THE UNIVERSITY OF BUEA

P.o box 63

Buea, South West Region

Cameroon

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

REPUBLIC OF CAMEROON

PEACE - WORK - FATHERLAND

CEF440:INTERNET PROGRAMMING (J2EE) AND MOBILE PROGRAMMING

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

Group 28: members: TASK 4

Names	Matricule
TCHUIDJAN JORDAN BRYANT	FE21A320
FUKA NEVILLE TENYI	FE21A199
NFOR WILLY LINWE	FE21A254
FRU BELTMOND CHINJE	FE21A198
EYONG OSCAR ENOWNYUO	FE21A187

COURSE INSTRUCTOR:
Dr. NKEMENI VALERY

ACADEMIC YEAR 2023-2024

Table of contents

1. Introduction

System modeling is a process of creating a simplified representation of a system to understand its structure, behavior, and interactions. It involves using various graphical and textual notations to describe the components, relationships, and functions of a system. System modeling helps in visualizing and analyzing complex systems, facilitating communication among stakeholders, and making informed decisions during the system development lifecycle.

System design refers to the process of defining and specifying the architecture, components, interfaces, and behavior of a system to meet specific requirements and objectives. It involves transforming the system requirements gathered during the analysis phase into a detailed blueprint that guides the implementation and construction of the system.

2. Modeling

A. Why is system modeling and Design important

System modeling and design are particularly important for a mobile app retrieval and searching for objects using image matching for the following reasons:

1. **Clarity of Requirements**:System modeling helps in clearly defining the requirements of the mobile app. It allows stakeholders to visualize and articulate the specific functionalities and interactions expected from the app, such as image capture, searching by image, and browsing search results. This clarity ensures that the app development aligns with the intended objectives.

2. **Architecture and Component Design:** System modeling enables the design of a well-structured architecture for the mobile app. It helps in identifying the key components, such as the user interface, image processing module, retrieval engine, and external interactions (e.g., database server or cloud services). By defining the architecture and component interactions, system modeling ensures a robust and efficient design.

3. **Behavior and Flow Visualization: System modeling allows for visualizing the behavior and flow of the mobile app. Through sequence diagrams or use case diagrams, stakeholders can understand how different functionalities, such as image capture, preprocessing, feature extraction, and image matching, interact with each other. This visualization aids in identifying potential bottlenecks or areas for optimization.

4. Data Representation and Handling:System modeling helps in designing the representation and handling of data within the mobile app. It allows for defining the data structures used to store images, extracted features, and other relevant information. Modeling data relationships and associations enable efficient data retrieval and matching processes.

5. Scalability and Deployment Planning: System modeling facilitates planning for scalability and deployment of the mobile app. By considering factors like increased user traffic or larger image databases, the modeling process helps identify potential scalability challenges. It allows for exploring cloud-based solutions, load balancing mechanisms, or distributed computing to support the app's scalability and performance.

Overall, system modeling and design provide a structured and systematic approach to ensure that the mobile app retrieval and object search using image matching meet the desired requirements, exhibit efficient performance, and provide a seamless user experience. It enables stakeholders to collaborate effectively, make informed decisions, and deliver a well-designed and functional mobile app.

The various diagrams were implemented to model our app functionality

a. Usecase diagram
b. Context diagram
c. Class diagram
d. Sequence diagram
e. Deployement diagram

A. Usecase diagram

A use case diagram is a graphical representation in UML (Unified Modeling Language) that depicts the interactions between actors (users or external systems) and a system under consideration. It provides a high-level overview of the functionalities or features of a system from the perspective of its users.

In the context of mobile app retrieval and searching for objects using image matching, here are some potential use cases that could be included:

1. **User Registration:** This use case involves the process of a user registering an account in the mobile app. It may include providing personal information, creating a username and password, and setting up a profile.

2. **User Login:** This use case represents the action of a user logging into the mobile app using their registered credentials.

3. **Search by Image:** This use case focuses on the functionality of searching for objects using image matching. The user can capture or upload an image and initiate a search based on that image. The system then matches the image against a database or algorithm to find similar or matching objects.

4. **Object Retrieval:** This use case describes the process of retrieving information about a specific object based on the search performed. The app displays relevant details such as the name, description, price, or other relevant information about the matched objects.

