THE UNIVERSITY OF BUEA
P.O Box 63,
Buea, South West Region
CAMEROON
Tel: (237) 3332 21 34/3332 26 90
Fax: (237) 3332 22 72

REPUBLIC OF CAMEROON

Peace-Work-Fatherland

FACULTY OF
ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

CEF440: INTERNET PROGRAMMING (J2EE) AND MOBILE PROGRAMMING

DESIGN AND IMPLEMENTATION OF A MOBILE-BASED ARCHIVAL AND RETRIEVAL OF MISSING OBJECTS APPLICATION USING IMAGE MATCHING

TASK FOUR: **SYSTEM MODELLING AND DESIGN**

Course Supervisor:
Dr. NKEMENI VALERY
University Of Buea

GROUP 1 MEMBERS

	NAME	MATRICULE
1	MEWOABI NGUEFACK DORE	FE21A239
2	ALEANU NTIMAEH ENOW	FE21A134
3	OJONG-ENYANG OYERE	FE21A297
4	FONGANG KELUAM PAUL DIEUDONNE	FE21A193
5	TANWIE BRUNO ADEY	FE21A316

Contents

1. INTRODUCTION

1.1. Background:

In an era characterized by rapid technological advancement and an ever-increasing reliance on mobile applications, the need for innovative solutions to everyday challenges has never been more pressing. One such challenge is the distressing experience of losing personal belongings, be it a cherished item of sentimental value or a practical necessity crucial for daily routines. The emotional toll and inconvenience caused by such losses are undeniable, often compounded by the arduous process of attempting to retrieve or replace the missing objects.

Recognizing the urgency of addressing this issue, the proposed project sets out to develop a groundbreaking mobile application aimed at revolutionizing the archival and retrieval of missing objects through the power of image matching technology. This application will not only streamline the process of reporting lost items but also enhance the likelihood of successful recovery, thereby alleviating the stress and frustration associated with such incidents.

1.2. System modelling and Design:

The System Modeling and Design phase is crucial in the development of the "Mobile-Based Archival and Retrieval of Missing Objects Application Using Image Matching." This phase translates user requirements into a blueprint that guides the construction of the software, ensuring that the final product is robust, scalable, and meets user needs effectively. In the context of our project, this phase ensures that the complex functionality of image matching for identifying and retrieving missing objects is well-defined and systematically organized. It lays the groundwork for implementing features such as image capture, storage, community interaction, and efficient retrieval mechanisms. By meticulously planning and documenting the system's architecture and interactions, we mitigate risks, address potential issues early, and streamline the development process.

In this phase, we began with the architecture design, defining the overall structure of the application, including major components and their interactions, ensuring scalability, security, and performance by outlining the system's physical and logical layers. Next, we created a context diagram to provide a high-level view of the system and its environment, illustrating interactions between the system and external entities such as users, databases, and external services. The use case diagram was developed to capture functional requirements and interactions between actors like users and administrators, identifying key functionalities such as creating posts, viewing posts, commenting, and image matching. We then created a class diagram to depict the system's static structure by showing classes, their attributes, methods, and relationships, facilitating object-oriented design and helping in identifying reusable components. A sequence diagram was used to describe how objects interact in a particular sequence to perform a function, visualizing the flow of operations for critical processes like image matching and post retrieval. Finally, the deployment diagram was developed to show the physical deployment of the system on hardware, mapping software components to hardware nodes, ensuring clarity on how the system will run in real-world environments. By rigorously designing each aspect of the system, we create a

comprehensive guide that ensures the application is developed efficiently, aligns with user expectations, and is maintainable and extensible for future enhancements.

2. SYSTEM DETAILED DESIGN SPECIFICATION

2.1. SYSTEM ARCHITECTURE

System architecture defines the structured solution that meets all technical and operational requirements, while optimizing common quality attributes such as performance, security, and manageability. It encompasses the overall design and organization of a software system, delineating how different components and services interact within a network to achieve specific functionalities. By breaking down the system into distinct layers or tiers, system architecture enables scalability, maintainability, and flexibility, ensuring that the system can evolve with changing requirements and technologies.

Our "Mobile-Based Archival and Retrieval of Missing Objects Application Using Image Matching" employs a 4-tier (n-tier) architecture, which is divided into the following components and submodules:

1) Presentation Tier (Client Tier):
   This includes the components that the user directly interacts with. They include:

   i) User Interface (UI): Developed using cross-platform frameworks like React Native or Flutter to ensure consistency across iOS and Android devices.
   ii) User Interaction components: Handles user inputs, image capture, uploads, and displays search results and notifications.
   iii) Client-Side Logic: Basic validation and preprocessing of user inputs before sending them to the server.

2) Application Tier (Business Logic Tier):
   This is the core part where all the business logic and algorithmic modules for handling various processes and operations are found. We can locate services or modules such as:

   i) API Gateway:
   Acts as a single-entry point for client requests, handling routing, authentication, rate limiting, and request/response transformation.

   ii) Business and logic services
   This involves services such as

   - **User Management Service**: Manages user authentication, registration, and profile management.
   - **Image Processing Service**: Handles image uploads, preprocessing, and integration with image recognition APIs.
   - **Notification Service**: Manages push and email notifications.
   - **Search Service**: Facilitates search functionalities, including filtering and sorting of search results.
   - **Location Service**: Integrates geolocation data for location-based search and mapping features.

