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

Names	Matricule
TCHUIDJAN JORDAN BRYANT	FE21A320
FUKA NEVILLE TENYI	FE21A199
NFOR WILLY LINWE	FE21A254
MOHAMAN BELLO	FE18A040

COURSE INSTRUCTOR:
Dr. NKEMENI VALERY

ACADEMIC YEAR 2023-2024

Table of contents

Introduction:

Requirement gathering, also known as requirements elicitation or requirements discovery, is the process of capturing, analyzing, and documenting the needs, expectations, constraints, and objectives of stakeholders for a particular system, software, or project. It is a critical phase in the software development life cycle (SDLC) and serves as the foundation for designing and building a solution that meets the stakeholders' requirements.

The purpose of requirement gathering is to understand what the stakeholders want to achieve with the system or software and to define the features, functionalities, and constraints that the solution should encompass. It involves interacting with stakeholders, collecting information, and documenting the requirements in a structured and unambiguous manner.

Processes involved in requirement gathering

The process of requirement gathering typically involves the following steps:

1. **Identify Stakeholders:** Identify the individuals or groups who have a vested interest in the system or software being developed. This may include end-users, clients, managers, domain experts, technical staff, or regulatory authorities.

2. **Engage with Stakeholders:** Establish communication channels with stakeholders to gather their input and understand their perspectives. This can involve conducting interviews, surveys, workshops, or focus groups to elicit their requirements, expectations, and constraints.

3. **Analyze Current Processes and Artifacts:** Review existing documentation, processes, or artifacts related to the project to gain insights into the current state of affairs. This helps in understanding the context, identifying gaps, and assessing the impact of the proposed solution.

4. **Document Requirements:** Document the gathered requirements in a structured and organized manner. This can be done using various techniques such as requirement specifications, use cases, user stories, or functional and non-functional requirements documents. The documentation should be clear, concise, and unambiguous.

5. **Validate Requirements**: Share the documented requirements with stakeholders for their review and validation. Seek their feedback, clarify any ambiguities or conflicts, and ensure that the requirements accurately reflect their needs and expectations.

6. **Prioritize Requirements:** Assign priorities to requirements based on their importance, impact, and feasibility. This helps in making informed decisions about the scope, timeline, and resources required for the project.

7. **Manage Changes:** Requirements may evolve and change throughout the project lifecycle. Establish a change management process to handle any modifications, additions, or deletions to the requirements. This involves evaluating the impact of changes, obtaining stakeholder approval, and updating the documentation accordingly.

8. **Maintain Traceability:** Establish traceability between requirements and their sources, stakeholders, and other project artifacts. This helps in tracking the origin and evolution of requirements and ensures that they are properly addressed during the development and testing phases.

Requirement gathering is an iterative process that involves continuous collaboration and communication with stakeholders. It requires active listening, effective questioning, and the ability to capture both explicit and implicit requirements. The gathered requirements serve as a basis for system design, development, and testing, and play a crucial role in ensuring that the final solution meets the stakeholders' expectations and delivers the intended value.

Processes involved in requirement gathering for Mobile-Based Archival and Retrieval of Missing Objects Application using Image Matching

When gathering requirements for a mobile app for archival and retrieval of missing objects using image matching, it is important to engage with stakeholders, understand their needs, and define the scope of the project. Here is an outline of the requirements gathering process for this type of mobile app:

1**. Identify Stakeholders:**
- Determine the primary stakeholders involved, such as users, administrators, and law enforcement agencies.
- Identify any secondary stakeholders, including developers, database administrators, and support staff.
- Schedule meetings or interviews with stakeholders to gather their input.

2. **Conduct Stakeholder Interviews:**
- Interview stakeholders to understand their expectations, needs, and goals for the application.
- Ask questions to gather information about the missing object identification process, existing challenges, and desired functionality.
- Discuss potential use cases, specific requirements, and any constraints or limitations.