5. **Filter and Sorting:** This use case involves allowing users to apply filters or sorting options to refine their search results. Users can specify criteria such as price range, category, location, or other attributes to narrow down the search results.

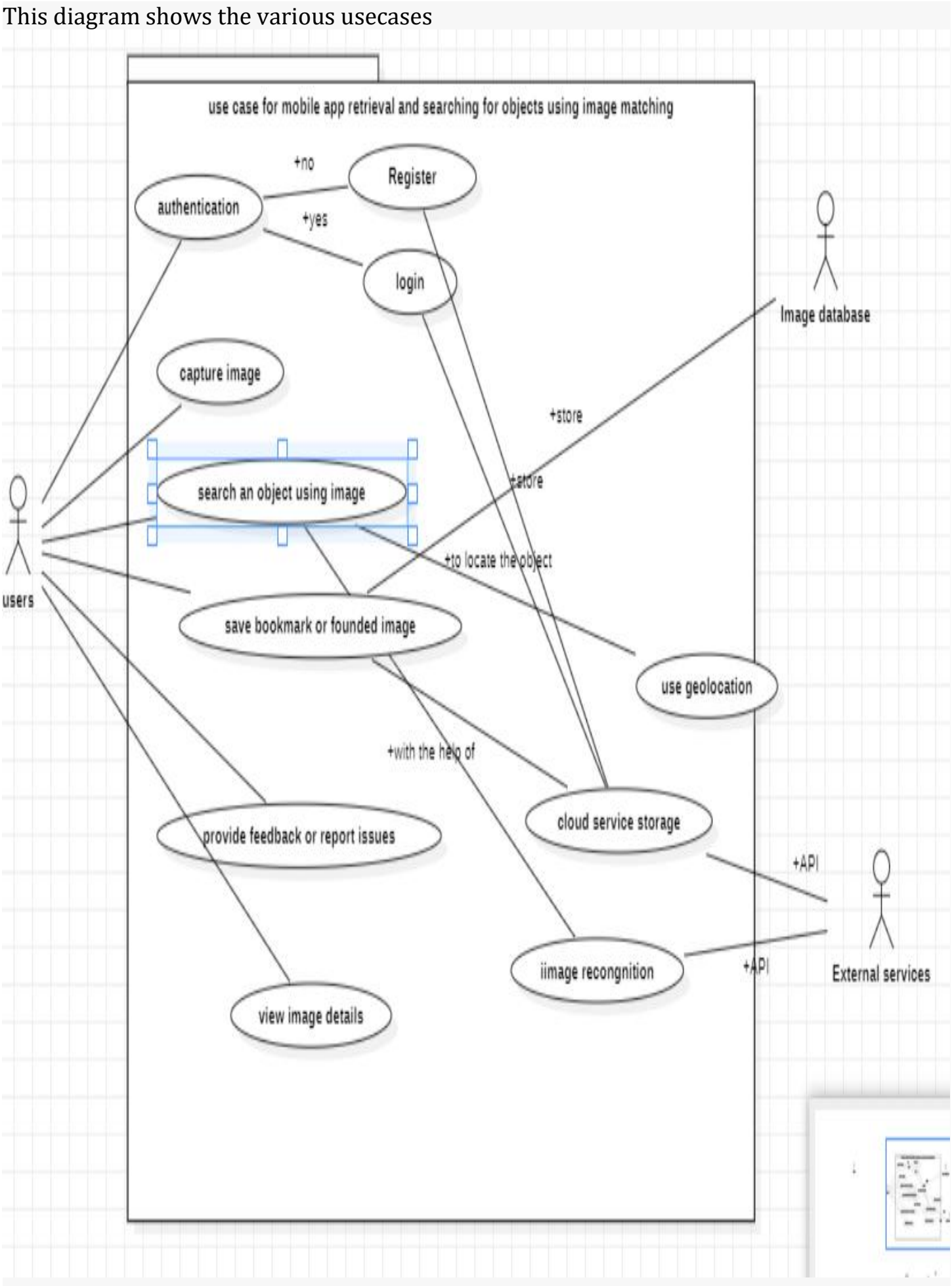
6. **Save Search Results:** This use case enables users to save or bookmark specific search results for future reference or comparison.

7. **User Feedback:** This use case allows users to provide feedback or ratings for the search results or the overall app experience. It helps improve the system and enhance user satisfaction.

8. **Notifications:** This use case involves sending notifications to users regarding new search results, updates, recommendations, or other relevant information.