3) Data Tier (Persistence Tier):

This Layer contains the models and responsible for data validation and verification as well as the databases for storing and retrieving this data throughout the app. It includes components such as

i) Database:

NoSQL Database: Use a NoSQL database like MongoDB to store unstructured data, including images and logs, also to store structured data such as user information, search metadata, and notifications.

ii) Object Storage:

Use cloud storage solutions like Firebase storage, Amazon S3 or Google Cloud Storage for storing large binary objects (e.g., images).

iii) Caching:

Implement caching mechanisms (e.g., Redis) to store frequently accessed data and reduce database load.

4) Integration Tier (Service Integration Layer):

This includes third party integrated modules which are vital to the functioning of the application such as:

i) *Image Recognition APIs*: Integrate with services like Google Cloud Vision or AWS Recognition for image matching.
ii) *Notification Services*: Use services like Firebase Cloud Messaging (FCM) or AWS SNS for push notifications.
iii) *Geolocation Services*: Integrate with mapping and location services like Google Maps API for geolocation features.

The Architecture for the system is as depicted in the diagram on the next page:

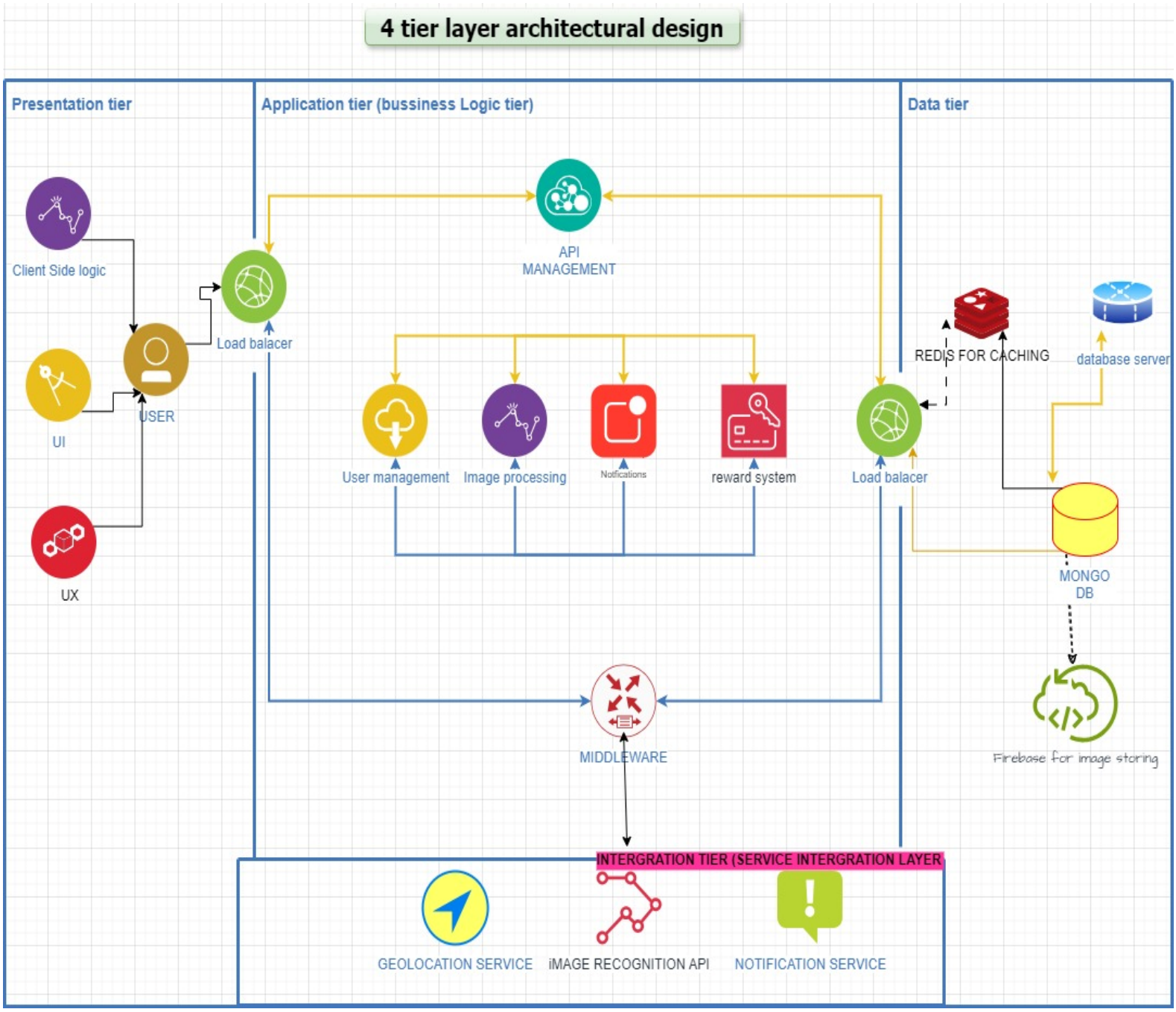

Fig2.1. System Architecture Design for a mobile based application for archival and retrieval of lost items using image matching, showing all layers as well as their modules and relations

2.2. CONTEXT DIAGRAM

The context diagram for the "Mobile-Based Archival and Retrieval of Missing Objects Application Using Image Matching" provides a high-level overview of the system's interactions with external entities. It illustrates how users and external services engage with the application, highlighting the primary roles and their respective actions. The context diagram for the system is as seen below