3. **Define Functional Requirements:**
- Based on the stakeholder interviews, identify the core functionalities required for the app, such as:
  - Image capture: Determine the desired features for capturing images of missing objects, including camera access, image quality, and image metadata.
  - Image matching: Understand the requirements for comparing captured images with the database, including the matching algorithm, performance expectations, and accuracy thresholds.
  - Database storage: Determine the necessary data fields and structures for storing missing object information, such as descriptions, images, timestamps, and contact details.
  - Search and retrieval: Define the search criteria and mechanisms for retrieving missing object information from the database, including filters, sorting options, and search result presentation.
  - User authentication: Determine the requirements for user registration, login, and authentication mechanisms to ensure authorized access to the app and its functionalities.
  - Notification system: Discuss the need for real-time notifications or updates regarding missing objects, such as alerts when a potentially matching image is found.

4. **Consider Non-functional Requirements:**
   - Identify non-functional requirements that define the characteristics and qualities of the app, such as:
     - Performance: Determine the expected response times for image matching and database queries.
     - Scalability: Discuss the anticipated user base and potential growth, and ensure the app can handle increasing data and user load.
     - Security: Identify security requirements, such as data encryption, secure authentication, and access control measures to protect sensitive information.
     - Usability: Consider user experience (UX) and user interface (UI) design principles, including intuitive navigation, clear instructions, and accessibility features.
     - Compatibility: Determine the targeted mobile platforms (e.g., iOS, Android) and their versions for the app's development and deployment.

5. **Prioritize and Document Requirements:**
   - Prioritize the identified requirements based on their importance and impact on the overall functionality of the app.
   - Document the requirements in a clear and concise manner, ensuring they are unambiguous, measurable, and traceable.
   - Create user stories, use cases, or requirement specifications to capture the requirements effectively.

6. **Validate Requirements with Stakeholders:**
   - Share the documented requirements with stakeholders for their review and validation.
   - Seek feedback and clarification on any ambiguities or potential gaps.
   - Ensure that the requirements accurately reflect the stakeholders' needs and expectations.

Throughout the requirements gathering process, maintain open lines of communication with stakeholders to address any changes, refine requirements, and manage expectations. Regularly review and update the requirements documentation as the project progresses, and involve stakeholders during the verification and validation phases to ensure alignment with their needs.

Process 1: Identifying stakeholders

Stakeholders are individuals, groups, or organizations that have a vested interest in the outcome of a project or the use of a system or software being developed. They can include end-users, clients, managers, subject matter experts, technical staff, regulatory authorities, or any other entity that may be affected by or have an influence on the project.
Identifying stakeholders is important in the requirement gathering process for several reasons:

1. **Understanding Needs and Expectations:** Different stakeholders have different needs, expectations, and perspectives. By identifying and engaging with stakeholders, you can gain a comprehensive understanding of their requirements and ensure that the final solution meets their expectations. It helps in avoiding assumptions and developing a solution that truly addresses their needs.

2. **Gathering Diverse Perspectives:** Stakeholders bring diverse viewpoints and expertise to the project. They may have unique insights, domain knowledge, or constraints that need to be considered during requirement gathering. By involving stakeholders from various roles and backgrounds, you can gather a wide range of requirements and ensure a more comprehensive solution.

3. **Resolving Conflicts and Negotiating Trade-offs**: Stakeholders may have conflicting requirements or priorities. Identifying stakeholders allows you to identify these conflicts early on and facilitate discussions to resolve them. It helps in negotiating trade-offs and finding a balance that satisfies the needs of different stakeholders.

4. **Mitigating Risks and Compliance:** Stakeholders may include regulatory authorities or compliance officers who can provide guidance on legal and regulatory requirements that need to be addressed. By involving them in the requirement gathering process, you can ensure that the solution conforms to applicable standards, regulations, and best practices.

5. **Communication and Collaboration:** Stakeholders are key participants in the requirement gathering process. Engaging with them establishes clear communication channels, builds trust, and fosters collaboration. It helps in maintaining transparency, managing expectations, and ensuring that stakeholders feel heard and valued throughout the project.