These are just a few examples of potential use cases for mobile app retrieval and searching for objects using image matching. The specific use cases may vary depending on the requirements and functionality of the app being developed.

This diagram shows the various usecases



B. Context diagram

In the context diagram of a mobile app retrieval and searching for objects using image matching, the entities involved can include:

Mobile App: This entity represents the mobile application itself, which is the main focus of the system. It is the interface through which users interact with the app's functionalities, including image capture, searching by image, and viewing search results.

User: The user entity represents individuals who interact with the mobile app. Users can capture images, perform searches, and browse the search results. They initiate actions and provide input to the app.

Image Database: This entity represents the database or storage system where the images are stored. It contains a collection of images that the app uses for matching against user-captured images. The image database stores the images and associated metadata, such as object names, descriptions, and tags.

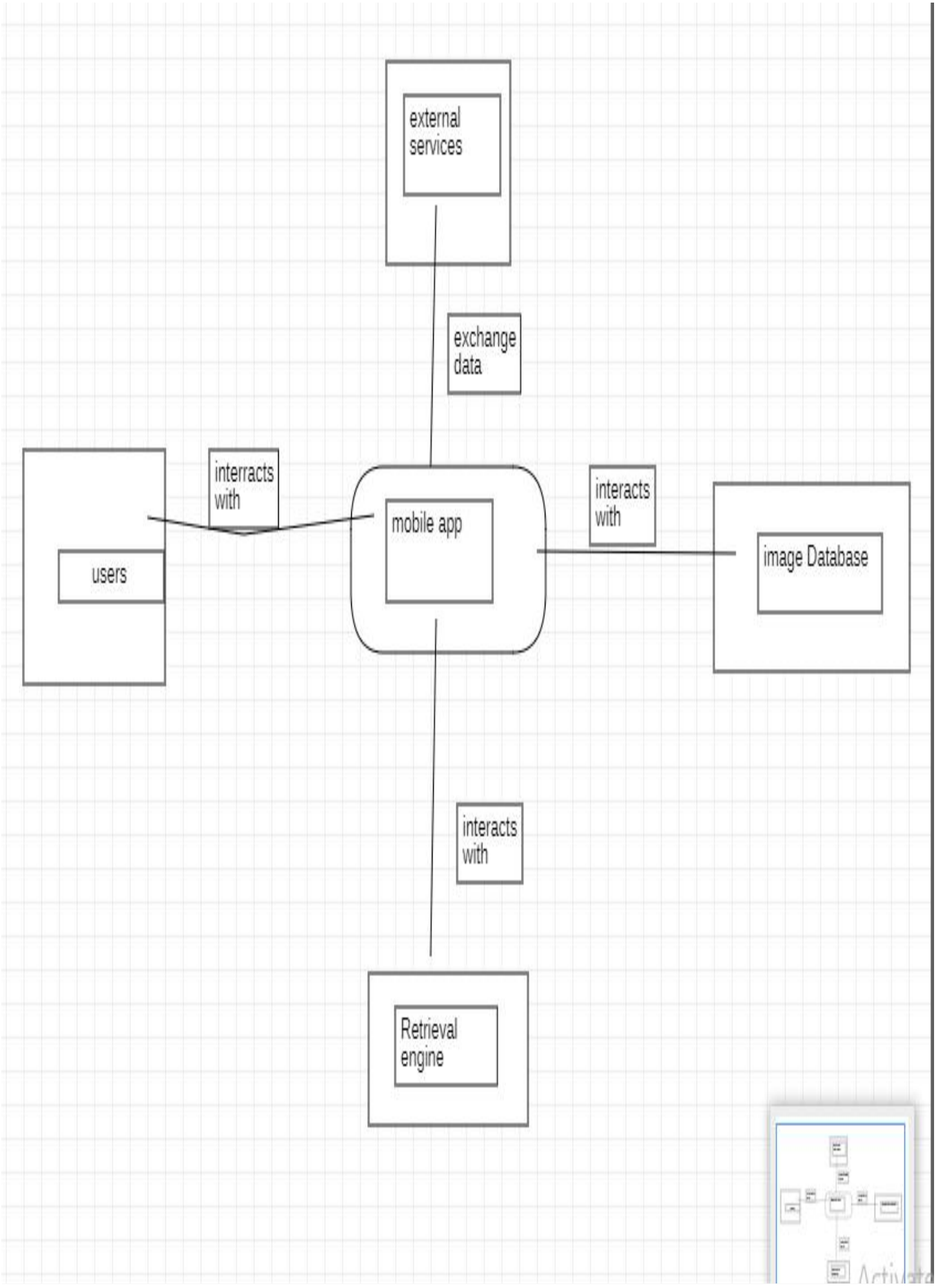
Image Processing Module: This entity represents the module within the mobile app responsible for processing the images captured by the user. It performs tasks like resizing, normalization, and noise reduction to enhance the quality of the captured images before further processing.

