# UNIVERSITY OF BUEA

**FACULTY OF ENGENEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGENERING**

**System Modeling Report: Archival and Retrieval of Missing Object App**

## CEF440: INTERNET PROGRAMMING AND MOBILE PROGRAMMING

**TASK 4: REPORT**

|  |  |
| --- | --- |
| **NAMES** | **MATRICULES** |
| NSEM CONFIDENT NJOCK | FE21A285 |
| NZEMTEJUH SYLVANUS | FE21A296 |
| NKWENKAM JENNIFER | FE21A279 |
| VTALAH DE GERAUD | FE21A329 |
| OLETIA AJONGAKUE SYNCLAIR | FE21A190 |

Supervisor: Dr Nkemeni Valery

**Table of content**

**[System Modeling Report: Archival and Retrieval of Missing Object App 2](#_Toc18967)**

**[Introduction 3](#_Toc20001)**

**[Literature Review 4](#_Toc8833)**

[Existing Systems 4](#_Toc25045)

**[System Design 5](#_Toc5295)**

**[1. Class diagram 6](#_Toc30483)**

**[2. Context diagram 8](#_Toc14015)**

[Interactions: 8](#_Toc31775)

[Flow Description: 9](#_Toc13316)

**[3. Usecase diagram 9](#_Toc29877)**

**[4. Sequence diagram 11](#_Toc14386)**

**[5. ACTIVITY DIAGRAM 13](#_Toc24828)**

**[6. DEPLOYMENT DIAGRAM 17](#_Toc16655)**

**[1. Mobile Client (Flutter App): 17](#_Toc13552)**

**[2. Server: 17](#_Toc29361)**

**[3. Cloud Storage: 18](#_Toc11317)**

**[4. External Services (if any): 18](#_Toc10884)**

**[1. Mobile Client (Flutter App): 18](#_Toc25974)**

**[2. Server: 19](#_Toc12210)**

**[3. Cloud Storage: 19](#_Toc18872)**

**[4. External Services: 19](#_Toc5031)**

# Introduction

### Project Overview

The Archival and Retrieval of Missing Object App is designed to provide an efficient solution for locating and retrieving lost items using image matching algorithm. The app leverages advanced image matching algorithms, such as Speeded-Up Robust Features (SURF), to compare user-uploaded images of missing objects with a database of archived items. This innovative approach aims to streamline the process of finding lost belongings by utilizing mobile technology, making it accessible and convenient for users. The app supports both Android and iOS platforms, ensuring broad usability.

### Objectives

The primary objectives of the Archival and Retrieval of Missing Object App project are as follows:

* Develop a User-Friendly Mobile Application: Create an intuitive and easy-to-use interface that allows users to archive and search for missing objects efficiently.
* Implement Advanced Image Matching Algorithms: Utilize algorithms like SURF to accurately match images of missing objects with archived images in the database.
* Ensure Reliable Data Storage and Retrieval: Design a robust backend system that securely stores object images and metadata, facilitating quick and accurate retrieval.
* Provide Real-Time Notifications: Implement a notification system to inform users immediately when a match for a missing object is found.
* Support Multilingual Capabilities: Enable the app to support multiple languages, making it accessible to a diverse user base.

### Scope

The scope of the Archival and Retrieval of Missing Object App project includes the following:

* User Registration and Authentication: Develop a secure system for user registration, login, and authentication to protect user data.
* Object Archival System: Implement functionality for users to upload images and details of objects they want to archive.
* Image Matching and Retrieval: Integrate SURF or similar image matching algorithms to compare uploaded images with archived objects.
* Notification System: Create a real-time notification system to alert users when a matching object is found.
* Database Design: Develop a scalable database to store object images, metadata, and user information.
* User Interface Design: Design a user-friendly interface that is consistent across Android and iOS platforms.
* Testing and Evaluation: Conduct comprehensive testing to ensure the app functions correctly and meets user needs.
* Documentation and Support: Provide thorough documentation and customer support to assist users in navigating the app.

By adhering to this scope, the project aims to deliver a reliable and efficient solution for the archival and retrieval of missing objects, leveraging modern technology to address a common problem faced by many users.

# Literature Review

## Existing Systems

Several existing systems and solutions aim to address the problem of locating and retrieving missing objects. These systems generally fall into two categories: manual reporting systems and automated image recognition systems.