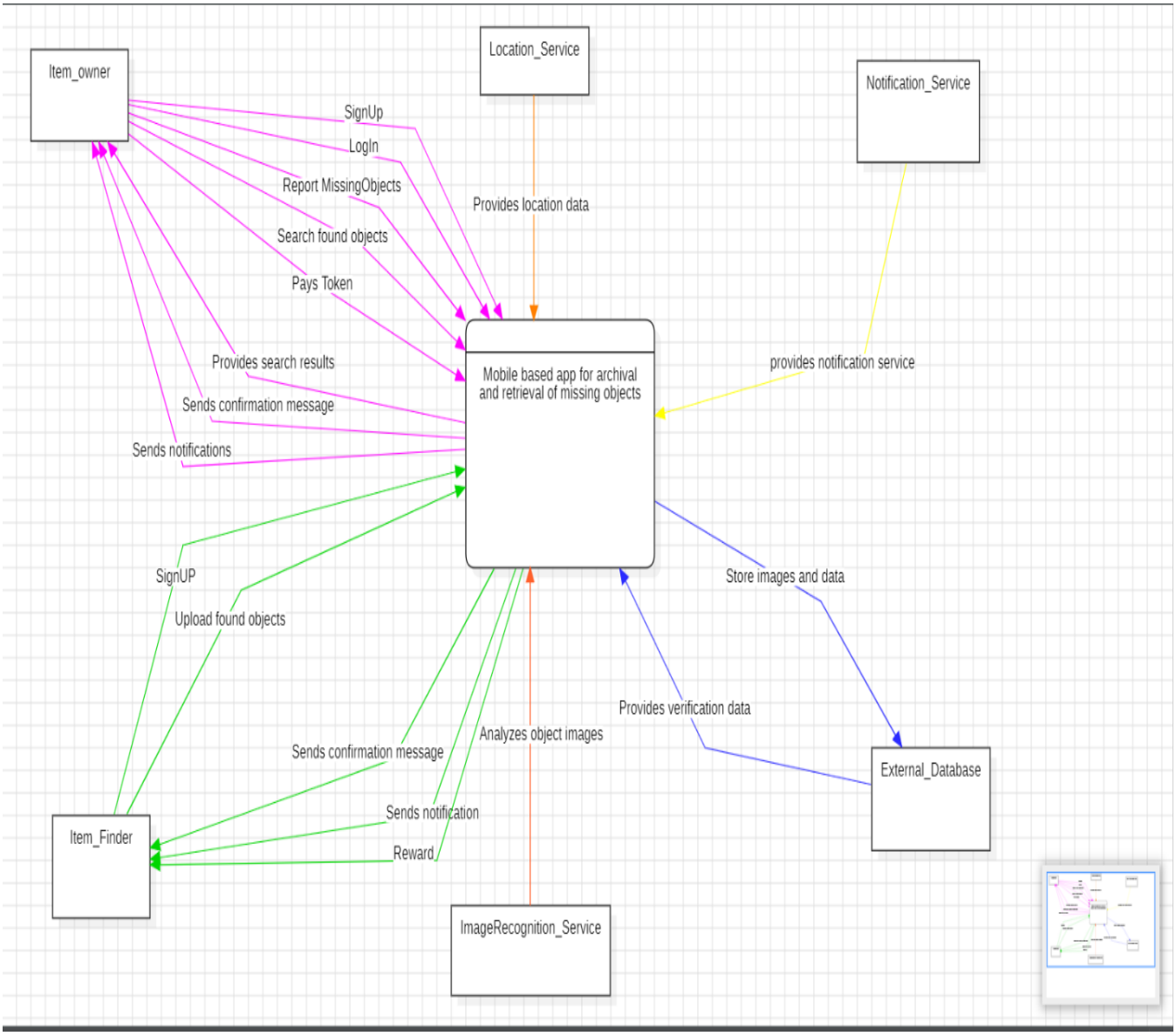

Fig2.2. Context Design for a mobile based application for archival and retrieval of lost items using image matching, showing the relationship between the system and all external entities

Explanation of context diagram

i. Item Owners:

Role: Individuals who have lost items and are looking to report them and search for found items.

Interactions with the System:

i) Sign Up: Item owners register on the system to create an account.
ii) Report Lost Item: They submit details about their lost items, including descriptions, photos, and the last known location.
iii) Search Found Items: Item owners search the database for items that have been found and reported.
iv) Receive Notifications: They receive updates regarding the status of their lost items, such as matches with found items or updates from the admin.

ii. Item Finders:

Role: Individuals who have found items and are reporting them to the system.

Interactions with the System:

i) Sign Up: Item finders register on the system to create an account.
ii) Report Found Item: They submit details about the items they have found, including descriptions, photos, and the location where the item was found.
iii) Receive Notifications: They receive updates about the status of the items they have reported, such as when an item owner claims their found item.

iii. Notification Services:

Role: External service responsible for sending notifications to users.

Interactions with the System:

i) Send Notifications: The system sends data to the notification service, which then sends emails, SMS, or app notifications to item owners and finders regarding status updates, matches, and other relevant information.

iv. Location Services:

Role: External service providing geolocation data.

Interactions with the System:

i) Provide Location Data: The system requests location data to help pinpoint where items were lost or found, improving search accuracy and providing context to reports.

v. External Database:

Role: Third-party databases for verification and additional data sources (e.g., police reports, other lost-and-found systems).