Retrieval Engine: The retrieval engine entity represents the core component within the app that performs the image matching and retrieval process. It compares the features extracted from user-captured images with the features stored in the image database, identifying relevant objects and returning the search results.

External Services: This entity represents any external services or APIs that the mobile app may interact with. For example, the app might integrate with cloud-based services for image recognition or utilize social media APIs for additional functionality.

Cloud Storage: This entity represents the cloud storage service that the app may use to store images or other data. It provides scalable and accessible storage for the app's data.

The context diagram provides a high-level view of the system and identifies the main entities and their relationships. It helps to establish the boundaries of the system and understand its external interactions. See the diagram below


external services
exchange data
interracts with
users
mobile app
interacts with
image Database
interacts with
Retrieval engine

C. Class diagram

To build a class diagram for a mobile app retrieval and searching system using image matching, several classes may be required. Here are some example classes that you can consider:

1. **MobileApp**: This class represents the mobile application itself and encapsulates the app's functionalities, user interface, and navigation flows.

2. **User:** The User class represents the app's users. It may contain attributes such as user ID, username, password, and methods for user authentication and account management.

3. **Image:** This class represents an image captured by the user. It may have attributes like image ID, file path, and metadata such as capture date and location. It can also include methods for image processing and feature extraction.

4. **Feature:** The Feature class represents the extracted features from an image. It may contain attributes or data structures to store feature information, such as feature vector or descriptors.

5. **ImageDatabase: T**his class represents the database or storage system that stores the images. It may have methods to store and retrieve images, search for images based on features, and manage image metadata.

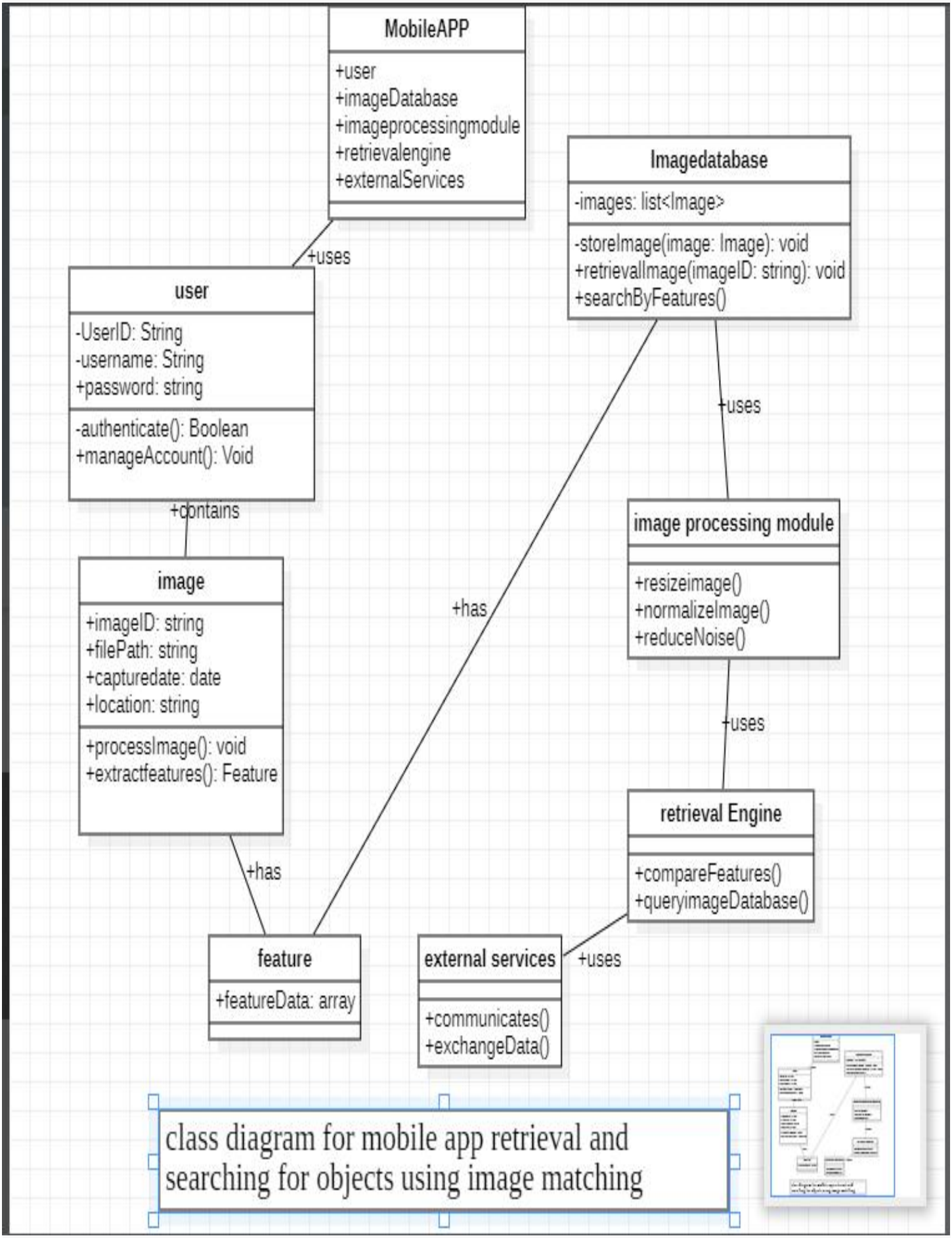
6. **ImageProcessingModule**: This class represents the module responsible for image processing tasks like resizing, normalization, and noise reduction. It may include methods for performing these operations on the captured images.

