***

Bidirectional data flow indicates that data can move in both directions between two components. For example, data flows between a process and an external entity can be bidirectional, allowing the process to send data to the external entity and receive data back. Represented by a double-headed arrow indicating that data can move in both directions.

Steps to create a Context Diagram



Step1: Define the System: Start by clearly defining the system you're analyzing or designing, outlining its boundaries and objectives.

Step2: Identify External Entities: Recognize all external entities that interact with the system, including users, other systems, or processes that exchange data with it.

Step3: View the Data Flows: Determine how data flows between the system and external entities, specifying the types of data and their direction (input or output).

Step4: Sketch the Diagram: Utilize standardized symbols and notation to create the context diagram. Position the system at the center and surround it with external entities, using arrows to depict data flow.

Step5: Label the Components: Ensure each element of the diagram is clearly labeled, including the system, external entities, and data flows, providing concise descriptions for clarity.
Step6: Review and Verify: Collaborate with stakeholders to review the context diagram, confirming its accuracy and completeness in representing the system's interactions.
Step7: Maintain Stage: Regularly update the diagram to reflect changes in the system or its external environment, ensuring it remains an accurate depiction of the system's context.

Let's understand Context Diagram for the archival and retrieval of missing object using an image matching algorithm

i. **Database**: A system or a collection of data related to the object, users and other stakeholders of the system, which can include information such as
   - User names, ID and contact,
   - Objects id and location etc.

ii. **User's info**: Requesting or providing information related to the user such as users, such as name, ID and contacts etc.…

iii. **Administrators**: People employed by an organization, often having a specific role or responsibility.

iv. **System** (A mobile application for the archival and retrieval of missing object using an image matching algorithm)

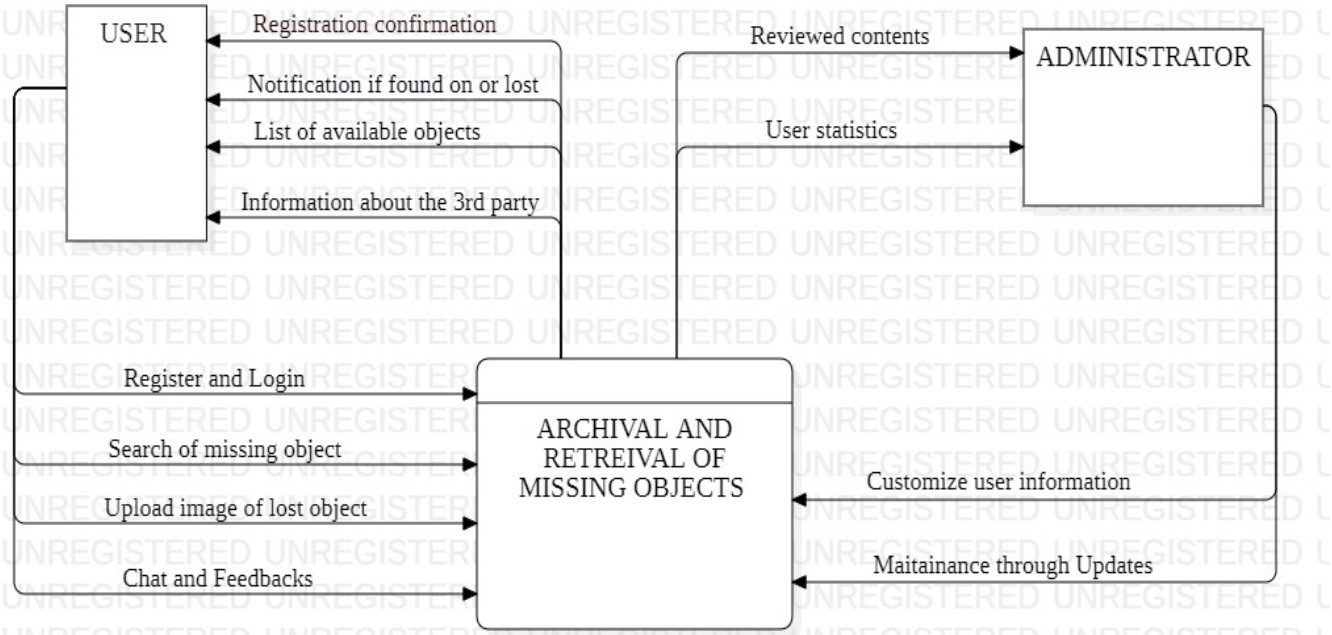
From the above information, a summary of the system is shown bellow

