THE UNIVERSITY OF BUEA

P.O Box 63 Buea

Buea South West Region Cameroon

Tel: (237) 233322760

REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

MINISTER OF HIGHER EDUCATION

FACULTY OF ENGINEERING AND TECHNOLOGY

Computer Engineering

CEF474 : INTERNET AND MOBILE PROGRANMMING

MOBILE APPLICATION FOR ARCHIVAL AND RETRIEVAL OF MISSING OBJECTS UTILIZING IMAGE MATCHING TECHNIQUES:

SYSTEM MODELLING AND DESIGN

COURSE INSTRUCTOR:

Dr NKEMENI VALERY

MAY 2024

PRESENTED BY

NAME	MATRICULE
EKUNDIME GLEAN MAKOGE	FE21A433
DJOUMESSI LEKANE WENDY FORTUNE	FE21A173
INDAH RISCOBELLE MBAH	FE21A204
KEDJU PRECIOUS NGWE LEKUNZE	FE21A211
NGUEDIA JEATSA JOYCE GRACE	FE21A263

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

This document serves as a blueprint for a mobile app that utilizes image matching to archive and retrieve lost objects. It outlines the system's architecture, key components, and functionalities to guide development and guarantee all user needs and technical requirements are addressed for a successful missing object management system.

1.2 Scope

This document goes through the modelling and design of the archival and retrieval of missing objects. It outlines system design. This comprehensive blueprint ensures a smooth development process and successful app deployment for managing lost objects through image matching.

1.3 Overview

This document outlines the design of the mobile app for archival and retrieval of missing objects. Using the app, users will be able to upload information about a missing object, view different missing objects that have been uploaded on the application, claim an uploaded object by uploading an image of his/her object and image matching will be done with this image and the one he/she is claiming to be his/hers, the finder pays the founder, and communicate with the founder. The app offers functionalities like user registration, object archiving, search, location tracking, and lost item reporting, ensuring a comprehensive system for managing and recovering lost possessions.