7. **RetrievalEngine:** The RetrievalEngine class represents the core component responsible for image matching and retrieval. It may contain methods for comparing features, querying the image database, and returning search results.

8. **ExternalServices:** This class represents external services or APIs that the app interacts with, such as cloud-based image recognition services or social media integration. It may have methods for communication and data exchange with these external services.

These are just example classes, and the actual classes required for your system may vary depending on your specific requirements and design. It's important to identify the key entities and their relationships to determine the appropriate classes for your class diagram.

The diagram belows shows the class diagram for our app and shows how each class interates with each other



D. Sequence diagram

A sequence diagram can illustrate the various sequences of interactions and messages exchanged between objects within the mobile app retrieval and searching system using image matching. Here are some sequences that can be represented on a sequence diagram:

1. **User Authentication Sequence:**
   - User sends authentication credentials (username and password) to the `MobileApp`.
   - `MobileApp` invokes the `authenticate()` method of the `User` object to verify the credentials.
   - The `User` object returns the authentication status (`true` or `false`) back to the `MobileApp`.
   - The `MobileApp` proceeds with app functionality if authentication is successful.

2. **Image Capture and Processing Sequence:**
   - User captures an image using the `MobileApp`.
   - `MobileApp` creates an instance of the `Image` object and passes the image data to it.
   - The `Image` object calls the `processImage()` method to perform preprocessing tasks like resizing, normalization, and noise reduction.
   - The `Image` object then calls the `extractFeatures()` method to extract features from the processed image.
   - The `Image` object returns the extracted features to the `MobileApp`.