System (A mobile application for the archival and retrieval of missing object using an image matching algorithm)
External entities: Administrators and users of the mobile application

Processes: they would include;
- search for a missing object,
- account register and login,
- image uploading,
- Downloads list of available objects
- Chats and feedbacks
- updates

Data stores: user database

VISUAL REPRESENTATION:



EXPLANATION OF THE DIAGRAM:

2. USE CASE DIAGRAM

Details functionalities based on the context. Focuses on user actions (navigation, report hazards, etc.). Shows interactions between users and the app functionalities.This use case diagram effectively illustrates the interactions between the actors and the system, highlighting the primary functionalities and their

dependencies.

ACTORS AND USECASES

These are the various users of the system. We have 3 major in our system, who are

1) Primary actors, Users

Primary actors are stakeholders who actually uses the features of the system for their own personal need. They will be able to perform actions such as

- ✓ Login: this is done but authentication (entering the correct login criterials) and it leads to authorization (granting permission to the right people, permitting them to view their pages).
- ✓ Upload Profile: user have the possibility to set up their profile the change it a number of times they want.
- ✓ Report Lost Object: this is done by uploading the image of the lost item, attached to it some description. These two info (image and description are being sent to the matching algorithm for further investigation).
- ✓ Report Found Object: in case the user finds an object, he will also send the image of the object, plus some description. Then, he will upload some of his personal information.

- ✓ View Achieved Object list: once the user opens the app, he will automatically see the list of found and missing object.
- ✓ Send Feedback: this feature permits the user to send his critics into the app, that will help the technical team to ameliorate the app based on the preferences of our users.

2) Secondary actors, Administrators

These are actors that provide us with the app, maintain and update the app for it smooth move. Our secondary actors in this app are the administrators.
They will have the following features