Interactions with the System:

i) Provide Verification Data: The system queries the external database to verify reported items, cross-reference lost and found reports, and enhance data accuracy.

vi. Image Recognition Service:

Role: External service that analyzes images of lost and found items.

Interactions with the System:

i) Analyze Item Images: The system sends images of reported lost and found items to the image recognition service, which analyzes and provides metadata, tags, and potential matches to help identify and match items.

2.3. USE CASE DIAGRAM

The use case diagram for the "Mobile-Based Archival and Retrieval of Missing Objects Application Using Image Matching" delineates the functional requirements of the system from the perspective of its users. This diagram identifies the various actors interacting with the system and the key activities they perform. By mapping out these interactions, the use case diagram ensures that all user needs are addressed, guiding the development process to create a user-centric application.

Actors and Their Use Cases

Primary Actors:

i) Item Owners:

- Sign Up: Register on the system to create an account.

- Login: Access the system using their credentials.

- Report Lost Item: Submit details about their lost items, including descriptions, photos, and the last known location.

- Search Found Items: Look through the database for items that have been found and reported.

- Receive Notifications: Get updates regarding the status of their lost items, such as matches with found items or admin updates.

- Update Profile: Modify personal information and settings.

Logout: Exit the system securely.

ii) Item Finders:

- Sign Up: Register on the system to create an account.

- Login: Access the system using their credentials.

- Report Found Item: Submit details about the items they have found, including descriptions, photos, and the location where the item was found.

- Receive Notifications: Get updates about the status of the items they have reported, such as when an item owner claims their found item.

- Logout: Exit the system securely.

Secondary Actors:

None

Tertiary Actors:

i) Notification Services:

- Send Notifications: Send emails, SMS, or app notifications to users regarding status updates, matches, and other relevant information.

iii) External Database:

-Provide Verification Data: Supply additional data to verify reported items, cross-reference lost and found reports, and enhance data accuracy.

iv) Image Recognition Service:

-Analyze Item Images: Analyze images of reported lost and found items, providing metadata, tags, and potential matches to help identify and match items.

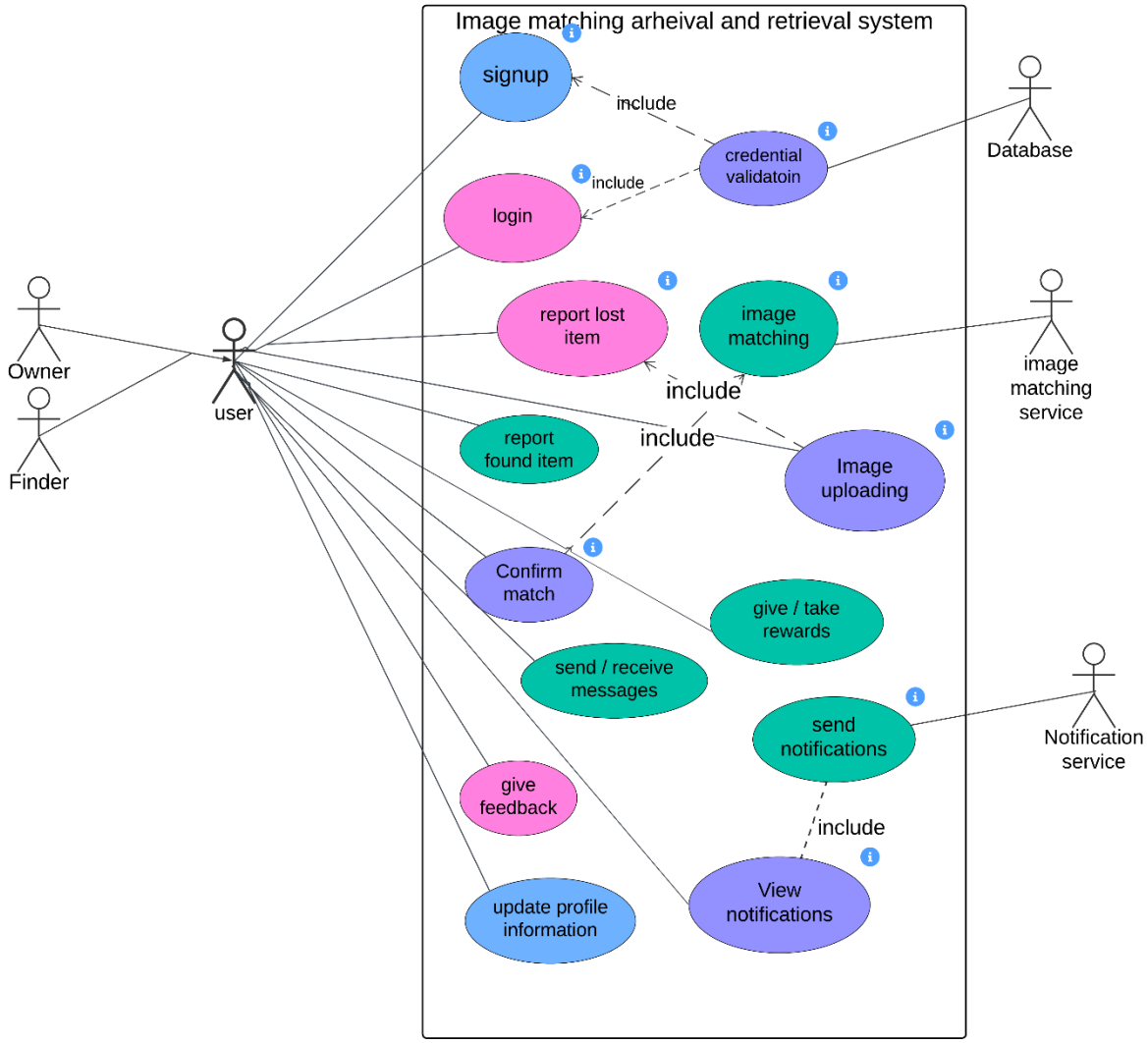

Fig2.3. UseCase diagram Design for a mobile based application for archival and retrieval of lost items using image matching, showing all primary and tertiary actors and their usecases in the system

