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 3

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

Catalog

A) Introduction:

Requirement analysis, also known as requirements engineering or requirements gathering, is the process of identifying, understanding, documenting, and validating the needs and expectations of stakeholders for a software system or application. It is a critical phase in the software development life cycle, as it lays the foundation for the successful design, development, and implementation of a solution that fulfills the desired objectives.

Requirement analysis involves gathering information from various stakeholders, such as business owners, users, customers, and subject matter experts, to understand their goals, processes, and constraints. The primary objective is to elicit and document functional and non-functional requirements that define what the software system should do, how it should behave, and the quality attributes it should possess

While we collected all the information in the requirement gathering phase now we have analyze that data . We separated them into two categories that is : High priority and low priority requirements

B) High priority requirements

To determine the high priority requirements for your app, it's important to consider the specific goals, target audience, and context of your application. However, here are some general categories of requirements that are often considered high priority in many app development scenarios:

1. **Core Functionality:** Identify the essential features and functionalities that are crucial for the basic operation and purpose of the app. These are the functionalities without which the app cannot fulfill its primary objectives. Examples could include user registration/authentication, content creation, search functionality, and basic user interactions.
2. **Performance and Responsiveness:** Users expect apps to be fast, responsive, and provide a smooth user experience. Therefore, prioritizing requirements related to performance optimization, minimal loading times, efficient data retrieval, and responsive user interfaces is often critical.
3. **Security and Privacy:** Ensuring the security and privacy of user data is a high priority requirement. Implementing secure authentication mechanisms, data encryption, protection against vulnerabilities, and compliance with relevant data protection regulations are crucial considerations.
4. **Usability and User Experience:** Users should find your app intuitive, easy to navigate, and visually appealing. Prioritize requirements related to user interface (UI) and user experience (UX) design, ensuring accessibility, clear navigation, and effective information presentation.
5. **Compatibility and Platform Support:** Determine the target platforms (e.g., iOS, Android) and devices (smartphones, tablets) for your app and prioritize requirements related to compatibility and platform-specific guidelines. Ensuring the app functions well across different devices and operating systems is critical.
6. **Offline Functionality:** Depending on the nature of your app, consider whether it should support offline functionality. This allows users to access certain features or content even without an internet connection. Prioritize requirements related to offline caching, data synchronization, and offline access to essential functionalities.
7. **Error Handling and Logging:** Implement effective error handling mechanisms to provide informative error messages and handle exceptions gracefully. Additionally, prioritize requirements related to logging and error reporting, enabling developers to identify and address issues promptly.
8. **Integration with External Services:** If your app requires integration with external services or APIs, prioritize requirements related to seamless integration, data exchange, and interoperability with those services. This allows your app to leverage the functionality and data provided by external systems.
9. **Analytics and Reporting:** Incorporating analytics and reporting capabilities can provide valuable insights into user behavior, app performance, and usage patterns. Prioritize requirements related to data collection, analysis, and visualizations to facilitate informed decision-making and continuous improvement.

Remember, the priority of requirements may vary based on your specific app and its objectives. It's crucial to engage in discussions with stakeholders, conduct user research, and consider feedback to determine the high priority requirements that align with your app's purpose and target audience.

So these various requirements above can be grouped into sub categories like : F**unctional and non- funcional requirements, platform (user- interface requirement), data requirements**

1. 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.

2. 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.

3.Data requirements

For a mobile app that focuses on retrieval and searching for objects using image matching, the data requirements can be outlined as follows:

1. **Object Image Database:** You will need a database to store the images of objects that users can search for. This database should be capable of efficiently storing and retrieving a large number of object images. Each object image may require additional metadata such as object name, description, tags, or other relevant information.

2. **User Image Input:** Define the mechanism for users to input images into the app for searching. This can involve capturing images using the device's camera or allowing users to upload images from their device's gallery. Ensure compatibility with different image formats and sizes.

3. **Image Preprocessing:** Implement image preprocessing techniques to enhance the quality and consistency of user-provided images. This may include resizing, normalization, noise reduction, or other preprocessing steps to improve the accuracy of image matching.

4. **Image Feature Extraction:** Utilize image feature extraction algorithms to extract relevant features from both user-provided images and object images in the database. These features can include shape, color, texture, or other visual descriptors that help in comparing and matching images.

5. **Image Matching Algorithms:** Develop or employ image matching algorithms that can compare the extracted features of user-provided images with the features of object images in the database. The algorithms should calculate similarity scores or match scores to rank the retrieved objects based on their similarity to the user's input image.

6. **Indexing and Retrieval:** Implement efficient indexing and retrieval mechanisms to quickly search and retrieve objects based on their features or metadata. This can involve techniques like reverse image indexing, content-based indexing, or utilizing specialized databases or search engines.

7. **Object Metadata:** Store additional metadata associated with each object image in the database. This metadata can include information like object category, attributes, location, user ratings, or user-generated tags. Determine the required metadata fields and their structure.

8. **Scalability and Performance:** Consider the scalability and performance requirements of the app, as image matching and retrieval can be computationally intensive processes. Ensure that the database and matching algorithms can handle a large number of concurrent users and provide fast and responsive search results.

9. **User Feedback and Ratings:** Implement features that allow users to provide feedback on the search results or contribute additional information about objects. This feedback can help improve the accuracy and relevance of future search results.

10. **Data Privacy and Security:** Address data privacy and security concerns related to user-uploaded images and associated metadata. Implement measures to protect user data, ensure secure transmission, and adhere to relevant privacy regulations.