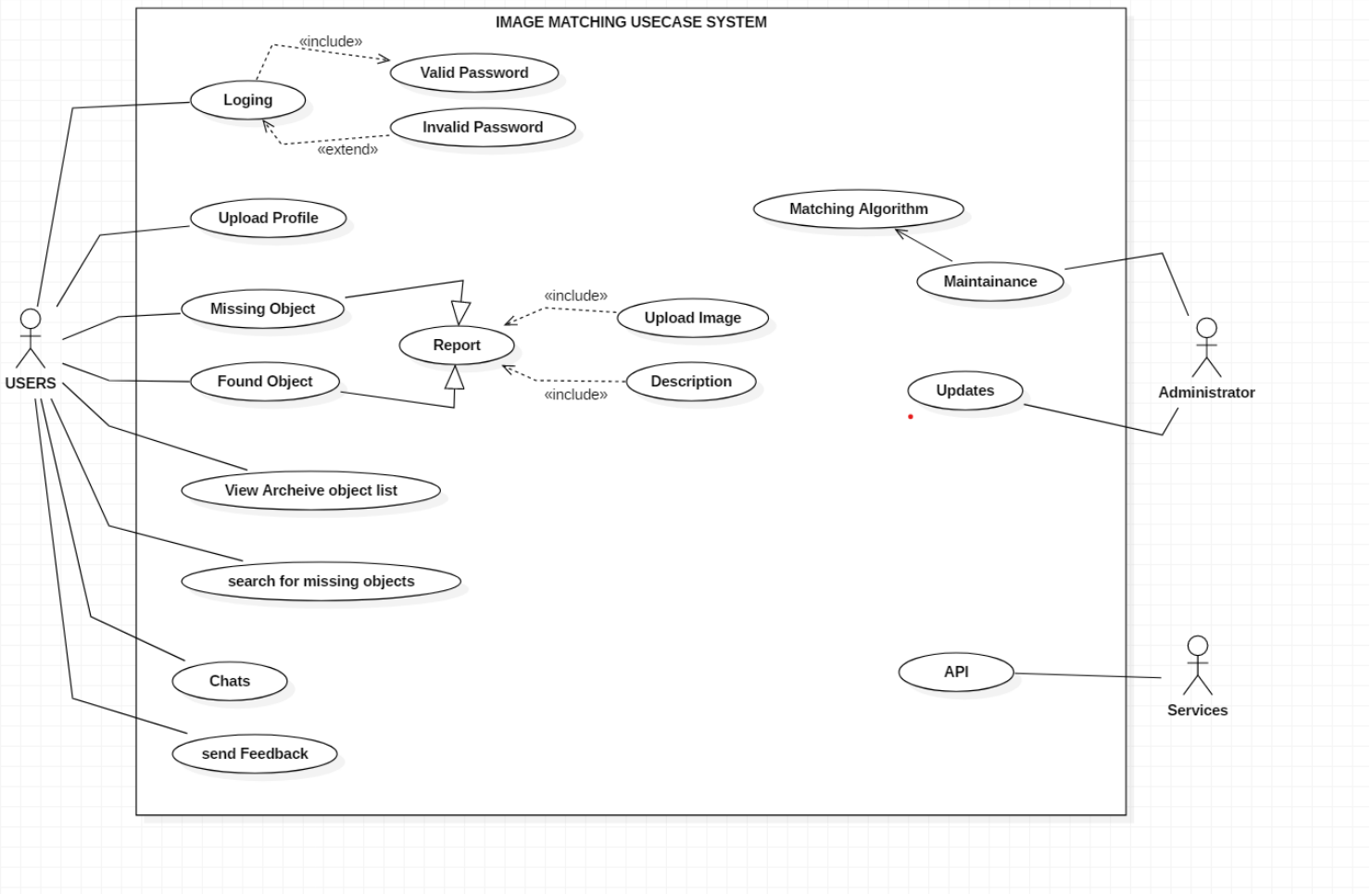
- ✓ Maintenance: the administrators will provide maintenance to the system every after a given period of time(3months). They will also ensure that the various components in the system such as the image matching algorithm function as expected.
- ✓ Update: the admin will constantly update the system and the users about any change made.

3) Tertiary Actors

These actors are not directly using the app, but they instead provide the admin with services that can be used within the app to ease their functioning.

- ✓ API: such as the image matching algorithm, email service e.t.c
- ✓ Phone Features: for example the user will be using his phone's camera to upload images into the app.

DIAGRRAMATICAL REPRESENTATION



3. SEQUENCE DIAGRAM

Dives into specific actions within use cases. **Zooms** in on **a specific use case** (e.g., navigation). llustrates the sequence of messages exchanged between app components to complete a usecase.

PARTICIPANTS:

1. Users: the actor interacting with the system.
2. System: The mobile application itself.
3. Server: the database into which all the information will be store
4. Matching algorithm service: service which help the system doing the matching.
5. Email: to send pop up messages to the user.
6. Camera: that help scanning object and uploading it to the system