6. **User-Centric Design:** End-users are among the most critical stakeholders in requirement gathering. Identifying them allows you to involve them directly in the process, understand their pain points, and design solutions that are user-friendly, intuitive, and aligned with their needs. User involvement helps in creating a user-centric design and improving user satisfaction.

7. **Change Management and Acceptance:** Stakeholders who will be impacted by the solution need to be involved and informed about the changes being implemented. By identifying and engaging them from the beginning, you can manage their expectations, address concerns, and increase the chances of acceptance and adoption of the final solution.

Overall, identifying stakeholders in requirement gathering is crucial for understanding their needs, gathering diverse perspectives, resolving conflicts, ensuring compliance, fostering collaboration, and ultimately delivering a solution that meets the expectations of all stakeholders involved. It helps in creating a sense of ownership and engagement, leading to a higher likelihood of project success.

Process 2: User Requirements Elicitation

To elicit user requirements for a mobile application, it is important to engage with end-users and involve them in the requirement gathering process. Here are some techniques and activities that can be used to elicit user requirements effectively:

1. User Interviews: Conduct one-on-one interviews with representative end-users to understand their goals, tasks, preferences, and challenges. Ask open-ended questions and encourage them to provide specific examples or scenarios related to the mobile application.

2. Surveys and Questionnaires: Distribute surveys or questionnaires to a larger group of end-users to gather their feedback and preferences. Use structured questions to collect quantitative data and open-ended questions to gather qualitative insights.

3. Contextual Inquiry: Observe and interact with end-users in their natural environment where they would typically use the mobile application. This helps in understanding their workflows, workarounds, and the context in which the application will be used.

4. Focus Groups: Conduct group discussions or focus groups involving multiple end-users. Encourage participants to share their experiences, ideas, and expectations regarding the mobile application. Facilitate brainstorming sessions to generate additional insights.

5. Prototyping and Usability Testing: Develop prototypes or mockups of the mobile application and involve end-users in usability testing sessions. Observe how they interact with the prototype, gather their feedback, and identify areas for improvement.

6. User Stories: Utilize user stories as a technique to capture user requirements in a concise and user-centric manner. User stories typically follow the format of "As a [user role], I want to [goal] so that [benefit]."

7. Use Case Analysis: Develop use cases to describe specific interactions and scenarios that the mobile application should support. Identify the actors, their goals, and the steps required to achieve those goals. Use use cases to validate and refine user requirements.

8. Persona Development: Create personas representing different types of end-users. These personas are fictional characters that embody the characteristics, goals, and behaviors of actual user groups. Use personas as a reference to identify user requirements that align with different user profiles.

9. User Feedback and Iterative Design: Throughout the development process, involve end-users in providing feedback on design iterations, prototypes, or beta versions of the mobile application. This iterative approach helps in refining and validating user requirements based on their direct input.

10. Collaborative Workshops: Conduct collaborative workshops involving end-users, designers, and developers. Use interactive techniques such as design thinking exercises, card sorting, or affinity diagramming to encourage participation and gather user requirements collectively.

Remember to document the user requirements in a clear and structured manner, ensuring they are specific, measurable, achievable, relevant, and time-bound (SMART). Regularly validate and refine the requirements with end-users to ensure that the mobile application meets their needs and expectations.

Process 3: Functional Requirements

Functional requirements define the specific functions, capabilities, and behaviors that a system or software application must possess to fulfill its intended purpose. In the case of a mobile-based archival and retrieval app for missing objects using image matching, the functional requirements could include:

User Registration and Authentication:

- Users should be able to create accounts and log in securely.
- Authentication mechanisms, such as email verification or password protection, should be implemented.

Object Submission:

- Users should be able to submit information about missing objects, including relevant details (e.g., description, category, date, location).
- Users should be able to upload images of the missing objects.

Image Matching:

- The app should employ image matching algorithms to compare submitted images with a database of archived images.
- The system should provide accurate matching results based on image similarity or specific matching criteria.