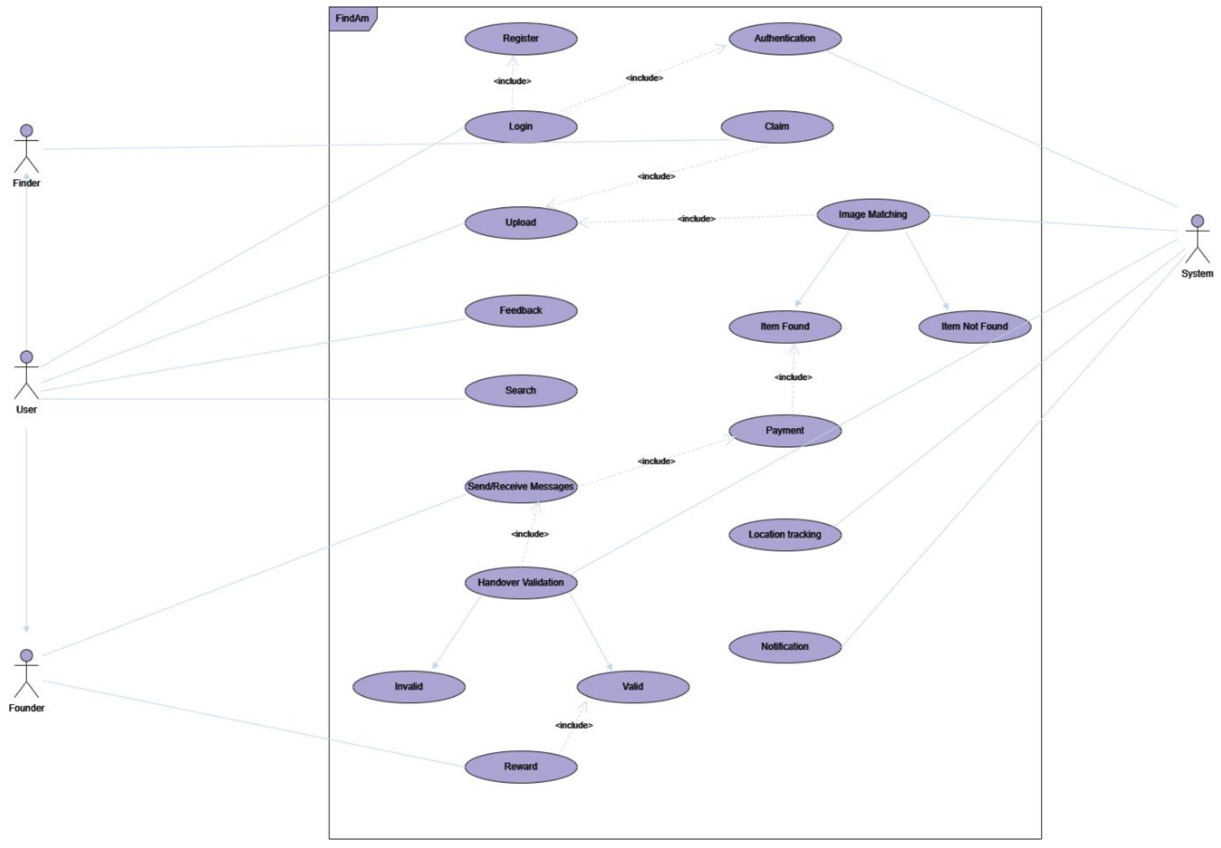
2. System Modelling

2.1 Interaction Model

The interaction model describes the dynamic exchanges between users and the application. Users interact with different features of the system. This interaction ensures a seamless experience for users using the app to report a lost objects and find their lost objects reporting lost objects.

2.1.1 Use Case Diagram

It describes a sequence of actions, including inputs, processes, and outputs, that achieves the goal of the system which is to permit users archive and retrieve documents through image matching. Each use case focuses on how users interact with the system to accomplish a specific task, ensuring that all user requirements are addressed and integrated into the system design. Each step, user input, and system response is outlined to provide a clear understanding of the user's journey through the application.



i- Actors

We have 2 actors here, the user and the system. The user is divided into two: the Finder and the Founder. The Finder is the one searching for an item, while the Founder is the one who sees the object and uploads it to the platform. The relationship here is an inheritance.

ii- Use Cases:

- Register: Users register
- Login: Users log into the system.
- Upload: Users upload images and post in the system
- Feedback: Users send feedbacks of the system
- Search: Users search for items using text based queries
- Send/Receive messages: Users send and receive messages among themselves
- Notification: Users receive alerts about the status of their reported or archived items. The system sends notifications to the user.
- Authentication: The system authenticates users for security measures
- Claim: The finder claims an object to be his/her own
- Image matching: Once a user uploads an image during the claim process, the system does the image matching which can be successful or not
- Item found: If the image matching is successful, this means the image is found and the user is taken to the next step
- Item not found: If the image matching is not successful, this means the image wasn't found to match the uploaded one and the process is aborted
- Payment: When the image matching is successful, the finder goes to do the payment which is received by the system. Once the payment is done, the finder can now message the founder to collect the object
- Handover validation: This is done as a confirmation before the money is sent to the founder and he can now give the finder his object, once the validation is valid
- Valid: This is when the handover validation is successful

- Invalid: This is when the handover validation isn’t successful
- Reward: Once the handover validation is successful, the founder is rewarded and the object handed to the finder
- Location tracking:
- Users view the location history of archived items.

2.2.2 Sequence Diagram

i. System overview (components)

- **Actor (User)**
- **Mobile App**

The mobile application is the user interface that allows users to interact with the system. It handles user requests for retrieval and archival of missing items and communicates with the server, image matching module, and payment gateway.

- **Authentication Module**

This module handles user authentication to ensure secure access to the app's features. It verifies user credentials and manages user sessions.

- **Image Matching Module**

The image matching module processes user-provided images to find similar items in the database. It sends requests to the server to retrieve metadata and file locations of similar images.

- **Server**

The server processes requests, interacts with the database to retrieve and store item metadata, and handles communications with the payment gateway.

- **Database**

The database stores metadata and file locations of items, as well as user and transaction information

- **Payment gateway**

The payment gateway handles payment transactions upon successful retrieval of missing item

ii. Authentication Sequence

- **Request**

The user request to access the app feature through the mobile app (UI)
The UI send an authentication request to the authentication module
The authentication module then through the server validates user credentials from the database.

- **Reply**

The server sends authentication response to the authentication module which is the sent to the mobile app.

iii. Retrieval Sequence Using Image Matching

- **Request**

The user requests to retrieve a missing item through the mobile application in which case the image matching module forwards the request to the server to check for a match.
The server queries for metadata of similar images from the database.

- **Reply**

The database returns to the server the request from search.
The image matching module returns the reply to the mobile app. This reply will either be for image found or image not found.

a. Image Found

I. Request

The user requests to retrieve missing item through the mobile app (UI) based on the matched image. The server queries the request to the database.