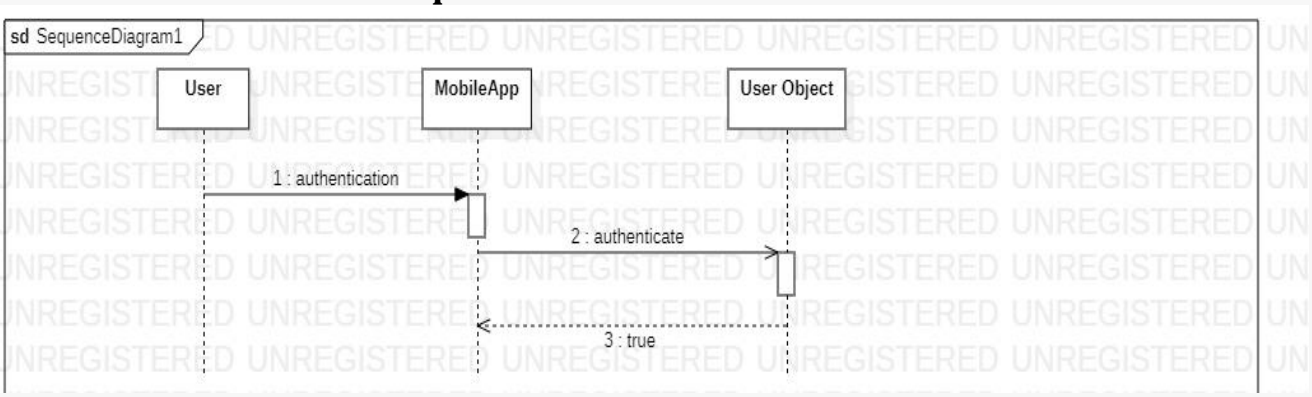
3. **Image Retrieval Sequence:**
   - User initiates a search by providing an image or selecting an image from the gallery.
   - The `MobileApp` creates an instance of the `Feature` object for the query image.
   - The `MobileApp` sends the query features to the `RetrievalEngine`.
   - The `RetrievalEngine` calls the `queryImageDatabase()` method of the `ImageDatabase` to find matching images based on the query features.
   - The `ImageDatabase` searches its stored images using the features and returns a list of matching `Image` objects to the `RetrievalEngine`.
   - The `RetrievalEngine` returns the list of matching images to the `MobileApp`.
   - The `MobileApp` displays the search results to the user.

4. **External Services Integration Sequence:**
   - The `MobileApp` interacts with external services, such as cloud-based image recognition services or social media APIs, for additional functionalities.
   - The `MobileApp` sends requests to the `ExternalServices` object.
   - The `ExternalServices` object communicates with the external services, exchanges data, and retrieves the results.
   - The `ExternalServices` object returns the results or data to the `MobileApp` for further processing or display.

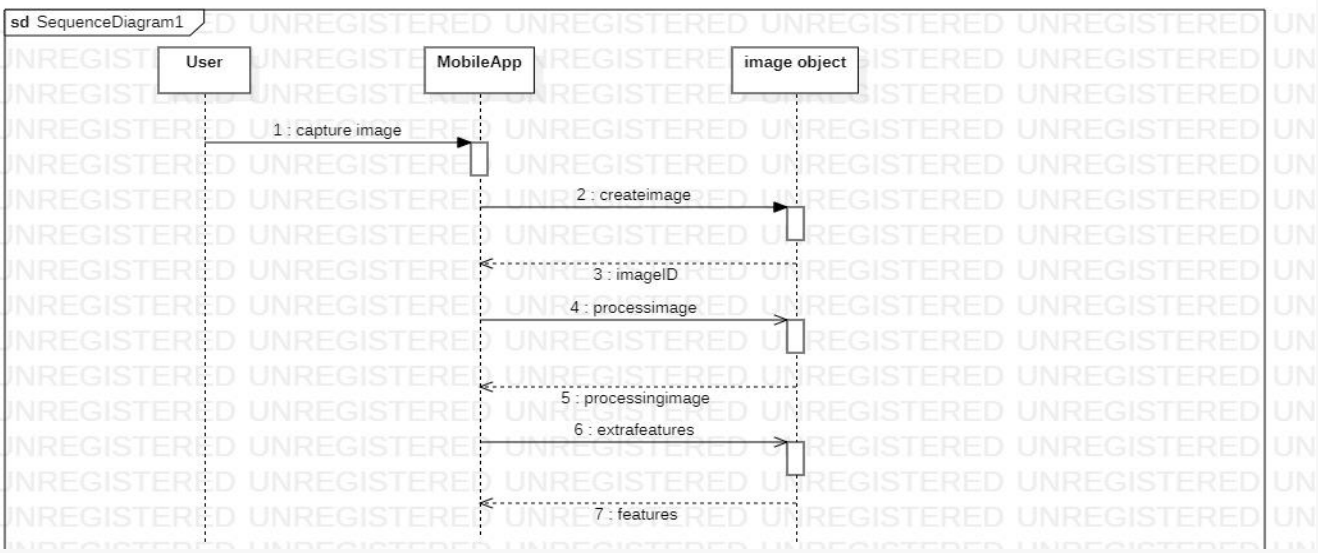
These sequences represent key interactions and exchanges of messages between objects in the system. A sequence diagram can visually represent these interactions, showing the order of messages, the objects involved, and the flow of control throughout the app's functionalities.