2.4. SEQUENCE DIAGRAM

The sequence diagram for our system illustrates the step-by-step interactions between various components during specific scenarios. It provides a concise visualization of how different parts of the system communicate and collaborate to achieve a particular task. By showcasing the sequence of method calls and responses, the diagram offers insights into the dynamic behavior of the system in real world scenarios, helping to validate functionality, refine design decisions, and ensure smooth operation.

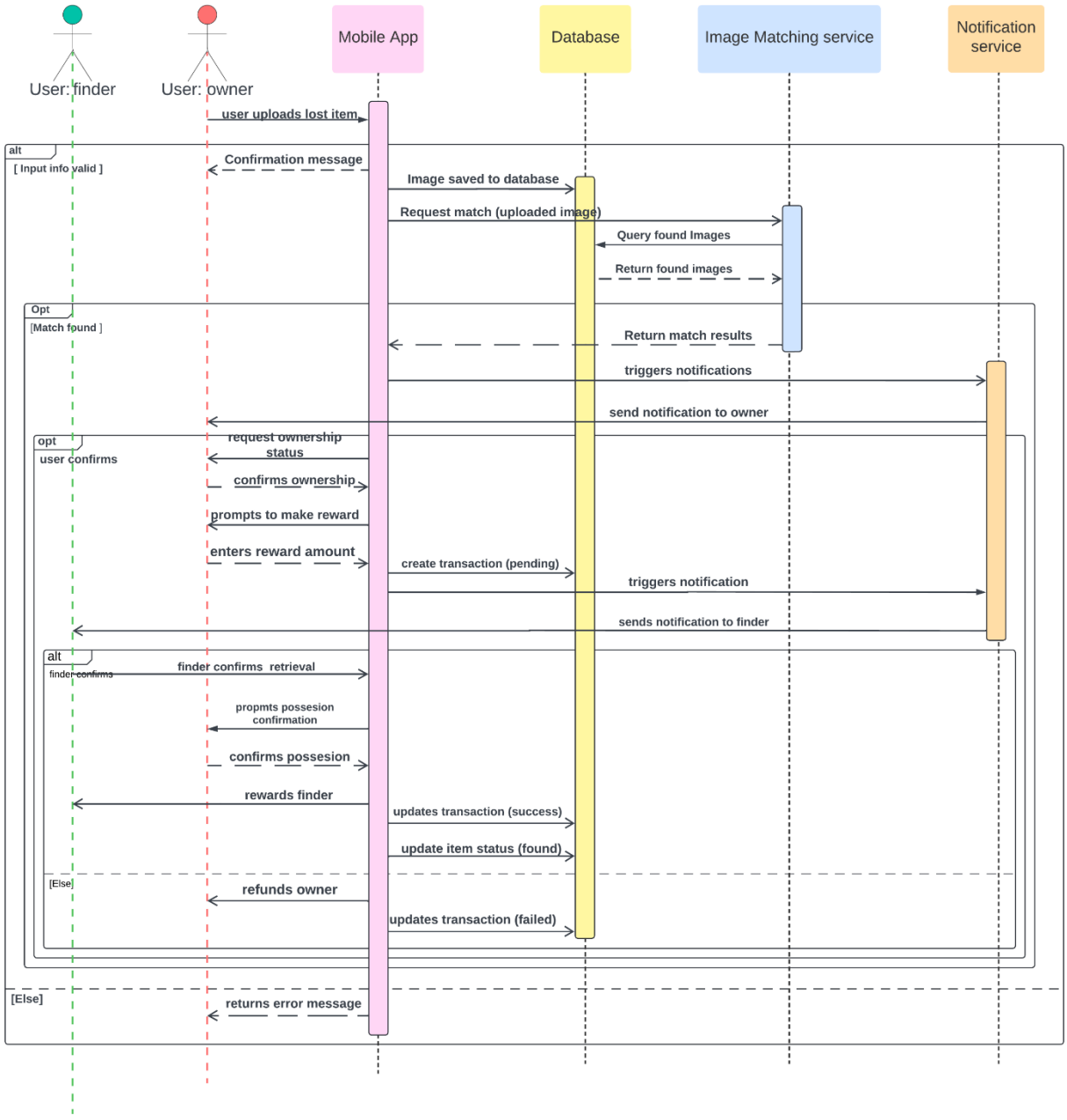

Fig2.4. Sequence diagram Design for a mobile based application for archival and retrieval of lost items using image matching, showing the order of occurrence of actions and processes in the system