#### Manual Reporting Systems

Manual reporting systems, such as lost-and-found services provided by institutions or websites like Craigslist, rely on users to post descriptions and images of lost or found items. These systems have the following strengths and weaknesses:

##### Strengths:

* Wide Reach:Platforms like Craigslist have a large user base, increasing the chances of lost items being reported.
* Community Involvement: Users can contribute by reporting found items, fostering a sense of community.

##### Weaknesses:

* Inefficiency: Manually browsing through listings can be time-consuming and cumbersome.
* Human Error: Descriptions and images provided by users can be inconsistent or inaccurate, making it difficult to match lost and found items.

#### Automated Image Recognition Systems

Automated image recognition systems, such as the "Find My" app by Apple or apps like Stolen Camera Finder, use technology to match lost items with a database of found items based on visual characteristics or metadata.

##### Strengths:

* Efficiency: Automated systems can quickly process and match images, saving time for users.
* Accuracy: Advanced algorithms can detect similarities between images more accurately than human comparison.

##### Weaknesses:

* Limited Database: The effectiveness of these systems depends on the size and quality of the database of found items.
* Privacy Concerns: Storing and processing user images raises privacy and security issues that need to be addressed.

# 

# System Design

System modeling is a critical aspect of developing the Archival and Retrieval of Missing Object App, as it provides a structured representation of the system's components, their interactions, and the overall architecture. By creating detailed models, we can ensure that the system is designed efficiently, meets user requirements, and functions reliably.

### 

### Class diagram

### class d

Figure 1: class diagram of an archival and retrieval of missing object app

The proposed class diagram includes five main classes: AppUser, MissingObject, Image, MatchingAlgorithm, and Search. Here is a summary of each class, their attributes, methods, and relationships:

* AppUser
  + Attributes:

userId, name, email, password, phoneNumber

* Methods:

register(), login(), updateProfile(), viewProfile()

* Relationships:

Association (One-to-Many)with MissingObject: A user can report multiple missing objects.

* MissingObject
  + Attributes:

objectId, title, description, dateLost, locationLost, status

* Methods:

reportMissing(), updateStatus(), viewDetails()

* Relationships:

Composition (One-to-Many) with Image: A missing object can have multiple images that are dependent on it.

Aggregation (Many-to-One) with AppUser: Multiple missing objects can be reported by a single user.

* Image
  + Attributes:

imageId, imagePath, uploadDate

* Methods:

uploadImage(), deleteImage(), viewImage()

* Relationships:

Composition with MissingObject: Images are an integral part of the missing object.

Association (Many-to-Many) with MatchingAlgorithm: The algorithm can match images from different missing objects.

* MatchingAlgorithm
  + Attributes:

algorithmId, algorithmName, algorithmType

* Methods:

matchImage(image1, image2): Boolean, configureParameters(params: Map<String, String>)

* Relationships:

Association with Image: The algorithm uses images to perform matching.

* Search
  + Attributes:

searchId, searchDate, searchResults

* Methods:

performSearch(image: Image): List<MissingObject>, viewSearchResults()

* Relationships:

Aggregation (Many-to-One) with Image: A search can involve multiple images.

Association (Many-to-One) with AppUser: A user can perform multiple searches.

This design ensures modularity and clarity, making the system easy to maintain and extend.

### Context diagram

This diagram is a context diagram for the "Find Missing Object App". It shows the interactions between the system and external entities (users and admin). Here's a detailed explanation of each part:

### Context Diagram Explanation

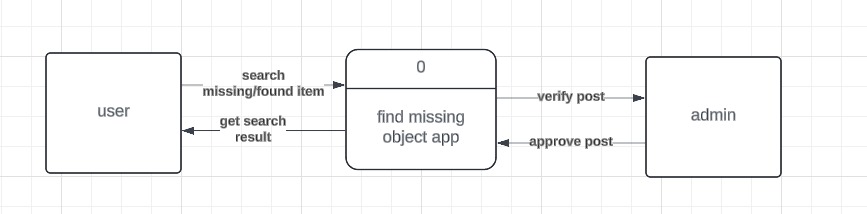


Figure 2: context diagram of an archival and retrieval of missing object.