II. Reply

The mobile app initiates a payment process for retrieval through a payment gateway. The server confirms payment and sends a request to the database to return the metadata and file location (information about the finder) of the missing item. The server now sends a confirmation by directly linking the finder and founder to communicate.

b. Image not found

The database returns a not found message which through the server will be seen on the mobile app.

iv. **Archival Sequence**

- **Request**

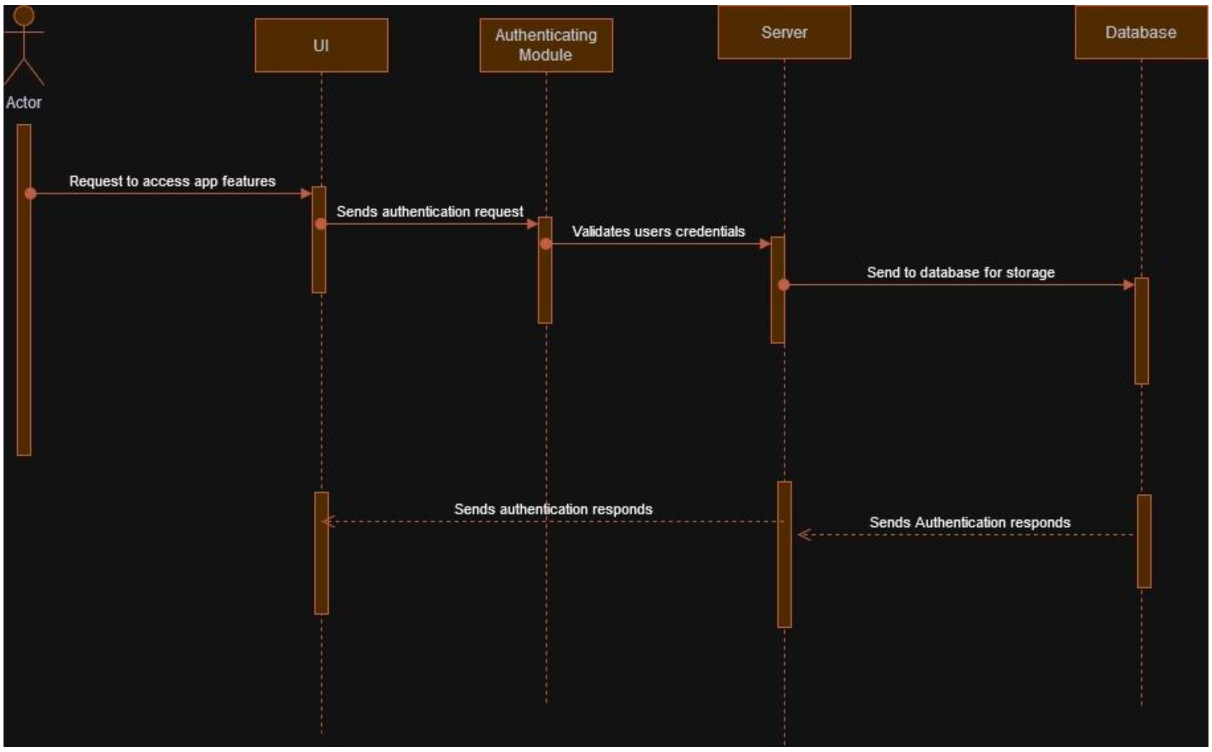
The user through the mobile app request to archive a missing item. The server queries the meta data of the item to archive to the database.

- **Reply**

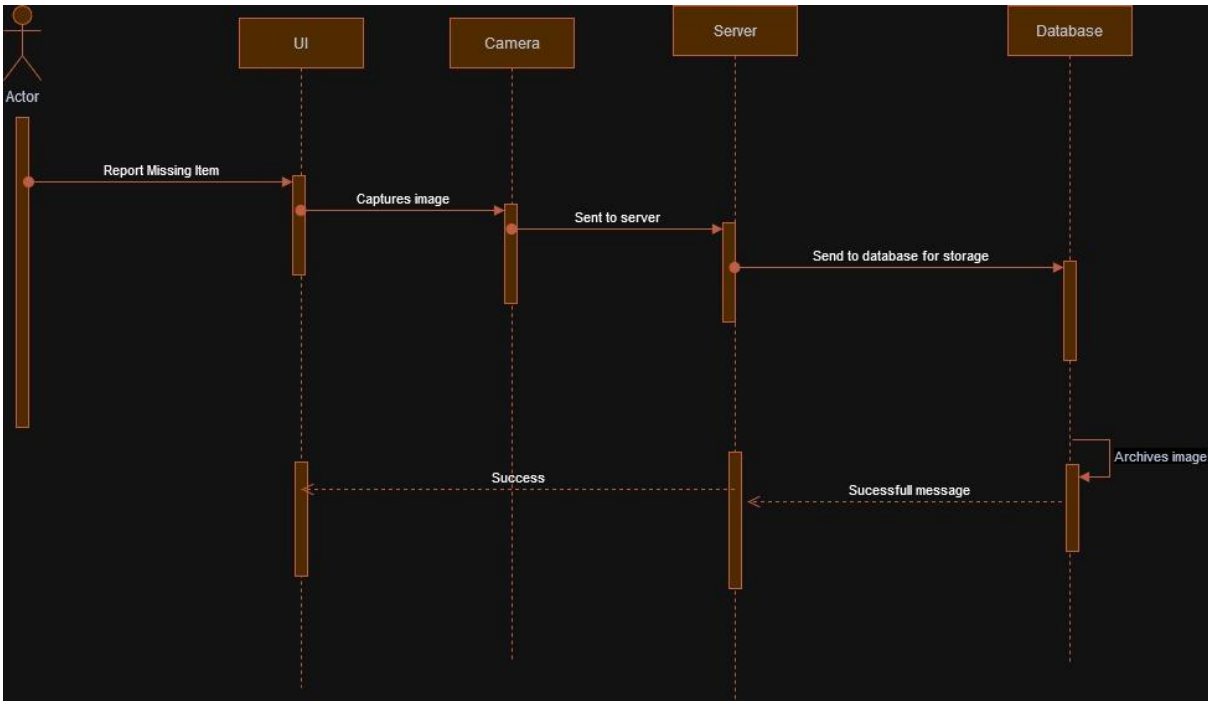
The database archives the item and sends a confirmation to the server which now sends a successful confirmation to the mobile app.