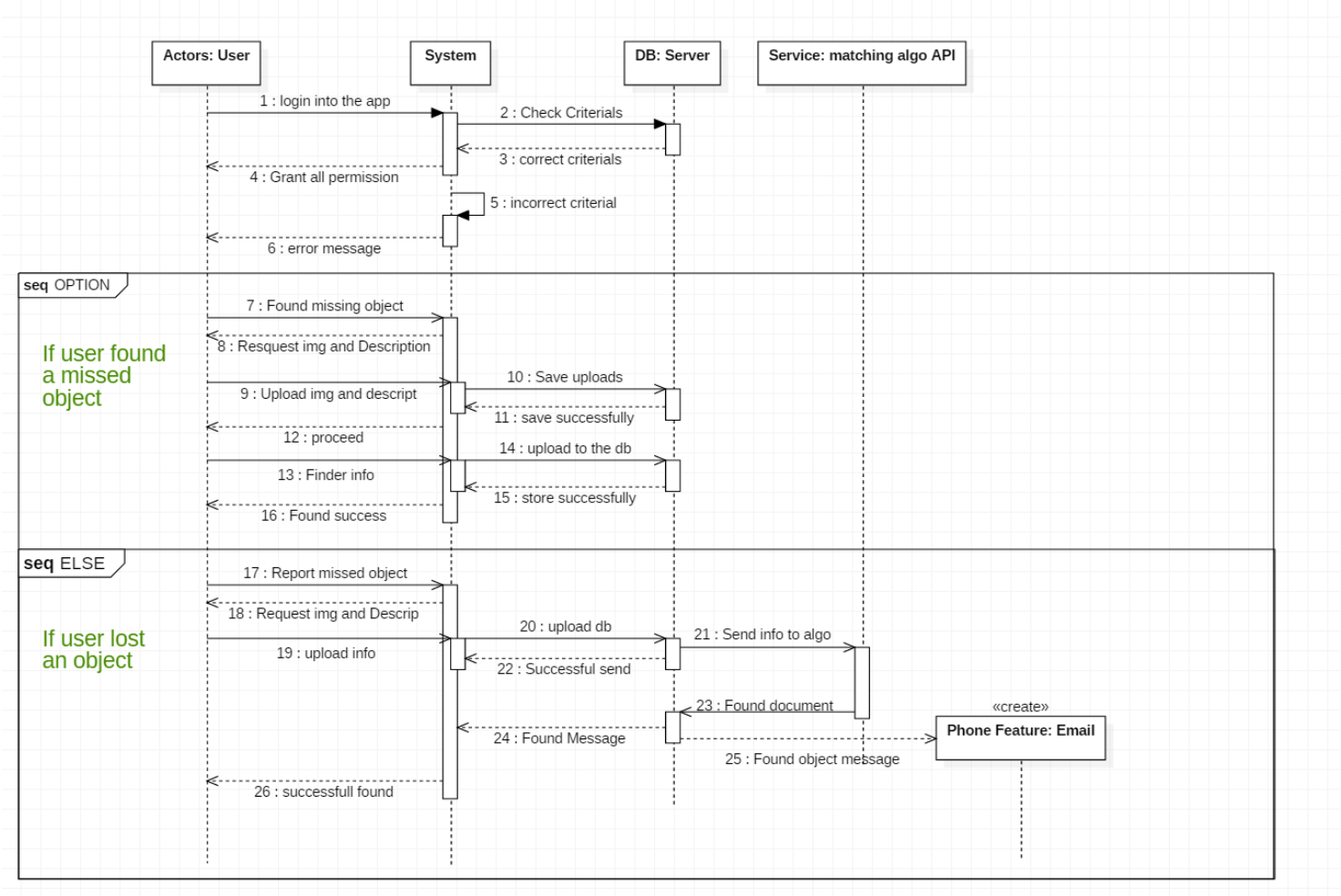
MESSAGE FLOW:

1. The user sends his login criteria to login into the app.
2. The info is sent to the database through the system's interface
3. The server check into its database and depending on weather the criterial are correct or not, it grants permission
4. Then the user chooses whether he has found or missed an object.
5. **If found**, the system will request for the object's image and description
6. The user will then provide the image and description, and it will be stored into the database. The serve will then send a "successful save message" to the user through the system user interface.
7. The system will now request for the finder personal information such as address, phone number, email and name. As the user provide the system with this information, the system will then forward it to the database.
8. Else, if lost, the system will request for the image and the object's description, then it will forward it to the server.
9. The server will then send the info to the database and the matching algorithm.
10. The image matching algorithm will then search for the image.