* External Entities:
  + User: Represents the end-users who interact with the application to search for missing or found items.
  + Admin: Represents the administrators who manage the application, including verifying and approving posts made by users.
* System:
  + Find Missing Object App: The system or application that facilitates the process of searching for and posting about missing or found items.

**Interactions:**

* User Interactions:
  + Search Missing/Found Item: Users can search the application for items that are missing or have been found.
  + Get Search Result: The application returns the results of the user's search, displaying relevant posts about missing or found items.
* Admin Interactions:
  + Verify Post: Admins receive posts from the application that need to be verified. This process involves checking the validity and accuracy of the posts.
  + Approve Post: After verification, admins approve the post, making it available to users in the search results.

**Flow Description:**

* User Searches for Items:
  + The user interacts with the "Find Missing Object App" to search for missing or found items.
  + The application processes the search query and returns the relevant search results to the user.
* Admin Verifies and Approves Posts:
  + Posts created by users about missing or found items are sent to the admin for verification.
  + The admin verifies the posts and, if they meet the necessary criteria, approves them.
  + Approved posts are then available to users for search results and other interactions.

This context diagram provides a high-level overview of how the "Find Missing Object App" interacts with its users and admins, defining the boundaries and external interactions of the system.

### Usecase diagram

### usecase

Figure 3: usecase diagram of an archival and retrieval of missing object.

This diagram is a use case diagram for the "Archive and Retrieve" application. It shows the interactions between different actors (users and admin) and the use cases (functionalities) of the system. Here’s a detailed explanation:

* Actors:
  + new: Represents new users who are interacting with the system for the first time.
  + old: Represents returning users who have previously used the system.
  + endUser: Represents the general user of the system, encompassing both new and old users.
  + admin: Represents the administrator who manages and oversees the system's operations.
* System:

Archive and Retrieve: The system or application that facilitates the process of managing, searching, and retrieving information about missing and found items.

#### Use Cases:

* End User Use Cases:
  + login: Allows users to log in to the application.
  + contact\_CS: Enables users to contact customer support for assistance.
  + lost: Allows users to report a lost item.
  + found: Allows users to report a found item.
  + upload: Enables users to upload information or images related to lost or found items.
  + search: Allows users to search for missing or found items in the application.
  + edit\_image: Users can edit images they have uploaded.
  + registration: Allows new users to register and create an account.
  + view\_history: Enables users to view their history of interactions with the application.
  + view\_notification: Allows users to view notifications related to their posts or account.
  + contact\_authorities: Enables users to contact relevant authorities regarding lost or found items.
* Admin Use Cases:
  + view\_activities: Admins can view the activities of users on the application.
  + delete\_account: Admins can delete user accounts.
  + delete\_image: Admins can delete images uploaded by users.
  + manage\_permission: Admins can manage user permissions and roles within the application.

#### Usecase Description:

* End User Interactions:
  + Login and Registration: Users (new and old) can log in or register to use the application.
  + Report Lost/Found Items: Users can report lost or found items, upload related information, and search for items.
  + Manage Content: Users can edit images, view their interaction history, and receive notifications.
  + Contact Support/Authorities: Users can contact customer support or relevant authorities for assistance.
* Admin Interactions:
  + View User Activities: Admins can monitor user activities on the platform.
  + Manage User Accounts: Admins have the authority to delete user accounts and images.
  + Permission Management: Admins can manage user permissions, ensuring proper access control within the application.

This use case diagram provides a clear understanding of the roles and functionalities within the "Archive and Retrieve" application, highlighting the interactions between users, admins, and the system.

### Sequence diagram

Sequence Diagram

In the context of a mobile-based application for archival and retrieval of missing objects using image matching, a sequence diagram represents the step-by-step interactions and flow of events between the different components of the system.

The key elements of the sequence diagram for this application include:

* Actors/Participants:
  + User: The person using the mobile application to archive or retrieve missing objects.
  + MobileApp: The mobile application that provides the user interface and coordinates the different functionalities.
  + ImageCapture: The component responsible for capturing images of the missing object.
  + ImageProcessing: The component that processes the captured images and extracts visual features.
  + ObjectArchiveServer: The server-side component that stores and retrieves information about archived missing objects.
  + ObjectRetrieval: The component responsible for providing the details to retrieve a missing object from the archive.
* Lifelines:

Each of the above participants have a lifeline representing their involvement in the sequence of events.