v. Schematic Diagrams

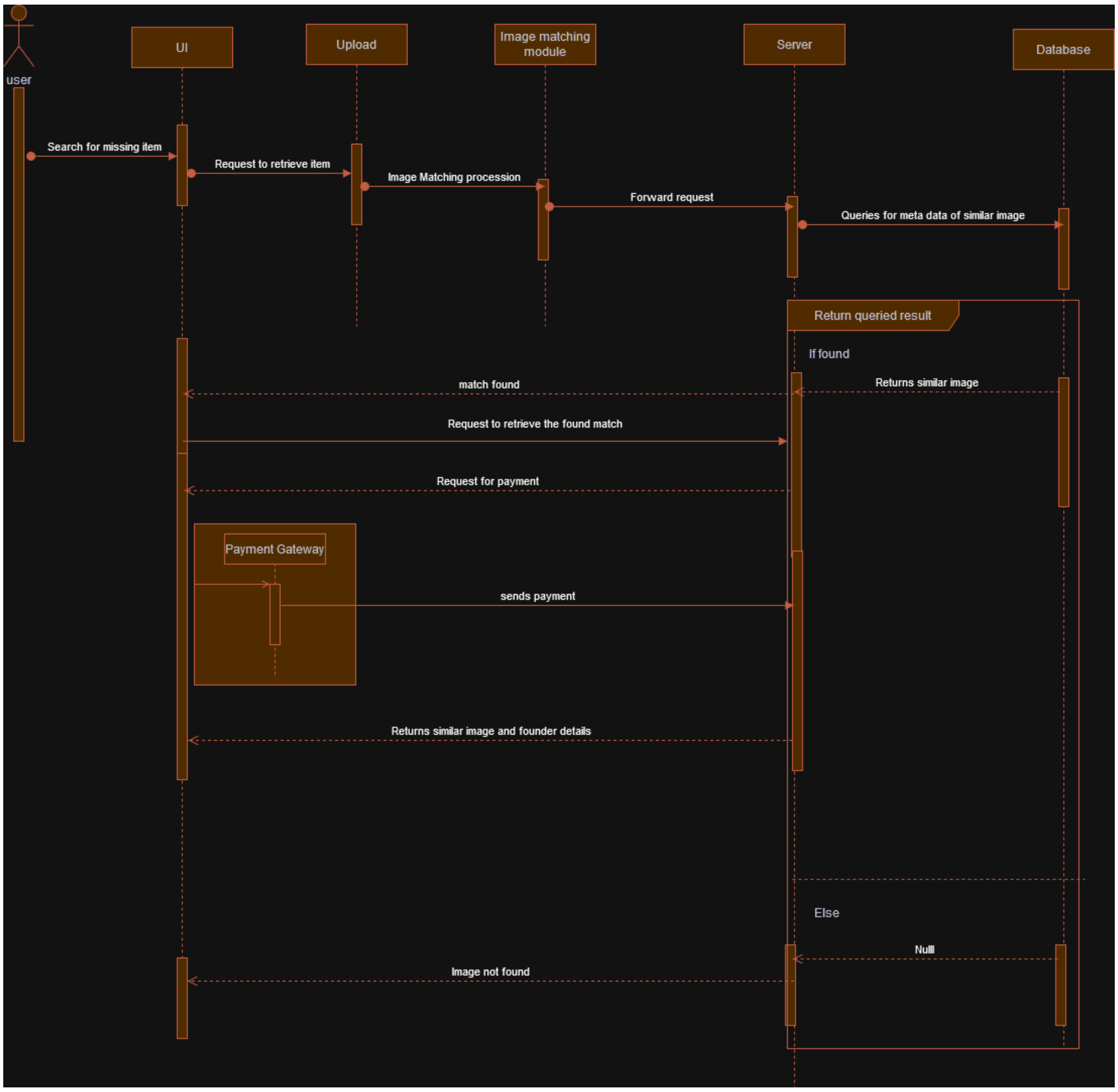
- **Authentication sequence**


Actor
UI
Authenticating Module
Server
Database
Request to access app features
Sends authentication request
Validates users credentials
Send to database for storage
Sends authentication responds
Sends Authentication responds