Search and Retrieval:

- Users should be able to search the app's database for archived missing objects using various parameters (e.g., keywords, object characteristics, location).
- The app should display search results with relevant information and matched images.

Notification and Alert System:

- The app should notify users when a potential match is found for a missing object they submitted.
- Users should receive alerts about new missing object reports or updates related to their submitted objects.

User Communication:

- Users should be able to communicate securely within the app, such as sending messages or providing additional information about missing objects.

User Profile Management:

- Users should be able to manage their profiles, including personal information, contact details, and notification preferences.

Data Storage and Security:

- The app should securely store and manage user-submitted data, including images and personal information.
- Adequate measures should be taken to protect user privacy and comply with relevant data protection regulations.

Reporting and Analytics:

- The app should provide reporting and analytics features to track user activity, search patterns, and system performance.
- Statistical insights or visualizations may be generated to identify trends or patterns related to missing objects.

App Settings and Preferences:

- Users should have the ability to customize app settings and preferences, such as language selection or notification settings.

These functional requirements are tailored specifically for a mobile app focused on archival and retrieval of missing objects using image matching. It's essential to further refine and specify these requirements based on the specific needs, constraints, and target users of your application.

Process 4: Non - Functional requirements

Non-functional requirements specify the qualities or attributes that a system or software application should possess, beyond its functional capabilities. These requirements describe the characteristics that define how the system should behave, perform, or interact with its environment. In the context of a mobile-based archival and retrieval app for missing objects using image matching, some non-functional requirements could include:

Performance:

- Response Time: The app should provide quick and responsive search and matching operations, delivering results within an acceptable time frame.
- Scalability: The system should be able to handle a growing number of users and a large database of archived images without significant degradation in performance.
- Throughput: The app should support concurrent requests and process them efficiently, ensuring smooth user experience.

Reliability and Availability:

- Fault Tolerance: The system should be resilient to failures or errors, minimizing the impact of any single point of failure.
- Availability: The app should be available and accessible to users, minimizing downtime for maintenance or updates.
- Data Backup and Recovery: Adequate mechanisms should be in place to regularly back up data and facilitate recovery in case of data loss or system failures.

Security:

- User Data Protection: The app should employ appropriate security measures to protect user data, including encryption during transmission and storage.
- Access Control: Access to sensitive functionalities or data should be restricted based on user roles and permissions.
- Authentication and Authorization: The app should ensure secure user authentication and enforce proper authorization to protect against unauthorized access.

Usability and User Experience:

- Intuitive Interface: The app should have a user-friendly interface, making it easy for users to navigate, submit information, and perform searches.
- Responsiveness: The app should provide smooth and seamless interactions, minimizing delays or lag.
- Accessibility: The app should adhere to accessibility standards, ensuring it can be used by individuals with disabilities.

Compatibility:

- Device Compatibility: The app should be compatible with a range of mobile devices, operating systems, and screen resolutions commonly used by the target users.
- Offline Functionality: The app should support limited functionality or caching of data for offline usage when an internet connection is not available.

Compliance:

- Legal and Regulatory Compliance: The app should comply with applicable laws, regulations, and industry standards related to data protection, privacy, and security.

Performance Efficiency:

- Resource Utilization: The app should use system resources efficiently, such as CPU, memory, and network bandwidth, to optimize performance and minimize battery consumption.

Maintainability and Extensibility:

- Modularity: The app's architecture should be designed to support easy maintenance, updates, and future enhancements.
- Code Quality: The app's codebase should follow best practices, maintainable coding standards, and be well-documented.
- Flexibility: The app's design should allow for the integration of new features or technologies in the future.

These non-functional requirements complement the functional requirements by addressing aspects related to performance, reliability, security, usability, compatibility, compliance, efficiency, maintainability, and extensibility. It's important to consider these requirements alongside the functional ones to ensure the overall success and quality of the mobile app.