* Messages:

The flow of messages between the participants, such as the user opening the app, the app requesting image capture, the user capturing images, the app sending images for processing, the server searching the archive, the app displaying the results, and so on.

* Time:

The vertical axis of the diagram represents the chronological order of the interactions, from top to bottom.

The sequence diagram for this application would typically include the following key steps:

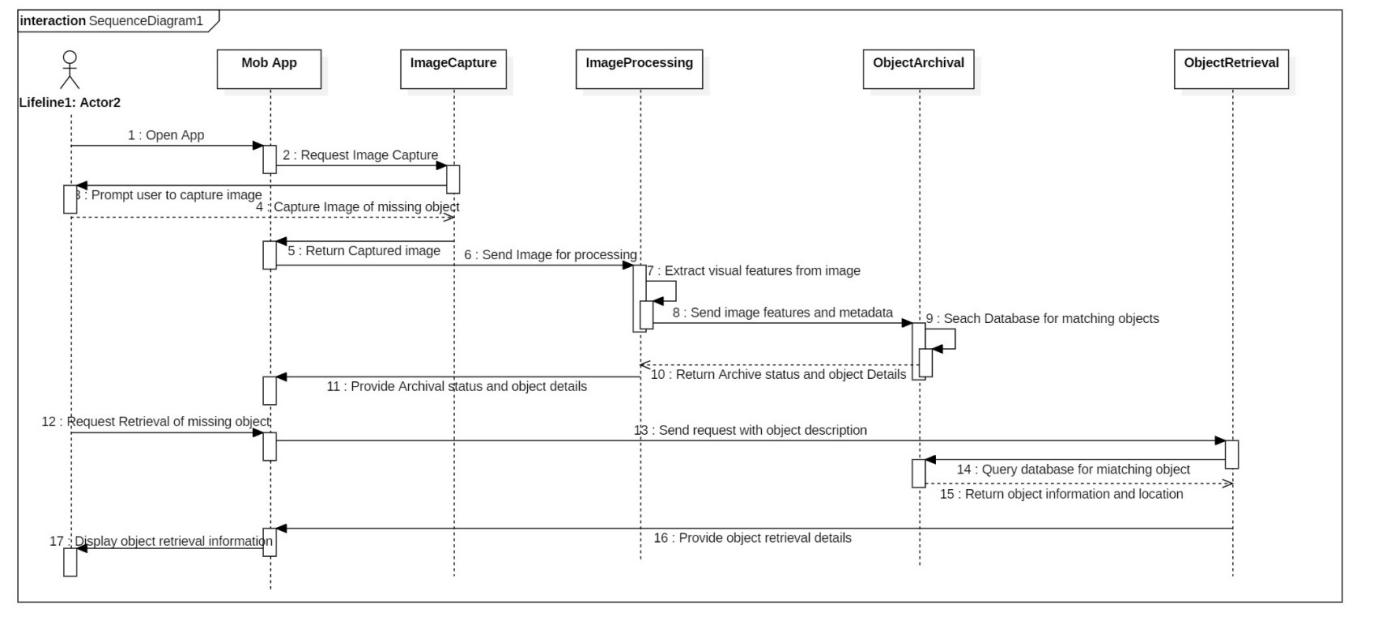


Figure 4: sequence diagram of an archival and retrieval of missing object app

**Explanation**

* The user opens the mobile application.
* The mobile app requests the image capture module to prompt the user to capture images of the missing object.
* The user captures the images and the image capture module sends them back to the mobile app.
* The mobile app forwards the captured images to the image processing module.
* The image processing module extracts visual features from the images.
* The image processing module sends the image features and metadata to the object archive server.
* The object archive server searches its database for matching objects.
* The object archive server returns the archival status and any available object details to the image processing module.
* The image processing module provides the archival status and object details to the mobile app.
* If the object is not found in the archive, the mobile app requests the object archive server to archive the new object.
* The object archive server stores the new object in the database.
* The object archive server confirms the object has been archived to the mobile app.
* If the object is found in the archive, the user requests the mobile app to retrieve the missing object.
* The mobile app sends the retrieval request with the object details to the object retrieval module.
* The object retrieval module queries the object archive server for the object's location.
* The object archive server returns the object's location to the object retrieval module.
* The object retrieval module provides the object retrieval details to the mobile app.
* The mobile app displays the object archival or retrieval information to the user.