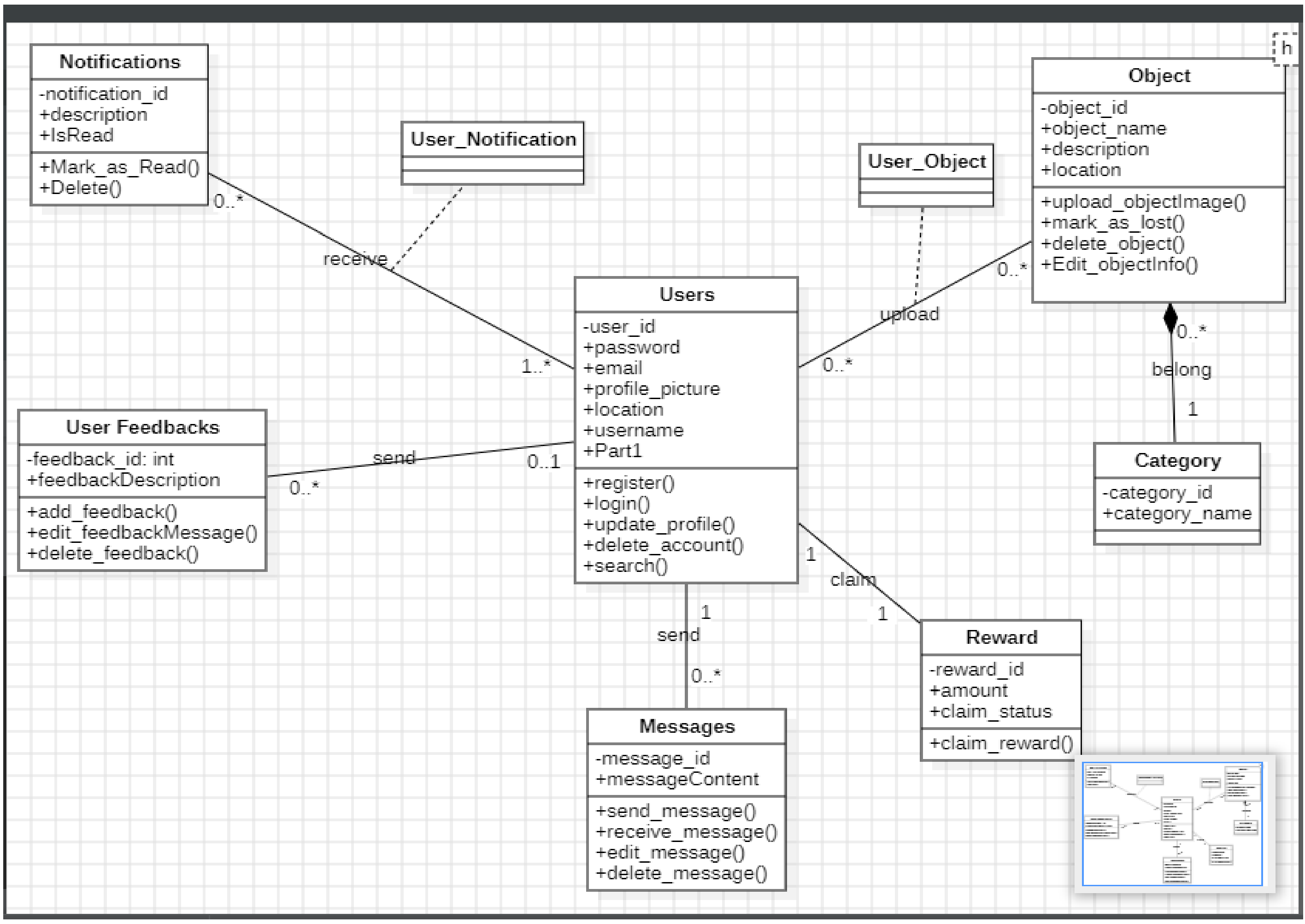
- **Achieving sequence**



- **Retrieval sequence**


user
UI
Upload
Image matching module
Server
Database
Search for missing item
Request to retrieve item
Image Matching procession
Forward request
Queries for meta data of similar image
Return queried result
If found
match found
Returns similar image
Request to retrieve the found match
Request for payment
Payment Gateway
sends payment
Returns similar image and founder details
Else
Null
Image not found

2.2.3 Class Diagram



i. **User**

- One user can have many objects (zero-to-many relationship).
- One user can have one associated reward for finding a specific item (one-to-one relationship).
- One user may receive notifications.(zero-to- many relationship)
- One user may send messages to finder.(zero-to- many relationship)
- One user may send feedbacks .(zero-to- many relationship)

ii. Item

- Many items belong to one user (many-to-one relationship).
- One item can be associated with one reward (one-to-one relationship).
- One item can be used in multiple searches and trigger notifications.

iii. Reward

- One reward is associated with one item (one-to-one relationship).