11. Once the algorithm successfully found the image, it will update the server, then the server will send the finder info to both the system and the user's email through the **feature email** provided but the **user's phone.**

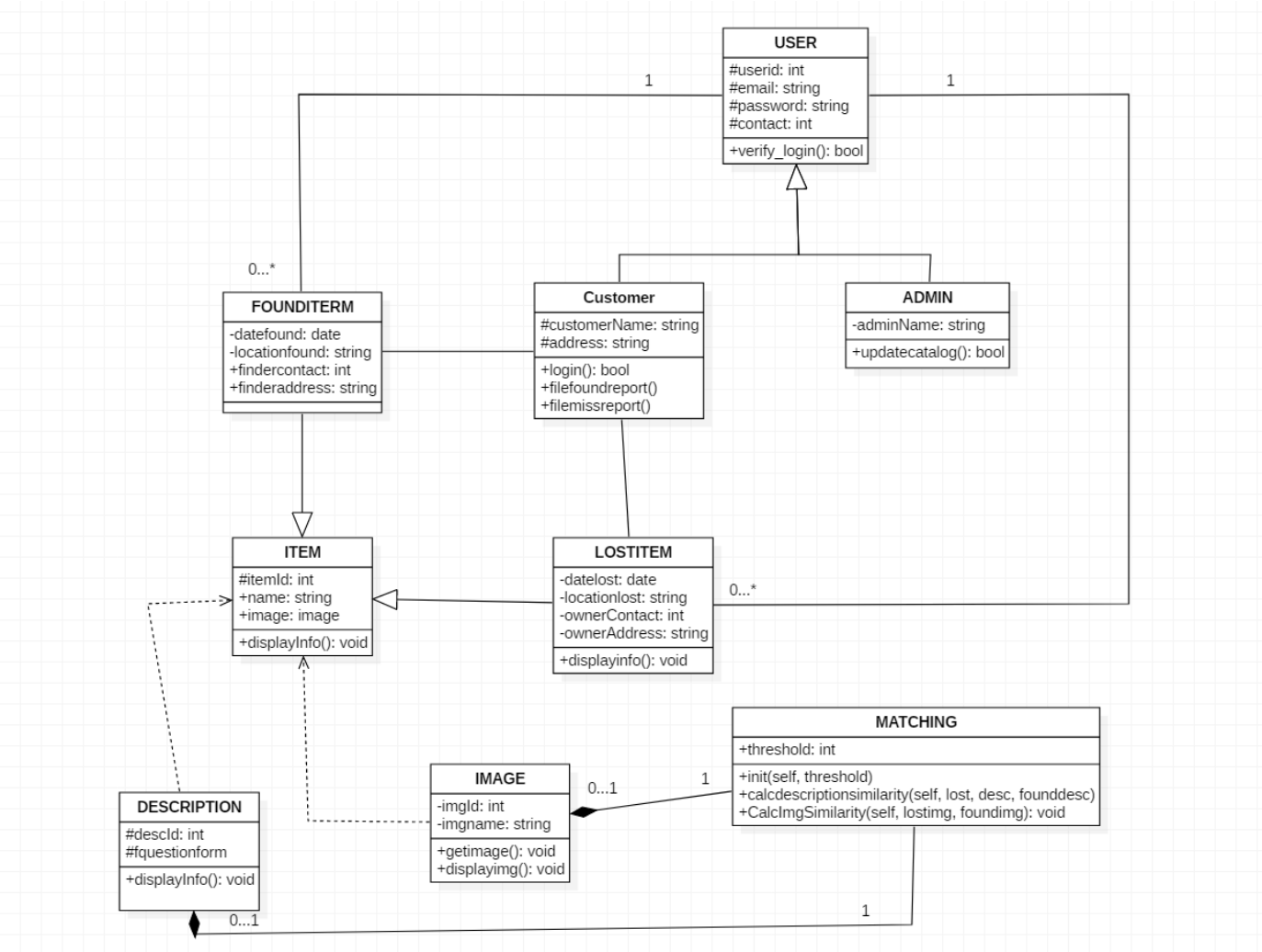
12. The system will also send a notification message to the user's interface.