11. **Analytics and Reporting:** Incorporate analytics capabilities to gather insights about user behavior, popular search queries, or usage patterns. This can help improve the app's functionality and user experience over time.

Ensure that the app's architecture, image processing algorithms, and database design are optimized for efficient and accurate object retrieval and matching based on user-provided images.

4.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.

C) Low priority requirements

A low priority requirement refers to a feature or functionality that is not critical for the core operation or immediate success of the mobile app. These requirements are considered less urgent and can be deprioritized compared to high priority or essential requirements. While they may enhance the overall user experience or provide additional functionalities, they can be deferred to later stages or iterations of the app development process. For the mobile app retrieval and searching for objects using image matching, some examples of low priority requirements could be:

1. **Social Media Integration:** Allowing users to share search results or object information on social media platforms such as Facebook, Twitter, or Instagram. While this can enhance user engagement and virality, it is not essential for the core functionality of the app.

2. **Advanced Filters and Sorting:** Implementing advanced filtering and sorting options for search results, such as sorting by price, popularity, or user ratings. While this can improve user convenience, it may not be crucial in the initial stages and can be added in future iterations.

3. **Augmented Reality (AR) Integration:** Integrating augmented reality features to allow users to visualize objects in their surroundings using the device's camera. This can provide an immersive experience but may require additional development effort and can be considered as an enhancement in later versions.

4. **Multiple Language Support:** Providing localization and support for multiple languages to cater to a wider user base. While this can increase accessibility, it may not be a priority in the early stages of the app and can be added as the app gains traction.

5. **User Profiles and Personalization:** Allowing users to create profiles, save search preferences, or receive personalized recommendations. While this can enhance user engagement and retention, it may not be essential for the core functionality of image retrieval and search.

6. **Offline Mode:** Enabling offline access to previously searched objects or cached search results when the device has no internet connectivity. While this can improve user convenience, it may not be a critical requirement initially and can be considered for future updates.

7. **Gamification Elements:** Incorporating gamification elements such as badges, achievements, or leaderboards to encourage user engagement and competition. While this can add a fun factor, it may not be a high priority in the initial stages and can be implemented later.

8. **Voice Search:** Integrating voice recognition capabilities to allow users to search for objects using voice commands. While this can provide an alternative input method, it can be considered as an additional feature to be implemented at a later stage.

It's important to note that the prioritization of requirements can vary based on the specific goals, target audience, and constraints of your app. It's recommended to involve stakeholders, conduct user research, and gather feedback to determine the appropriate prioritization of low priority requirements for your mobile app.

So these various requirements above can be grouped into sub categories like : **Integration and Business requirements**

1. 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.

2. Business requirements

A business requirement is a statement that describes a specific need or objective from a business perspective. It focuses on the goals, outcomes, or benefits that the mobile app should achieve to meet the organization's strategic objectives. Business requirements help align the app's functionalities with the overall business strategy. For a mobile app retrieval and searching for objects using image matching, some examples of business requirements could be:

1. **Increased User Engagement:** The app should aim to increase user engagement by providing a seamless and intuitive interface for image-based object retrieval and search. This can be achieved through features like easy image input, fast and accurate search results, and a visually appealing user interface.

2. **Enhanced User Experience:** The app should prioritize delivering a positive user experience by ensuring quick and relevant search results, intuitive navigation, and user-friendly interactions. The goal is to create a user interface that is easy to understand, visually appealing, and enjoyable to use.

3. **Competitive Advantage:** The app should offer unique and innovative features that differentiate it from competitors in the market. This can include advanced image matching algorithms, superior search accuracy, or additional functionalities that provide value to users and set the app apart from similar offerings.

4. **Monetization Opportunities:** Identify potential monetization strategies within the app. This could include options such as in-app advertisements, premium features or subscriptions, partnerships with e-commerce platforms, or data monetization through analytics and insights.

5. **Scalability and Performance:** Ensure that the app is designed to handle a growing user base and increased usage. It should be able to handle concurrent searches, provide fast response times, and scale effectively as the number of users and database size grows.

6. **Data Analytics and Insights:** Implement mechanisms to collect and analyze user data, search patterns, and user behavior to gain valuable insights. This can help improve the app's performance, identify user preferences, and make data-driven decisions to enhance the user experience and drive business growth.

7. **Business Integration:** Explore opportunities to integrate the app with existing business systems or platforms. This can include integration with e-commerce platforms, inventory management systems, or other business tools to provide a seamless experience for users and leverage existing infrastructure.

8. **Branding and Marketing:** Ensure that the app aligns with the organization's branding guidelines and marketing strategies. The app should reflect the brand's image, messaging, and values to reinforce brand recognition and enhance marketing efforts.

9. **User Retention and Loyalty:** Focus on features and functionalities that encourage user retention and loyalty. This can include personalized recommendations, notifications, rewards, or social sharing options that create a sense of community and keep users engaged with the app over the long term.

10. **Compliance and Security:** Ensure that the app complies with relevant legal and regulatory requirements, especially concerning user data privacy and security. Implement measures to protect user data, secure data transmission, and adhere to industry best practices.

These are just a few examples of business requirements for a mobile app retrieval and searching for objects using image matching. It's important to align these requirements with the overall business strategy and objectives to ensure that the app delivers value and supports the organization's goals.

3. 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 analysis 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 analysis 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 analysis 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.