This sequence diagram helps visualize the interactions between the different components of our mobile-based missing object archival and retrieval system, making it easier to understand the flow of the application and identify potential issues or areas for improvement.

### ACTIVITY DIAGRAM

1. **Primary user Activity diagram**

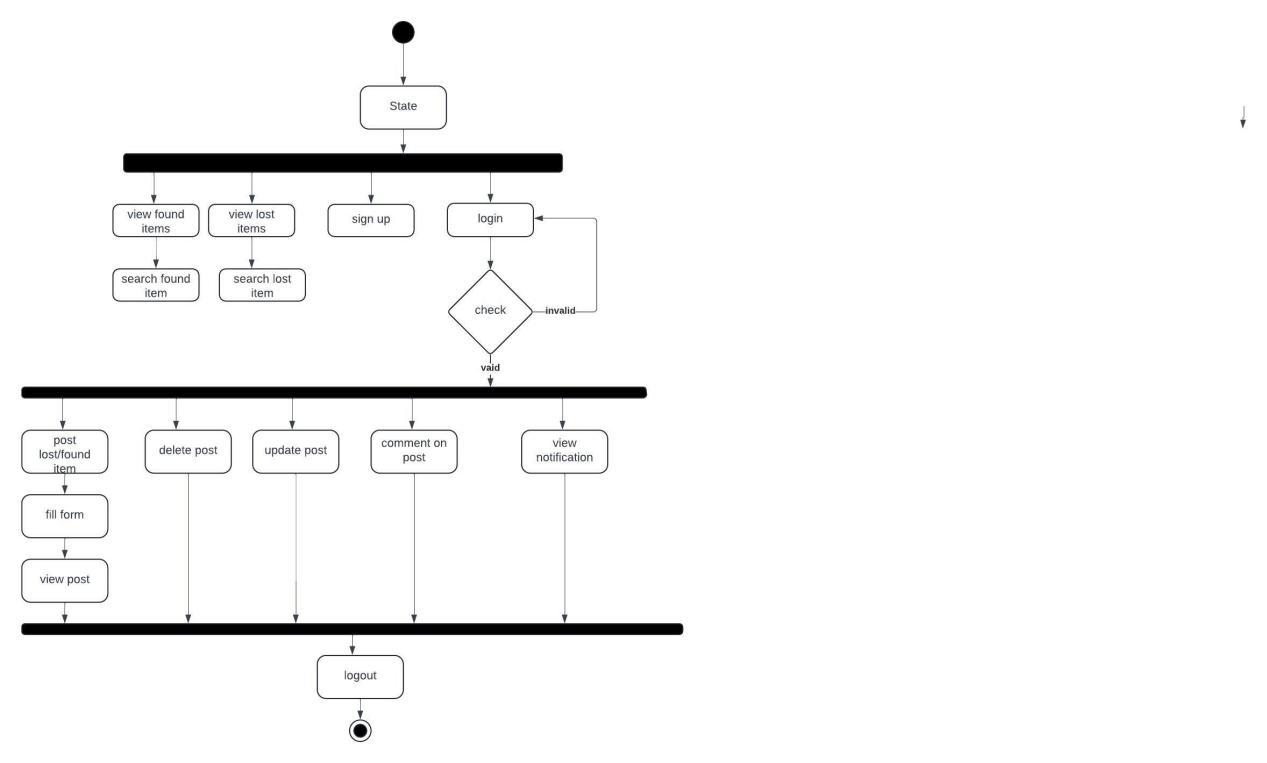


Figure 5: activity diagram of users in the archival and retrieval of missing object app

The diagram illustrates the flow and states of an archival and retrieval application for missing objects.