VISUAL REPRESENTATION



4. Class DIAGRAM

DIAGRAMATICAL REPRESENTATION



CLASSES AND THEIR ATTRIBUTES AND METHODS WITH EXPLANATION:

1. USER:

Attributes:

- Userid: int, email: string, password: string, contact: int

- **Methods:**

- verifylogin(): string

2. **Customer Class**:

- **Attributes:**

  - customername: string

  - address: string

- **Methods:**

  - Filefoundreport(): void

  - Filemissingreport(): void

---

3. **Admin Class**:

- **Attributes:**

  - adminname: string

  - Inherits attributes from User class.

- **Methods:**

  - updatedata(): bool

---

4. **LostItem Class:**

- **Attributes:**

  - dateLost: Date

  - locationlost: string

  - owneraddress: string

  - ownercontact: string

- **Methods:**

  - displayinfo(): void

5. **FoundItem Class:**

FoundItem Class:

- **Attributes:**

  - dateLost: Date

  - locationlost: string

  - owneraddress: string

  - ownercontact: string

- **Methods:**

  - displayinfo(): void

---

6. **Description Class**:

- **Attributes:**

  - id: int

  - itemName: string

  - name: string

  - image: Image

- **Methods:**

  - displayinfo(): void

---

7. **Image Class:**

- **Attributes:**

  - id: int

  - fileName: string

- **Methods:** - loadimage(): void

---

8. **ImageMatcher Class:**

- **Attributes:** - threshold(): int
- **Methods:**

  - init(self, threshold)():

  - calculate_description_smiliarity(self, lost_description, found_description)():

  - calculate_image_smiliarity(self, lost_image, found_image)():

RELATIONSHIPS:

- **Inheritance:**

  - Customer and Admin inherit attributes and methods from User class.

  - FoundItem inherits attributes and methods from LostItem class.

- **Composition:**

- LostItem, FoundItem, and Description classes have a composition relationship with the Image class, meaning they contain instances of the Image class.

- **Aggregation:**

- The ImageMatcher class has an aggregation relationship with the Image class, meaning it uses instances of the Image class.