Process 5: Document requirements

For the mobile-based archival and retrieval app for missing objects using image matching, several documents can help in documenting the requirements, design, and other aspects of the app. Here are some important documents that can be created:

Requirements Document: This document captures the functional and non-functional requirements of the app. It includes details about user interactions, system behavior, data inputs and outputs, performance expectations, security requirements, and any other specific requirements identified during the analysis phase.

User Stories: User stories provide a user-centric view of the app's features and functionalities. Each user story represents a specific user goal or requirement in a concise format, typically following the template: "As a [user role], I want to [goal] so that [benefit]." User stories help in understanding user needs and prioritizing development efforts.

Use Case Document: Use cases describe specific interactions or scenarios between users and the app. They outline the actors involved, their goals, and the steps required to achieve those goals. Use cases help in understanding the flow of interactions and can serve as a basis for testing and validation.

System Design Document: This document outlines the overall architecture and design of the app. It includes diagrams, such as system architecture diagrams, database schema diagrams, or sequence diagrams, to illustrate the components, relationships, and data flow within the app.

User Interface (UI) Design Document: This document presents the visual design and layout of the app's user interface. It includes wireframes, mockups, or interactive prototypes that depict the screens, navigation flow, and visual elements of the app.

Database Design Document: If the app involves a database for storing and retrieving data, a database design document can be created. It specifies the tables, relationships, data types, and constraints used in the database schema.

Test Plan and Test Cases: This document outlines the testing strategy and approach for the app. It includes test objectives, test scenarios, and test cases to verify that the app meets the defined requirements. It also specifies the testing methodologies, tools, and environments.

User Documentation: User documentation provides instructions and guidance on how to use the app. It may include user manuals, FAQs, or online help resources that help users navigate the app, understand its features, and troubleshoot common issues.

Deployment and Configuration Guide: This document provides instructions on deploying the app to production environments. It includes details on server requirements, installation steps, configuration settings, and any additional dependencies or third-party integrations.

Maintenance and Support Documentation: This document outlines guidelines and procedures for maintaining and supporting the app after its initial deployment. It includes information on bug reporting, issue tracking, version control, and release management processes.

These documents serve as valuable references throughout the app's development and maintenance lifecycle, ensuring that the app is built according to the defined requirements and providing guidance for future enhancements and updates. The level of detail and extent of documentation may vary depending on the project's scope, complexity, and specific requirements.

Process 6: User-interface requirements

User interface requirements for a mobile app focused on archival and retrieval of missing objects using image matching should aim to provide a seamless and intuitive user experience. Here are some user interface requirements to consider:

Responsive Design: The app should be designed to adapt to various screen sizes and orientations, ensuring a consistent and user-friendly interface across different mobile devices.

Intuitive Navigation: The app should have clear and intuitive navigation menus, buttons, and gestures, allowing users to easily move between different sections and functionalities.

Search Functionality: The search feature should be prominent and easily accessible, enabling users to search for missing objects using keywords, filters, or other relevant parameters.

Visual Feedback: The app should provide visual feedback, such as loading indicators, progress bars, or success/error messages, to keep users informed about ongoing actions or the status of their interactions.

Image Upload and Capture: The app should allow users to upload images of missing objects from their device's gallery or capture images using the device's camera. The image selection or capture process should be intuitive and seamless.

Image Display and Comparison: The app should display matched images or search results in a visually appealing and organized manner, allowing users to compare images and relevant information effectively.

Object Details and Descriptions: Each missing object entry should provide clear and comprehensive details, including descriptions, dates, locations, and any additional relevant information to assist in identification or retrieval.

User Feedback and Reporting: The app should provide a user-friendly interface for users to provide feedback, report issues, or update information about missing objects. This interface should be easily accessible and straightforward to use.

Notifications and Alerts: The app should utilize notification mechanisms, such as push notifications, to inform users about potential matches, updates on missing objects, or important app-related information.

Accessibility: The app should adhere to accessibility guidelines, ensuring that users with disabilities can access and use the app effectively. This includes features like text-to-speech support, high contrast options, and appropriate font sizes.