**Explanation**

* Top Section: State
  + Represents the initial state of the application.
* Middle Section: Navigation and actions
  + View Found Items: Allows the user to view a list of items that have been found.
  + View Lost Items: Allows the user to view a list of items that have been reported as lost.
  + Search Found Item: Enables the user to search through the found items using specific criteria.
  + Search Lost Item: Enables the user to search through the lost items using specific criteria.
  + Sign Up: Provides an option for new users to create an account.
  + Login: Allows existing users to log in to the application.
  + Check: Validates user credentials during login. If valid, proceeds to the next state. If invalid, prompts the user to try again.
* Bottom Section: Post Action
  + Post Lost/Found Item
  + Fill Form: Users fill out a form with details about the lost or found item.
  + View Post: Displays the submitted post with item details.
  + Delete Post: Users can delete their own posts about lost or found items.
  + Update Post: Users can update the details of their existing posts.
  + Comment on Post: Allows users to comment on posts, which can help in providing additional information or asking questions.
  + View Notification: Users can view notifications related to their posts or comments.
* Logout: Logs the user out of the application, returning them to the initial state.

**Flow Description**

* Initial State: The application starts from the "State" node.
* User Options: From the initial state, the user can either view found items, view lost items, search for found items, search for lost items, sign up, or log in.
* Authentication: If the user chooses to log in, their credentials are checked. If valid, they proceed to the main application functionalities. If invalid, they are prompted to log in again.
* Main Functionalities:
  + Users can post details about lost or found items by filling out a form and viewing their post.
  + Users can manage their posts by updating or deleting them.
  + Users can interact with other posts by commenting.
  + Notifications can be viewed for updates related to the user's posts or interactions.
* Logout: The user can log out at any time, returning the application to the initial state.

This structure ensures a clear and organized flow for users interacting with the archival and retrieval system, making it easy to manage and retrieve information about lost and found items.

1. **Secondary(admin) user activity diagram**

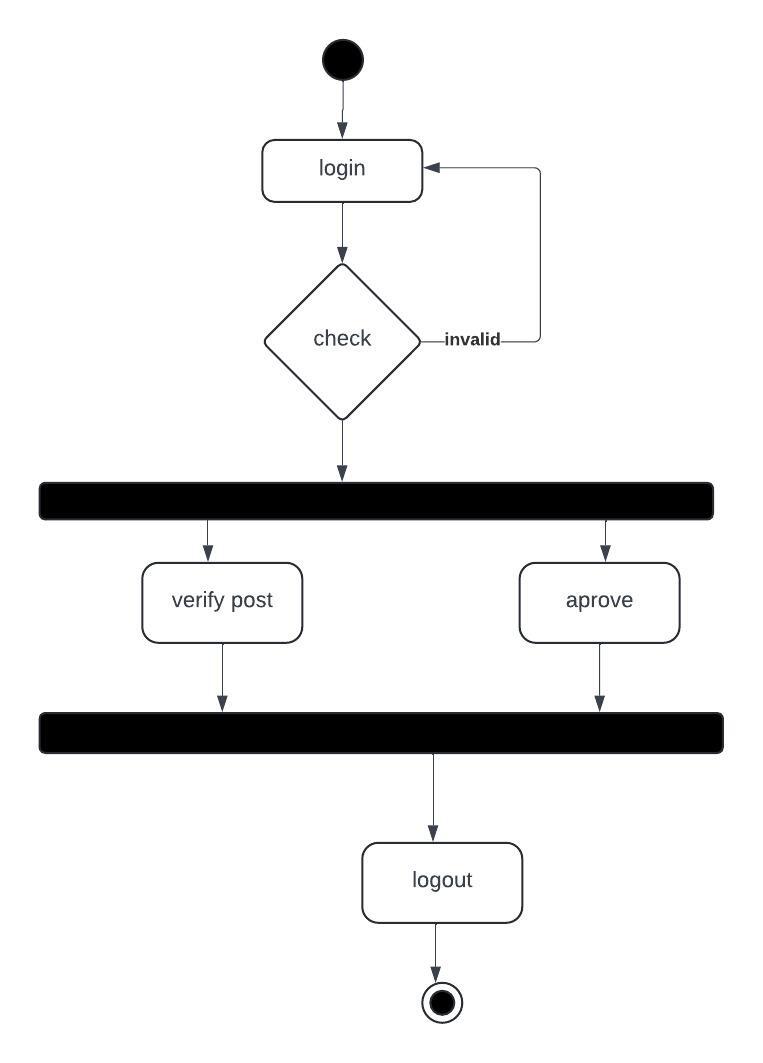


Figure 6: activity diagram of an admin in the archival and retrieval of missing object app