- One reward can be claimed by one user (one-to-one relationship).

iv. Notification

- Sent to one or more users based on specific events.

2.2 Context Model

2.2.1 Context Diagram

i. **System Overview**

The IARS is designed to interface with several external entities through data flows to perform its functions. These entities include users, an image database, an image recognition service, and a payment service.

ii. **External factors (entities)**

1. **User**:

   - ❖ **Role**: The user is the primary interactant with the system and is responsible for uploading images, submitting search queries, and retrieving images upon successful payment.
   - ❖ **Interactions**: Uploads images, submits search queries, and receives search results after payment confirmation.

2. **Image Database**:

   - ❖ **Role**: The database stores all images and meta data
   - ❖ **Interactions**: Receives and stores images and metadata from the system. It also provides images and metadata upon retrieval request.

3. **Image Recognition Service**:

   - ❖ **Role**: It analyses and recognizes content within uploaded images, returning descriptive data to the system.

- ❖ **Interactions**: Receives images for analysis and returns recognition data.

4. **Payment Service**:

    - ❖ **Role**: Processes payments for image retrieval, ensuring that retrievals are authorized only after successful payment.
    - ❖ **Interactions**: Receives payment requests and confirms payment status to IARS.

iii. **Data Flows**

1. **Upload Image**:
    - ❖ **Source**: From the user
    - ❖ **Destination**: To the system (FindAm)
    - ❖ **Description**: The user uploads an image to the system.
2. **Store Image**:

    - ❖ **Source**: From the system (FindAm)
    - ❖ **Destination**: Image Database
    - ❖ **Description**: FindAm stores the uploaded image in the database along with initial metadata.

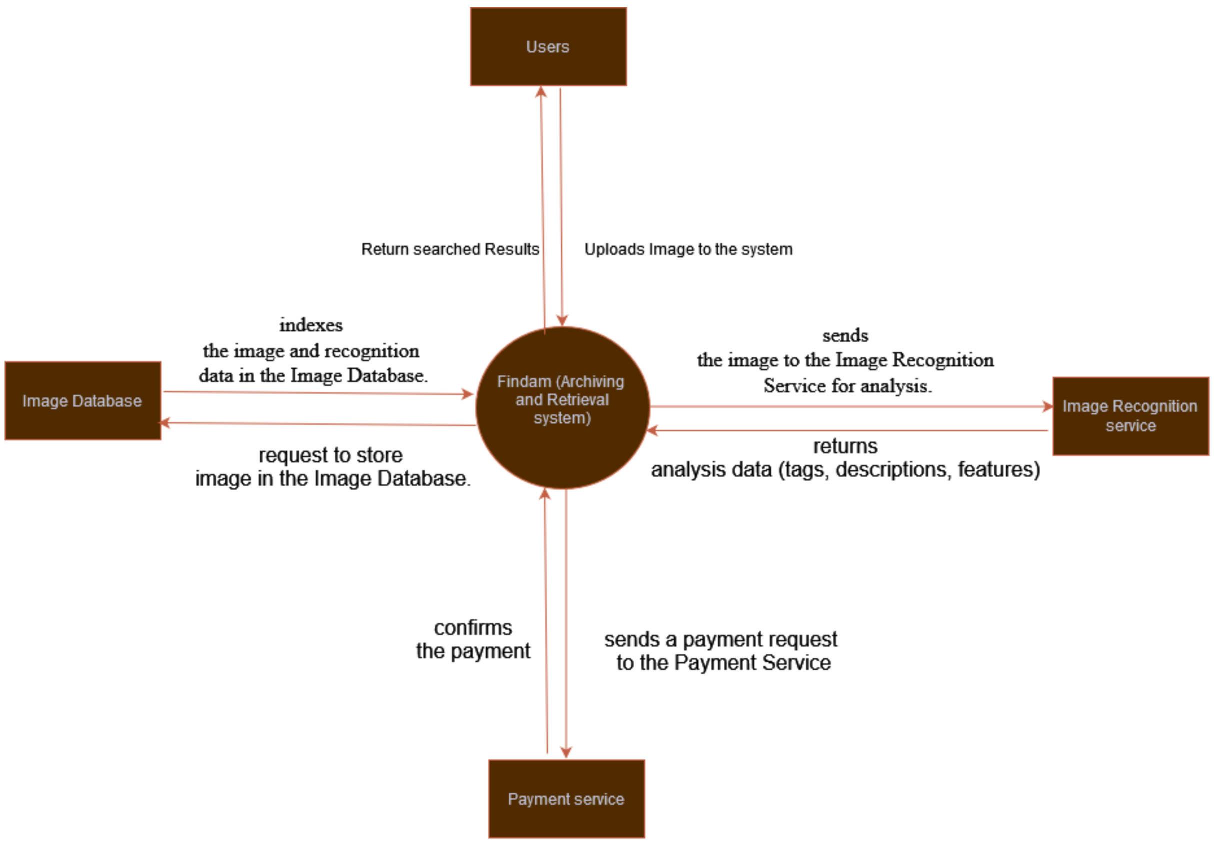
3. **Request Image Analysis**:

    - ❖ **Source**: From the system (FindAm)
    - ❖ **Destination**: Image Recognition Service
    - ❖ **Description**: FindAm sends the uploaded image to the recognition service for analysis.

4. **Receive Recognition Data**:
    - ❖ **Source**: Image Recognition Service
    - ❖ **Destination**: To System (FindAm)
    - ❖ **Description**: The recognition service returns analyzed data, including tags, descriptions, and features, to the system.
5. **Index Image**:
    - ❖ **Source**: From the Sytsem

- **Destination**: Image Database
- **Description**: The System indexes the image with the received recognition data and stores it in the database.

6. **Payment Request**:
   - **Source**: From system
   - **Destination**: Payment Service
   - **Description**: The system sends a payment request to the payment service when a search query is made.
7. **Payment Confirmation**:
   - **Source**: Payment Service
   - **Destination**: System (FindAm)
   - **Description**: The payment service confirms the payment to the stystem.
8. **Return Search Results**:
   - **Source**: From the system (FindAm)
   - **Destination**: User
   - **Description**: The system returns the search results to the user.


Users
Return searched Results
Uploads Image to the system
indexes the image and recognition data in the Image Database.
Image Database
request to store image in the Image Database.
Findam (Archiving and Retrieval system)
sends the image to the Image Recognition Service for analysis.
Image Recognition service
returns analysis data (tags, descriptions, features)
confirms the payment
sends a payment request to the Payment Service
Payment service

2.3 Structural Model

2.3.1 Deployment Diagram

i. Nodes and Components

- **User Device (Mobile Phone)**
  - ➢ Mobile App
  - ➢ Authentication Module
  - ➢ Image Matching Module
- **Web Server**
  - ➢ Application Server
  - ➢ Image Matching Service
  - ➢ Authentication Service
- **Database Server**
  - ➢ Database Management System (DBMS)
  - ➢ Image Metadata Database
  - ➢ User Data Database
- **Payment Gateway**
  - ➢ Payment Processing Service
- **Cloud Storage**
  - ➢ Image Storage Service