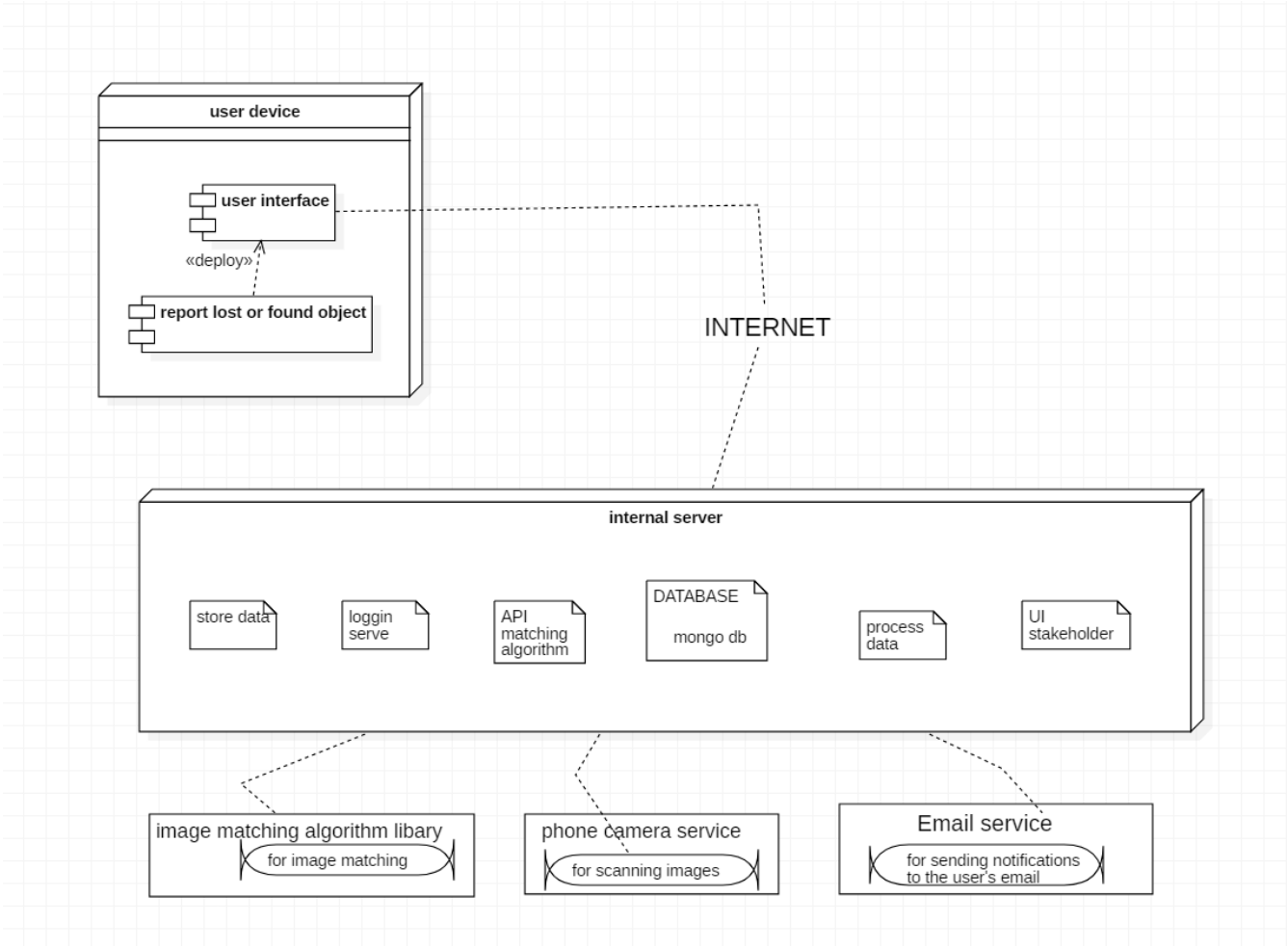
ADVANTAGES OF CLASS DIAGRAM

- **Visualization of Structure:** Class diagrams provide a visual representation of the structure of a system, illustrating the classes, their attributes, methods, and relationships.
- **Modularity:** Class diagrams promote modularity by breaking down a system into manageable and cohesive units (classes). Each class encapsulates its data and behaviors, fostering better organization and maintainability of the system.
- **Reusability:** They facilitate code reuse by identifying commonalities among classes. Through inheritance and composition relationships depicted in class diagrams, developers can leverage existing classes to build new ones, promoting a more efficient development process.
- **Abstraction:** They allow for abstraction by focusing on the essential components of the system, such as classes and their relationships, while hiding implementation details.

5. DEPLOYMENT DIAGRAM

In the context of the Unified Modeling Language (UML), a deployment diagram falls under the structural diagramming family because it describes an aspect of the system itself. In this case, the deployment diagram describes the physical deployment of information generated by the software program on hardware components. The information that the software generates is called an artifact. This shouldn't be confused with the use of the term in other modeling approaches like BPMN.

Deployment diagrams are made up of several UML shapes. The three-dimensional boxes, known as nodes, represent the basic software or hardware elements, or nodes, in the system. Lines from node to node indicate relationships, and the smaller shapes contained within the boxes represent the software artifacts that are deployed



1. USER DEVICE (CLIENT)

- Runs the application
- Displays the user interface (UI): This component is responsible for presenting information and options to the user.

2. INTERNET

Acts as the communication medium between the User Device and the internal server and external entities.

3. INTERNAL SERVER

- Processes Data: Handles the processing of data received from the user device.
- Stores Data: Manages the storage of data.
- Hosts API: Provides an API to communicate with the user device and other servers.
- Login server: verify login criteria
- Database, mongodb: stores images and user's information
- UI stakeholders: contains all the UI elements that needs to be displayed to the users.

4. IMAGE MATCHING SERVER

Provide image updates, that is whether an image was successfully matched with another image or description.

5. PHONE CAMERA SERVICES

Provides camera for snapping or scanning images.