2.4.1. Explanation of sequence diagram

The actions taking place in the sequence diagram are orderly listed and explained below

i) A user (the item owner) wants to report a missing item so he inputs the information about the item to the application

ii) If the input is correct: he receives a confirmation message from the application telling him he is going to be notified if a match for his item if found

iii) the Application stores that item information to the database

iv) The application initiates a request to perform a match to the image matching service
v) The image matching service queries the database for images of all found items
vi) The database complies by returning all the images of found objects in the database
vii) If there is a match, the image processing service returns that match status to the application or system
viii) The application request for a notification to be sent to the user by the notification service
ix) The notification service sends this notification to the owner alerting them that a potential match has been found
x) The application also prompts the user to confirm or deny the match if it their item or not respectively
xi) If the user confirms the match, then the application prompts the user to enter a reward to be given to the finder
xii) If the owner enters the reward, the application creates a transaction with a status of pending which is stored in the database
xiii) The application also requests for the notification service to send a notification to the finder that a user wants to retrieve the found item for them to communicate
xiv) If finder confirms that the item is actually that of the user using the description and metadata the user entered or after their discussions, the finder confirms the transaction and returns the object to the user
xv) When the owner confirms they have received their item, they confirm too and the application sends the reward to the finder
xvi) The transaction is changed in the database by the application to completed and the item status is updated to found so that it is not used in future matches by the image matching service
xvii) If the finder does not confirm, then the application refunds the user and the owner is refunded and the transaction status is set to fail

The same procedure above is repeated when the finder uploads a found item instead, except at the point of the image matching service since in this instance it instead requests for the lost items for matching and not the found items.

2.5. CLASS DIAGRAM

The class diagram for our "Mobile-Based Archival and Retrieval of Missing Objects Application Using Image Matching" outlines the core entities and their interactions within the system. It serves as a blueprint for the system's object-oriented design, detailing key classes such as User, Item, Notification, Search, Reward, Message, and additional supporting classes like. Each class encapsulates specific attributes and methods, ensuring a structured approach to managing user interactions, item reporting, notifications, searches, rewards, messaging, and third-party integrations. This diagram facilitates a clear understanding of the system's architecture and guides the development process, ensuring coherence and scalability.

1. User Class

- Attributes:

- UserID (int)
- Username (String)
- Password (String)
- Email (String)
- PhoneNumber (String)
- BiometricData (String)
- ProfilePicUrl (String)
- DateJoined (Date)

- Methods:

- register ()
- login ()
- logout ()
- updateProfile ()
-uplaodImage ()
- reportMissingItem ()
- declareFoundItem (

2. Item Class

Attributes:

- ItemID (int)
- ItemName (String)
- Description (String)
- ImageUrl (String)
- Date (Date) //*date lost or found*
- Location (String) //*location lost or found*
- ReportedBy (int) // *UserID of the finder or owner*
- status (Enum) lost/found

Methods

- addItem ()
- updateItem ()
- removeItem ()
- searchItem ()

3. Notification Class

Attributes:

- NotificationID (int)
- UserID (int)
- Message (String)
- DateSent (Date)
- ReadStatus (boolean)

Methods

- sendNotification ()
- markAsRead ()
- deleteNotification ()

4. Reward Class

Attributes

- RewardID (int)
- Amount (float)
- ItemID (int)
- UserID (int)
- Status (String) // Possible values: 'Unclaimed', 'Claimed', 'Released'

Methods:

- createReward ()
- updateReward ()
- claimReward ()
- releaseReward ()

5. Message Class

Attributes:

- MessageID (int)
- SenderID (int)
- ReceiverID (int)
- Content (String)
- Timestamp (Date)

Methods:

- sendMessage()
- readMessage ()
- deleteMessage ()

6. Transaction

Attributes:

- OwnerId (int)
- ItemID (int)
- FinderID (int)
- rewardID (int)
- status (Enum) // *complete, failed pending*

Methods:

- createTransaction ()
- updateTransaction (status)