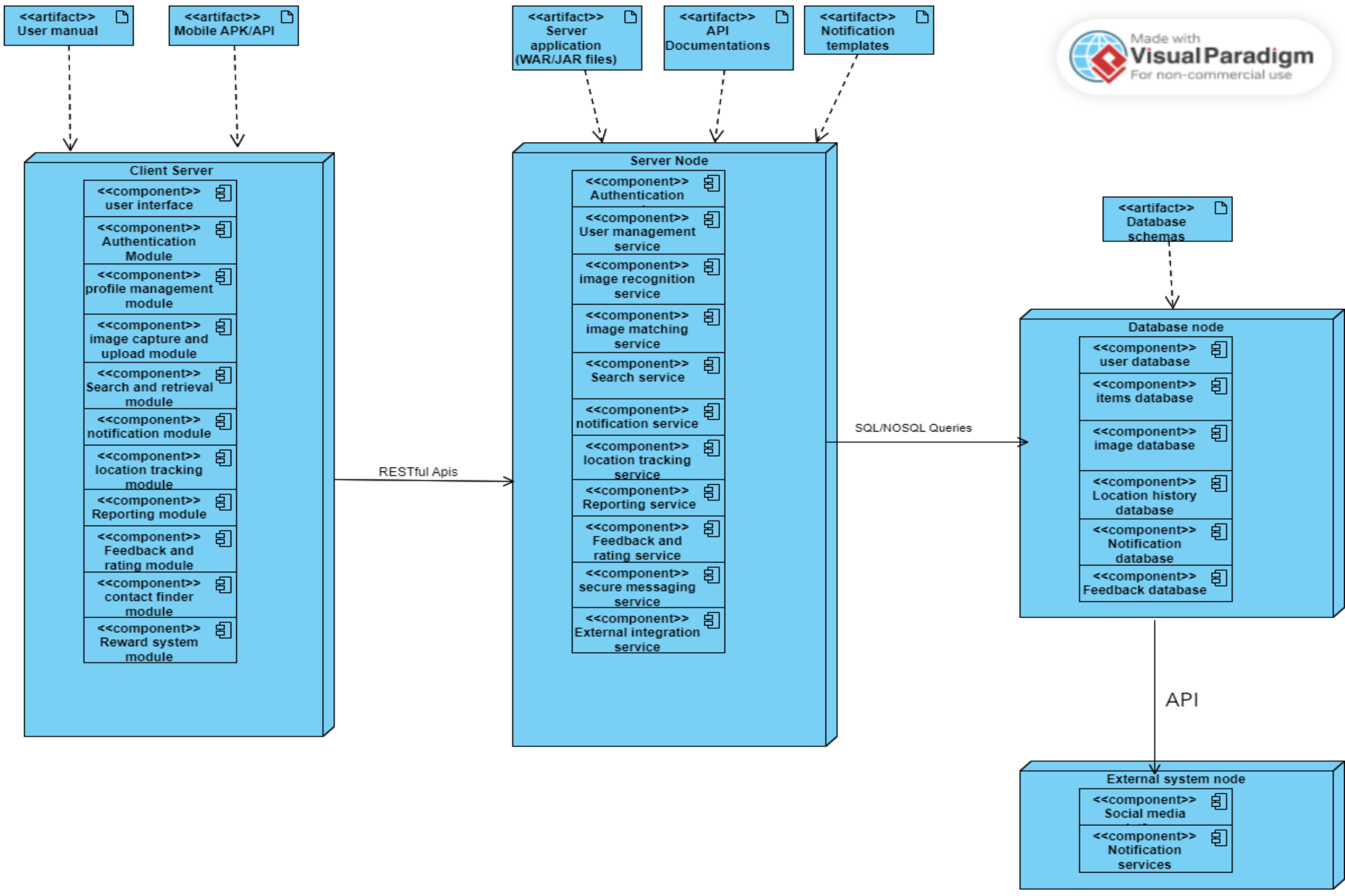
ii. Connections

- **Mobile App** on the **User Device** communicates with the **Application Server** on the **Web Server**.
- **Application Server** interacts with the **Image Matching Service** and **Authentication Service** on the **Web Server**.
- **Application Server** queries the **Database Server** for image metadata and user data.
- **Application Server** communicates with the **Payment Processing Service** on the **Payment Gateway** for handling payments.

- **Application Server** accesses **Cloud Storage** for storing and retrieving images.

iii. Deployment options

- **Cloud Deployment:** All components can be hosted on a cloud platform like AWS, Google Cloud Platform, or Azure. This offers scalability and flexibility.
- **On-Premise Deployment:** Some or all components can be hosted on the organization's own servers, providing more control but also requiring more maintenance.


<<artifact>> User manual
<<artifact>> Mobile APK/API
<<artifact>> Server application (WAR/JAR files)
<<artifact>> API Documentations
<<artifact>> Notification templates
Made with Visual Paradigm For non-commercial use
Client Server
<<component>> user interface
<<component>> Authentication Module
<<component>> profile management module
<<component>> image capture and upload module
<<component>> Search and retrieval module
<<component>> notification module
<<component>> location tracking module
<<component>> Reporting module
<<component>> Feedback and rating module
<<component>> contact finder module
<<component>> Reward system module
RESTful Apis
Server Node
<<component>> Authentication
<<component>> User management service
<<component>> image recognition service
<<component>> image matching service
<<component>> Search service
<<component>> notification service
<<component>> location tracking service
<<component>> Reporting service
<<component>> Feedback and rating service
<<component>> secure messaging service
<<component>> External integration service
SQL/NOSQL Queries
<<artifact>> Database schemas
Database node
<<component>> user database
<<component>> items database
<<component>> image database
<<component>> Location history database
<<component>> Notification database
<<component>> Feedback database
API
External system node
<<component>> Social media
<<component>> Notification services

3. Conclusion

The system modelling and design document provides a comprehensive blueprint for developing the mobile app. It covers user interactions, system components, and data flows, ensuring a seamless and secure experience. By detailing each stage of the development process, this guarantees that all user requirements are met, leading to a reliable and efficient solution for managing lost items. This structured approach ensures the successful implementation and deployment of the application.

4. References

1. GeeksforGeeks. (2023, May 22). Context Diagrams. https://www.geeksforgeeks.org/context-diagrams/
2. BoardMix. (n.d.). Tips for Creating a Context Diagram. https://boardmix.com/tips/what-is-user-flow-diagram/
3. OpenAI's ChatGPT (https://openai.com/chatgpt/) Conversation (May 28, 2024).
4. Creately. (n.d.). Sequence Diagram with Android Application https://creately.com/diagram/example/ht6pptvm1/mobile-app
5. Al-Qutaibi, A. M., & Abdullah, M. N. (2016). Mobile-Based Archival and Retrieval of Missing Objects Using Image Matching. https://www.researchgate.net/publication/283290085_Archival_and_Retrieval_of_Lost_Objects_using_Multi-feature_Image_Matching_in_Mobile_Applications