Consistent Design Language: The app should follow a consistent design language and branding, including color schemes, typography, and visual elements, to establish a cohesive and recognizable user interface.

Error Handling and Assistance: The app should provide clear error messages and guidance when users encounter errors or input validation issues. Additionally, help or support options should be easily accessible to assist users in case they require assistance.

Offline Functionality: The app should provide a seamless experience even when the device is offline. It should display appropriate messages or indicators and allow users to access certain features or view cached data until an internet connection is available.

These user interface requirements aim to create an engaging, user-friendly, and visually appealing app that facilitates efficient navigation, interaction, and retrieval of missing objects. Conducting user research and usability testing can further refine and validate these requirements to meet the specific needs and expectations of the target users.

Process 7: Integration requirements

In the mobile app for archival and retrieval of missing objects using image matching, there could be several external systems, services, or APIs that the app needs to integrate with. Here are a few potential examples:

Image Recognition API: The app may need to integrate with an image recognition API or service that performs the image matching and comparison algorithms. This API would analyze the uploaded or captured images and provide results indicating potential matches or similarities.

Geolocation Service: To enhance the search functionality, the app could integrate with a geolocation service or API that provides location-based data. This integration would enable users to search for missing objects based on their current location or specific geographical areas.

Cloud Storage Service: The app may require integration with a cloud storage service, such as Amazon S3, Google Cloud Storage, or Microsoft Azure Storage. This integration would allow users to upload and retrieve images from a secure and scalable storage solution.

Notification Service: To send push notifications to users regarding updates on missing objects, potential matches, or other important information, the app could integrate with a notification service like Firebase Cloud Messaging (FCM) or Apple Push Notification Service (APNs).

User Authentication Service: If the app includes user accounts and authentication, integrating with a user authentication service like Firebase Authentication or OAuth-based services (e.g., Google Sign-In, Facebook Login) can streamline the authentication process and ensure secure user access.

Mapping and Routing Services: If the app incorporates features like displaying missing objects on a map, providing directions to potential sighting locations, or visualizing search results geographically, integration with mapping and routing services like Google Maps API or Mapbox API may be necessary.

Social Media Integration: The app could allow users to share missing object information or potential matches on social media platforms. Integrating with popular social media APIs, such as Facebook, Twitter, or Instagram, would enable seamless sharing and engagement with social networks.

Analytics and Crash Reporting Service: Integrating with an analytics and crash reporting service like Google Analytics or Firebase Crashlytics can help track app usage, user behavior, and identify and report any crashes or errors for further analysis and improvement.

It's important to consider the specific needs and requirements of the app to identify the most suitable external systems, services, or APIs for integration. The choice of integration depends on factors such as functionality, scalability, security, availability, and the development platform or framework being used for the app.

Process 8: Potential risk requirements

The process of gathering and documenting requirements for a mobile app can face various risks and challenges. Here are some potential risks that may impact the successful gathering and documentation of requirements:

Incomplete or Ambiguous Requirements: There is a risk of not capturing all necessary requirements or documenting them in a clear and unambiguous manner. This can lead to misunderstandings, gaps in functionality, and subsequent rework during the development phase.

Changing Requirements: Requirements are prone to change due to evolving user needs, market dynamics, or technological advancements. If requirements are not effectively managed and controlled, frequent changes can disrupt the gathering and documentation process, leading to delays and scope creep.

Stakeholder Misalignment: Different stakeholders may have divergent opinions, priorities, or expectations regarding the app's requirements. Conflicting viewpoints and lack of alignment can hinder the agreement on key requirements and impede progress.

Communication Issues: Poor communication between the project team and stakeholders can result in misunderstandings, misinterpretations, or inadequate information exchange. This can lead to inaccuracies, inconsistencies, and gaps in the documented requirements.