The diagram illustrates the flow and states of the archival and retrieval application specifically for an admin role. Here’s a detailed explanation:

* Top Section(Initial State): Represents the initial state of the admin panel of the application.
* Middle Section(Login Process):
  + Login: The admin starts by logging into the application.
  + Check: The login credentials are verified. If valid, the admin proceeds to the main functionalities. If invalid, the admin is prompted to log in again.
* Bottom Section(Admin Actions):
  + Verify Post: Admins can verify posts made by users about lost or found items. Verification ensures the authenticity and correctness of the information before it is published.
  + Approve: After verification, the admin can approve the post to be visible to all users. This step might involve checking the details, confirming the legitimacy of the item, and ensuring it meets the platform’s guidelines.
* Logout: The admin can log out of the application, returning to the initial state.

**Flow Description**

* Initial State: The application starts from the initial state.
* Admin Login: The admin logs in to the system.
* Check Credentials: The admin's login credentials are checked. If valid, the admin proceeds. If invalid, the admin is prompted to try logging in again.
* Main Functionalities:
  + Verify Post: The admin can review posts submitted by users about lost or found items. This may involve checking the details provided, cross-referencing with other reports, and ensuring the post is genuine.
  + Approve Post: After verification, the admin can approve the post, making it visible to all users on the platform.
* Logout: The admin can log out at any time, returning the application to the initial state.

This structure ensures a controlled and secure flow for admins managing the archival and retrieval system, allowing them to oversee the integrity and accuracy of the information being shared on the platform.

### DEPLOYMENT DIAGRAM

A deployment diagram for an archival and retrieval of missing objects mobile application using image matching algorithms like SURF and developed in Flutter would typically include the following components:

1. Mobile Client (Flutter App):

* User Interface
* Image Capture Module
* Image Processing Module
* Network Communication Module

2. Server:

* Web Server
* REST API
* Image Matching Algorithm (e.g., SURF)
* Database Server (to store image data and metadata)
* Authentication Server (for user authentication)

3. Cloud Storage:

* Image Storage
* Metadata Storage

4. External Services (if any):

* Push Notification Service
* External APIs (if the application uses any external image databases or services)

Here is a simplified deployment diagram:

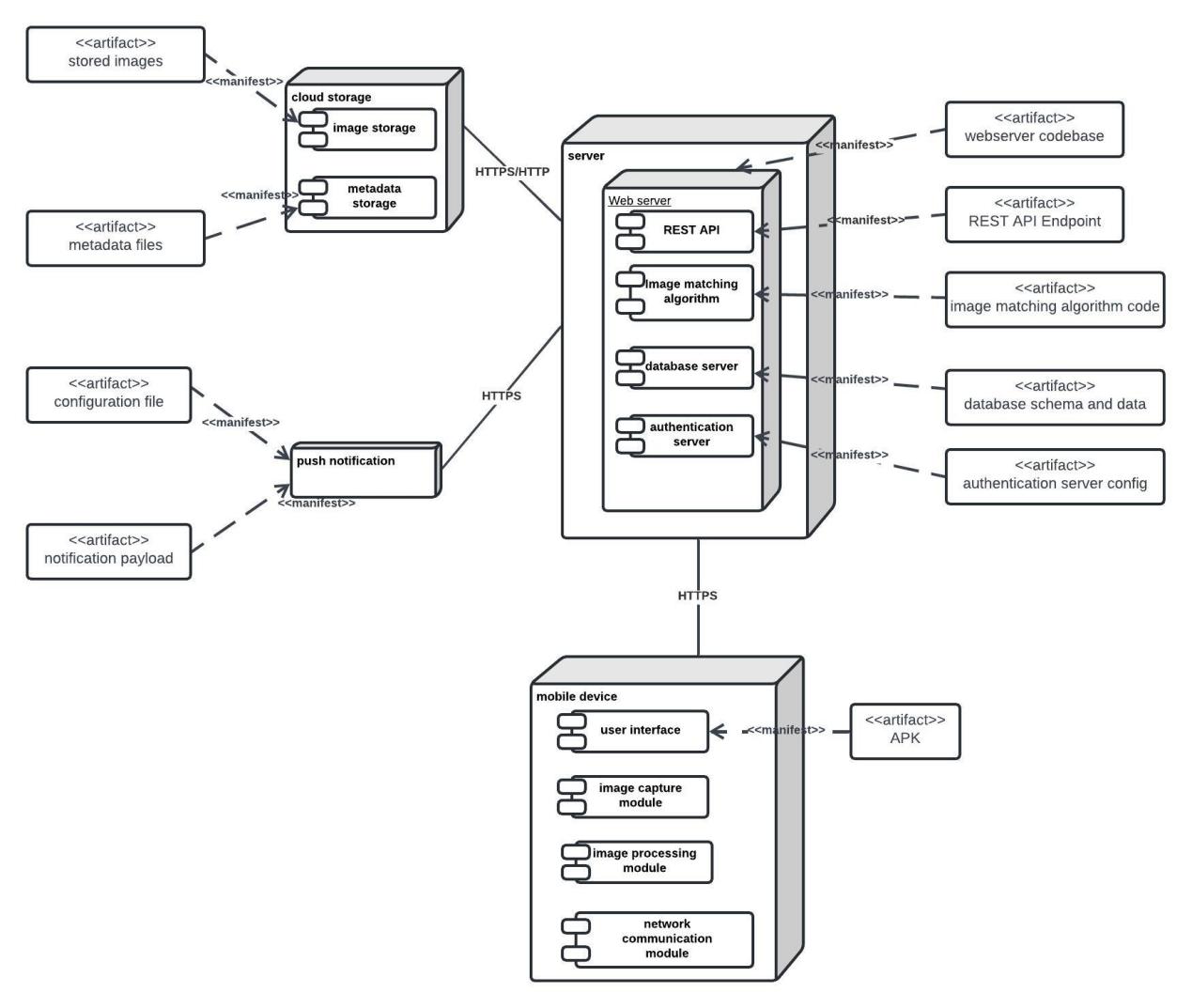


Figure 7: deployment diagram diagram of an archival and retrieval of missing object app

### Explanation:

1. Mobile Client (Flutter App):

* User Interface: Front-end where users interact with the app.
* Image Capture Module: Allows users to capture images of missing objects.
* Image Processing Module: Pre-processes images before sending to the server.
* Network Communication Module: Manages data transfer between the client and server.

2. Server:

* Web Server: Handles incoming requests from the mobile client.
* REST API: Provides endpoints for various functionalities (image upload, retrieval, user management, etc.).
* Image Matching Algorithm: Processes images using algorithms like SURF match and retrieve similar images.
* Database Server: Stores image data, metadata, and user information. Authentication Server: Manages user authentication and security.

3. Cloud Storage:

* Image Storage: Stores the actual images uploaded by users.
* Metadata Storage: Stores metadata related to images (e.g., description, tags, timestamps).

4. External Services:

* Push Notification Service: Sends notifications to users (e.g., when a matching image is found).
* External APIs/Services: Interacts with other image databases or services if required.

This deployment diagram provides a high-level overview of the components involved in the archival and retrieval of missing objects mobile application and their interactions.

# Conclusion

# This report has detailed the development process of an archival and retrieval app for missing objects, emphasizing the critical role of UML diagrams in system modeling and design.

# The context diagram provided an overarching view of the system's interactions with external entities, laying the foundation for understanding the app's environment. The use case diagram identified key functionalities and interactions between users and the system, ensuring all essential features were captured.

# The class diagram depicted the app's structural framework, defining the relationships and interactions between different objects within the system. This was crucial for organizing and managing data effectively. The deployment diagram illustrated the hardware and software architecture, ensuring a seamless and efficient deployment across various platforms.

# Activity diagrams mapped out the workflows and processes, enabling a clear visualization of the system's operations and enhancing process optimization. Sequence diagrams further detailed the interactions between objects over time, providing a dynamic perspective on the app's functionality and aiding in the identification of potential issues early in the development process.

# In conclusion, the use of UML diagrams was instrumental in the systematic and efficient development of the archival and retrieval app. These diagrams facilitated clear communication, thorough planning, and robust design, ultimately contributing to a well-structured and functional application. This approach underscores the importance of UML in developing complex systems and sets a solid groundwork for future enhancements.

# Reference

1. Jayed Hossain. (2019, May 10). uml diagrams for an archival and retrieval of missing object app. <https://www.slideshare.net/slideshow/lost-and-found-web-project/144729773>