Certainly! Here are sequence diagrams for each of the sequences mentioned:

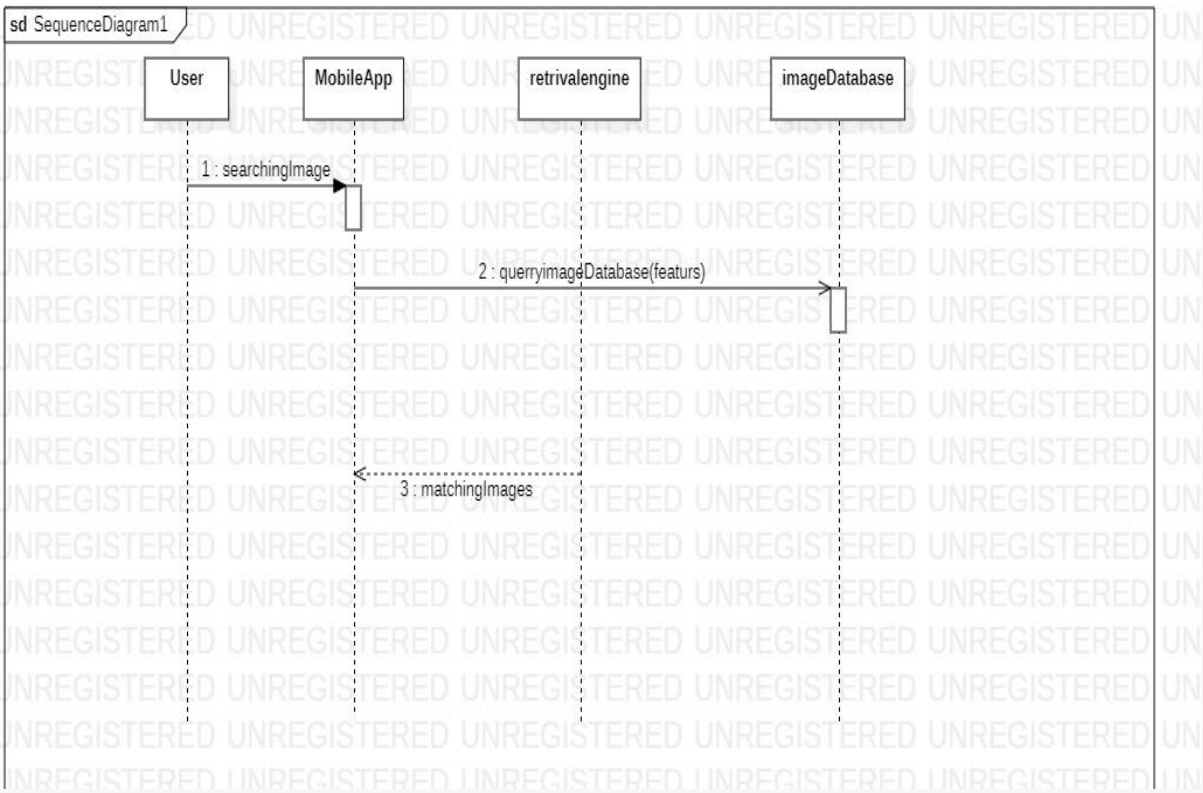
1. **User Authentication Sequence:**



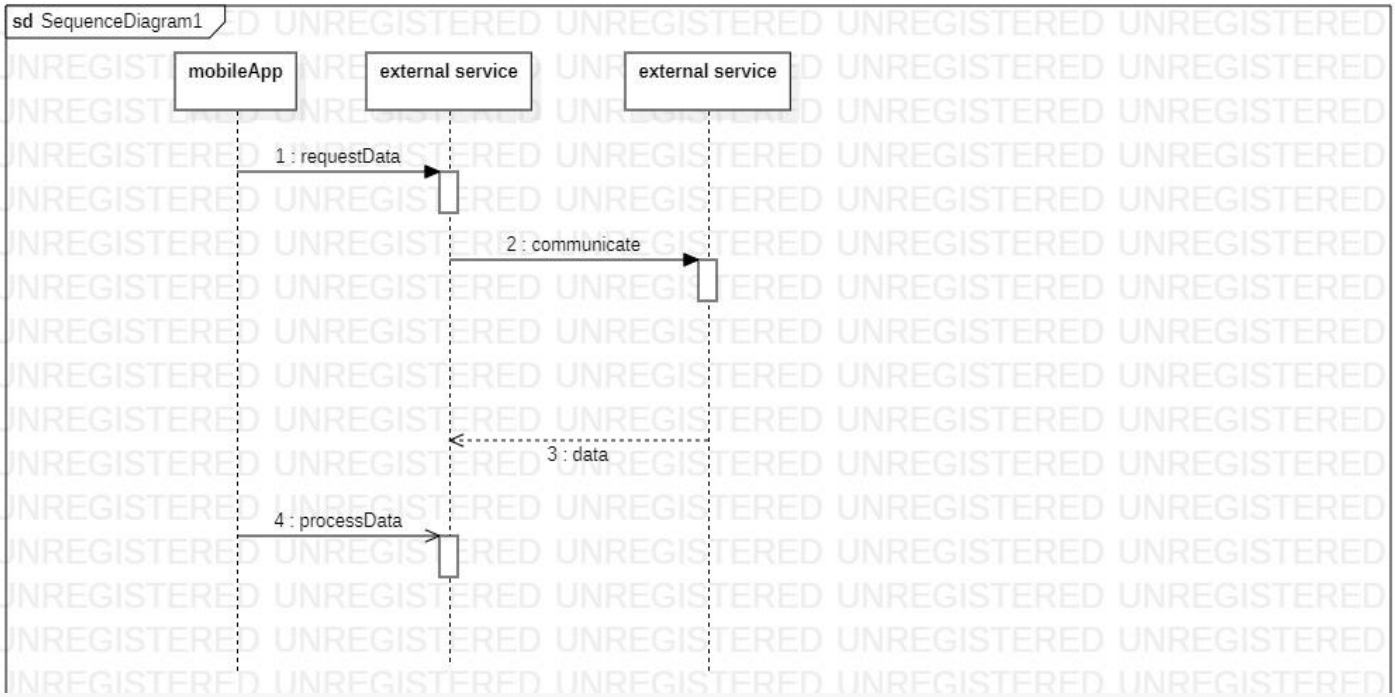
2. **Image Capture and Processing Sequence:**



3. **Image Retrieval Sequence:**



3. External Services Integration Sequence:



Please note that these sequence diagrams provide a high-level representation of the interactions between the involved objects and the flow of messages. The actual sequence diagrams can be further detailed by including method calls, parameters, and return values as required based on your specific system design and interactions.

E. Deployment diagram

A deployment diagram in software engineering is a diagram that depicts the physical deployment of software components (artifacts) on hardware nodes (devices or servers) in a system. It shows how software artifacts are distributed across different nodes in a network and how these nodes are interconnected.

A deployment diagram provides a high-level view of the system's architecture, illustrating the relationships between software components and hardware resources. It helps in understanding the physical deployment configuration and can be useful for system administrators, network engineers, and developers.

Key elements in a deployment diagram include:

Based on the mobile app retrieval and searching system using image matching, the key elements that can be considered for drawing a deployment diagram are:

1. **Mobile Devices:** Represent the physical devices such as smartphones or tablets on which the mobile app will be installed and deployed.

2. **Server**: Represents the server or hosting environment where the backend components of the system are deployed. This server may include the necessary hardware resources and software infrastructure to support the app's functionality.

3. **Mobile App:** Represents the mobile application itself, which is installed and executed on the mobile devices. This includes the user interface, app logic, and client-side functionality.

4. **Backend Server/Application:** Represents the server-side components or application that supports the mobile app's functionality. This can include databases, APIs, processing modules, and other backend services required for image retrieval, storage, and processing.

5. **Image Database:** Represents the database or storage system where images are stored. This can be a separate server or service that handles image storage and retrieval.

6. **External Services:** Represents any external services or APIs that the mobile app interacts with, such as cloud-based image recognition services or social media APIs.

7. **Network Connections:** Represent the network connections or communication channels between the mobile devices, backend server, image database, and external services. These connections can be depicted using lines or arrows connecting the respective nodes.

These elements provide a basic overview of the key components involved in the system's deployment. A deployment diagram can then illustrate how these elements are distributed across the physical devices and servers, showing the connections and relationships between them.

This is the deployment diagram for our app