The class diagram is represented as in the image below

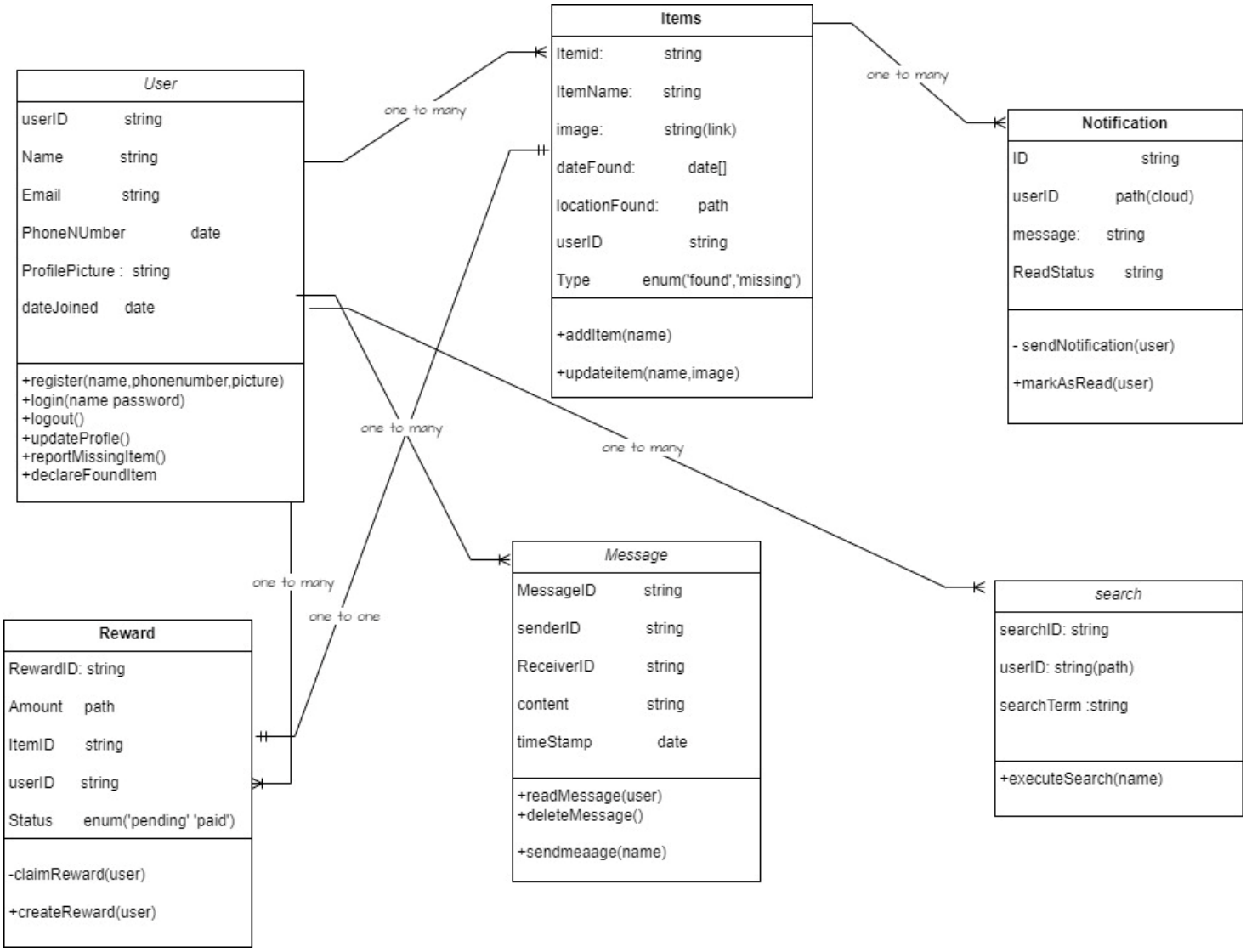

Fig2.5. Class diagram Design for a mobile based application for archival and retrieval of lost items using image matching, showing all entities and the relation between them

2.6. DEPLOYMENT DIAGRAM

The deployment diagram illustrates the architecture of the lost-and-found item reporting system, highlighting the interaction between the mobile client applications and the backend services hosted in the cloud infrastructure. This system allows users to report lost or found items through mobile applications, which then processes these reports using backend services to facilitate image matching, user notifications, and real-time chat communication.

Key Components:

i) Client Devices:

The system supports two types of client devices: Mobile App (iOS) and Mobile App (Android). Users interact with these applications to log in, report items, and communicate via chat.

ii) Backend Services:

A set of cloud-based services handle various functionalities:

a) Authentication Service: Manages user authentication for login and signup.

b) User Management Service: Manages user profiles and related data.

c) Item Management Service: Handles reporting of lost and found items.

d) Image Processing Service: Processes and matches images of items.

e) Notification Service: Sends notifications to users.

f) Chat Service: Facilitates real-time communication between users.

g) Databases: The system utilizes several databases to store data:

h) User DB: Stores user information.

I) Item DB: Stores information about reported items, including images.

j) Chat DB: Stores chat messages between users. All of the databases mentioned are a centralized database which stores information of everything in the system.

k) External Services: An external Push Notification service is used to send notifications to users.

iii) Communication and Dependencies:

The diagram showcases the communication lines between the mobile applications and backend services, as well as dependencies between various backend components and databases. Each interaction and dependency is labeled for clarity, illustrating how data flows through the system and how different services interact to provide a seamless user experience.