Lack of User Involvement: Insufficient involvement of end-users and relevant stakeholders throughout the requirements gathering process can result in a disconnect between the app's functionalities and user needs. It is crucial to actively engage users to ensure their requirements and perspectives are adequately represented.

Technical Feasibility Constraints: Certain requirements may be technically challenging or infeasible within the given constraints, such as platform limitations, security requirements, or budgetary constraints. Identifying and addressing such constraints early in the process is essential to avoid unrealistic expectations and delays.

Scope Creep: There is a risk of the project's scope expanding beyond the initial defined boundaries during the requirements gathering phase. Additional features or functionalities that are not properly evaluated or controlled can lead to schedule delays, increased costs, and compromised quality.

Lack of Domain Knowledge: Insufficient understanding of the domain or industry in which the app operates can impact the ability to gather accurate and relevant requirements. It is important to involve subject matter experts who possess the necessary domain knowledge to ensure comprehensive and effective requirements gathering.

Cultural or Language Barriers: In projects involving diverse stakeholders across different cultures or languages, challenges may arise due to differences in communication styles, language proficiency, or cultural norms. These barriers can hinder effective requirements gathering and documentation.

Resource Constraints: Limited availability of resources, such as time, budget, or skilled personnel, can pose challenges in adequately conducting requirements gathering activities. Insufficient resources may lead to rushed or incomplete documentation, impacting the quality of the final requirements.

To mitigate these risks, it is essential to employ effective requirements gathering techniques, establish clear communication channels, actively involve stakeholders, conduct regular reviews and validations, and maintain a flexible and iterative approach to requirements documentation.

Conclusion

Based on the information provided above, requirements gathering for a mobile app focused on archival and retrieval of missing objects using image matching is a crucial and complex process that requires careful attention to various factors. Here are some key points to consider:

User-Centric Approach: The app should be designed with a strong focus on user needs and expectations. Involving end-users and stakeholders throughout the requirements gathering process is essential to ensure that the app effectively addresses their requirements and provides a seamless user experience.

Clear and Comprehensive Documentation: The requirements should be documented in a clear, unambiguous, and comprehensive manner. This documentation serves as a reference for development, testing, and future enhancements. It should capture both functional and non-functional requirements, including user interactions, system behavior, data inputs and outputs, performance expectations, security requirements, and more.

Integration with External Systems and APIs: The app may need to integrate with external systems, services, or APIs to enhance its functionality. These integrations could include image recognition APIs, geolocation services, cloud storage services, notification services, mapping and routing services, and more. Identifying and integrating with the appropriate external systems is crucial to ensure seamless and efficient operations.

Risk Management: Various risks can impact the requirements gathering process, such as incomplete requirements, changing requirements, stakeholder misalignment, communication issues, and technical feasibility constraints. It is important to proactively identify and address these risks through effective communication, stakeholder engagement, risk analysis, and mitigation strategies.

Iterative and Agile Approach: Requirements gathering for a mobile app is an iterative process that may evolve throughout the project lifecycle. Adopting an agile methodology enables flexibility, frequent feedback loops, and continuous refinement of requirements. This approach helps manage changing requirements, accommodate user feedback, and deliver a high-quality app.

Collaboration and Domain Knowledge: Collaboration among the project team, stakeholders, and domain experts is vital for successful requirements gathering. Involving subject matter experts who possess domain knowledge and involving users in the process ensures that requirements accurately reflect the needs of the target audience and the industry context.

Usability and Accessibility: The app should prioritize usability and accessibility, ensuring that the user interface is intuitive, visually appealing, and caters to users with disabilities. Considerations like responsive design, intuitive navigation, clear feedback, and adherence to accessibility guidelines contribute to a positive user experience.

In conclusion, requirements gathering for a mobile app for archival and retrieval of missing objects using image matching requires a user-centric, iterative, and collaborative approach. Clear documentation, integration with external systems, risk management, and consideration of usability and accessibility are crucial aspects of this process. By addressing these factors effectively, the app can be developed to meet user needs, provide a seamless experience, and improve the chances of successful adoption and usage.