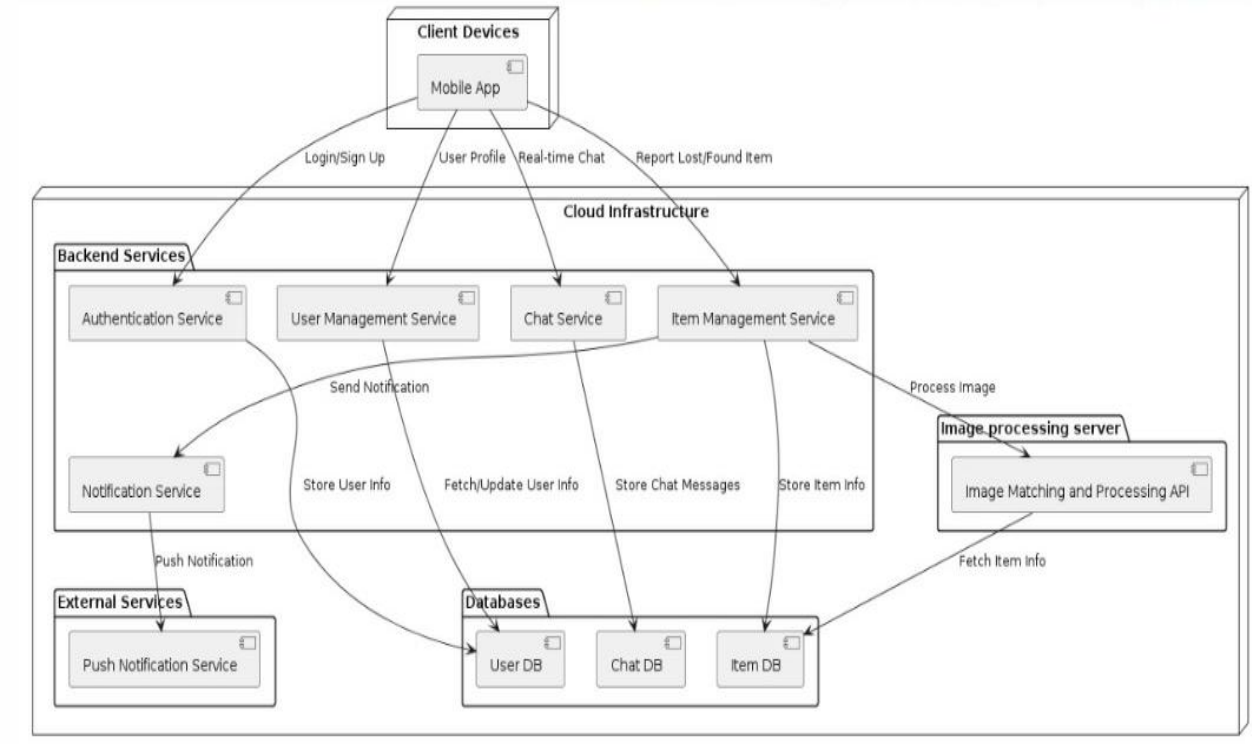

Fig2.6. Deployment diagram Design for a mobile based application for archival and retrieval of lost items using image matching, showing all software modules and the hardware units on which they are deployed as well as the communications and interactions between them

As seen above, This deployment diagram provides a high-level view of how the system components interact and are deployed across a cloud infrastructure, ensuring scalability and reliability.

3. BENEFITS OF GOOD SYSTEM DESIGN AND MODELLING

The design phase is pivotal to the success of the "Mobile-Based Archival and Retrieval of Missing Objects Application Using Image Matching" project for several reasons:

i) Clarity and Organization: The design phase provides a clear and organized blueprint of the system, outlining how various components will interact. This clarity is essential for ensuring that all team members have a shared understanding of the system architecture and functionality, reducing miscommunication and aligning development efforts.

ii) Requirement Validation: By translating user requirements into detailed diagrams and models, the design phase helps validate that all specified needs are being addressed. This ensures that the final system will meet user expectations and provides an opportunity to identify and resolve any discrepancies or gaps in the requirements early on.

iii) Risk Mitigation: Detailed design allows for the identification and mitigation of potential risks before actual development begins. This proactive approach helps in anticipating issues related to scalability, security, and performance, thereby avoiding costly fixes later in the development process.

iv) Facilitation of Development: A well-documented design serves as a guide for developers, providing them with the necessary details to implement the system effectively. This facilitates a smoother development process, as developers can follow the predefined structure and interactions, reducing the likelihood of errors and inconsistencies.

v) Enhancing Communication: The design phase produces various diagrams (such as context, use case, class, sequence, and deployment diagrams) that serve as visual tools for communication among stakeholders, including developers, designers, project managers, and clients. These visuals help in conveying complex ideas more effectively and ensuring that everyone is on the same page.

vi) Basis for Testing: The design phase lays the groundwork for creating test cases and scenarios. By understanding the system's architecture and interactions, testers can develop comprehensive test plans that cover all aspects of the system, ensuring thorough validation and verification of the final product.

vii) Scalability and Maintenance: Designing the system with scalability and maintainability in mind ensures that the application can handle growth in user base and data volume. Additionally, a well-designed system is easier to maintain and extend, facilitating future updates and enhancements without significant rework.

viii) Cost and Time Efficiency: Investing time in the design phase can save both time and money in the long run. By addressing potential issues and refining the system's structure before development, the project can avoid costly changes and delays during later stages, leading to a more efficient development process.

4. CONCLUSION

In conclusion, the System Modeling and Design phase is instrumental in shaping the success of the "Mobile-Based Archival and Retrieval of Missing Objects Application Using Image Matching" project. This phase provides a comprehensive blueprint that guides the development process, ensuring clarity, organization, and alignment with user requirements. By addressing potential risks, validating requirements, and facilitating clear communication among stakeholders, the design phase mitigates issues that could arise during development and sets a solid foundation for building a robust, scalable, and maintainable system.

The skills acquired during this phase are invaluable for any software development endeavor. Proficiency in creating architecture, context, use case, class, sequence, and deployment diagrams enhances one's ability to translate complex requirements into actionable designs. This phase also hones analytical and problem-solving skills, as it involves anticipating and addressing potential challenges before they become critical issues. Additionally, effective communication and documentation skills are developed, which are crucial for ensuring that all team members and stakeholders have a unified understanding of the system's design. Overall, the knowledge and expertise gained during the System Modeling and Design phase are critical for successful project execution and future endeavors in software development.

REFERENCES

1. https://www.geeksforgeeks.org/use-case-diagram/ date: 27/05/2024

2. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/ date: 27/05/2024

3. https://venngage.com/blog/context-diagram/ date: 27/05/2024

4. https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/ date: 27/05/2024

5. https://agilemodeling.com/artifacts/deploymentdiagram.htm date: 27/05/2024

6. https://aws.amazon.com/what-is/architecture-diagramming/ date: 27/05/2024