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

REPUBLIC OF CAMEROON

Peace-Work-Fatherland

FACULTY OF
ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

CEF440: INTERNET PROGRAMMING (J2EE) AND MOBILE PROGRAMMING

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

TASK THREE: **REQUIREMENTS ANALYSIS**

Course Supervisor:
Dr. NKEMENI VALERY
University Of Buea

GROUP 1 MEMBERS

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

Table of Contents

1. INTRODUCTION

1.1. Background:

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

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

1.2. Requirement Analysis Process:

The requirements analysis phase in software development is foundational, orchestrating the translation of stakeholder needs into actionable guidelines for system development. This pivotal phase ensures clear communication between the development team and stakeholders, aligning with business objectives and user preferences while defining project scope to prevent scope creep and maintain focus. Simultaneously, it identifies potential risks early, enabling proactive measures to minimize rework and delays. Furthermore, by ensuring that the final product meets specified criteria, it enhances stakeholder satisfaction and facilitates accurate project budgeting and scheduling, thereby reducing the likelihood of overruns. Ultimately, the requirements analysis phase serves as a critical compass, guiding architectural decisions, system design, and implementation strategies, ensuring the successful execution of software projects.

In this documentation for the Requirements Analysis For the mobile-based archival and retrieval of missing objects application, the requirements analysis will proceed through several key steps. By initially reviewing the requirements gathering phase and the stakeholder requirements outlined. These requirements will then be categorized into functional and non-functional categories, ensuring clarity and specificity. Prioritization will follow, highlighting essential features for initial implementation. User activity and flow diagrams will be created to visualize system interactions. Constraints, dependencies, assumptions, and risks will be meticulously identified to inform project planning and mitigation strategies.

2. REQUIREMENTS ANALYSIS PROCESS: STEPS AND METHODOLOGY

2.1. Review of the Requirements Gathering phase

2.1.1. Identified Stakeholders

This was the first step involved in the gathering phase for the project in which project stakeholders were identified to be:

- Any individual: Any one can be an end user of the image retrieval and archival system
- Development Team: Responsible for developing the system
- The Business owners: Those sponsoring the development process and making the profits of the system which in this case is the development team

2.1.2. Elicitation techniques

This involved performing interviews, online survey and brainstorming to understand the stakeholder requirements and the functioning of related systems on how their methodologies can be improved to better our system. Interviews were performed on related organizations such as the Taxi Union, Travelling Agencies and the Police, as well as on random individuals to get detailed insights on project requirements. An online survey was done to access the general public needs, acceptance and reluctance status as well as individual features deemed essential by these individuals and last but not the least brainstorming to come up with functionalities not listed in the previous methods but that might be vital for the success of the project.

The next phase, that is the requirements specification phase involves categorizing these requirements obtained from the gathering phase into functional or nonfunctional requirements

2.2. Requirement Specification

Requirement's specification is the process of translating stakeholder needs into clear guidelines for system development. It involves categorizing requirements into functional (what the system should do) and non-functional (how it should perform) aspects. This phase ensures a shared understanding of project objectives, guiding the development team towards building a solution that meets stakeholders' expectations.

2.2.1. Functional Requirements

Based on the requirements outlined by users during the gathering and elicitation phase, the following functional requirements were derived

1. Image recognition and Real time image processing

This is the core functionality of the image processing software as it the primary reason why users access the system. The platform implements an image recognition algorithm to analyze uploaded images and identify potential matches from other images which have been stored in the system's database. This is done through the integration of an image recognition and image processing API.

2. User Authentication and Security

This feature enables users to be able to create accounts and log in securely into the system directly from their mobile phones. This feature also enables users to submit visual information for the application to analyze ensuring security in the user private data and access to only authorized users in the system. It is implemented through the use of passwords and biometric which should match those previously given by the user in the database.

3. Database Integration

The application is connected to a database which contains information about missing items, user information, descriptions and other relevant details. The database is updated every time to ensure that it is accurate and ensures correct outputs. An inaccurate database leads to mismatch and wrong outputs.

4. User Interface Design

An intuitive and user-friendly interface is designed which makes navigation through the application easy, uploading images smooth and searching for missing items quick. Also, providing clear instructions on how to use the app hence attracts more users who have lost items to look up to the application for help in locating those items.

5. Notification System

Keeping users informed about potential matches for their missing items increases engagement and encourages active participation in the search process. The app will have a notification system to alert users when a potential match for their missing item is found. Notifications can be sent via push notifications or email, depending on user preferences.

6. Feedback Loop

The application allows users to provide feedback which helps improve the app's accuracy over time, enhancing user satisfaction and the overall effectiveness of the platform. Users will have the ability to

provide feedback on potential matches suggested by the application. This feedback helps improve the accuracy of the system over time by providing valuable data for training the image recognition algorithms.

7. Location-based Services

While important for some users, this feature may not be as critical as others, depending on the nature of the missing items and the user's search preferences. Utilizing GPS technology to enable location-based searching for missing items. Users can filter search results based on their current location or specify a particular area of interest, improving the relevance of search results.

8. Image Capture

Although convenient, users may be willing to upload images from their device's gallery if real-time image capture is not immediately available. Integration with the device's camera functionality to allows users capture images directly within the app. This feature enhances user convenience by enabling them to quickly upload images of missing items without leaving the application.

9. Offline Mode

Providing offline functionality for users to access certain features of the application, such as viewing previously searched items or accessing saved data, even when they are not connected to the internet. This enhances usability and ensures uninterrupted access to essential features.

10. Reward system to those who find missing items

This ensures that the system remains active as users will be more motivated to find and return found goods especially those which might not be of direct use or benefit to them. This functionality can be considered the fuel or driving force of the application keeping an engaged community.

11. Texting and calling service integration

To facilitate communication between the finder and the owner of a missing item

12. Live location between someone who has found
To facilitate meeting of the finder and the owner of the item for retrieval

13 Free access and availability to all communication updates on missing items.

14 A tap to fill in info and address if someone comes across a missing item

15 A place where users can report a missing object with: location, time and date and if available, the registration number pasted on it

16 An agency where missing things can be found and kept

17 Image uploading and customizable searches or reports

18 A sort of community where for reporting missing and found items

2.2.2. Non- Functional Requirements

Nonfunctional requirements specify how the system should perform, giving the quality attributes, constraints and characteristics that governs the system behavior. They are also referred to as quality attributes and are critical in ensuring that the software meets the expectations and needs of the users. Below are some of the non-functional requirements our system will implement.

1) Performance

The system should be able to process image recognition requests quickly and efficiently, providing near-real-time results to users. Response times for image processing and database queries should be optimized to minimize user wait time. While giving the correct and accurate output of the image processing.

2) Scalability

The application should be designed to handle a potentially large volume of users and image uploads. It should be able to scale horizontally to take in an increasing number of users without sacrificing its performance or its efficiency in the output.

3) Reliability

The system should be highly reliable, with minimal downtime or service interruptions. This requires robust error handling, fault tolerance, and backup mechanisms to ensure continuous operation. This means when the system is going through a fault or update, the users should not be able to suspect a thing.

4) Availability

The application should be available to users whenever they need it, with high availability and uptime. This may involve implementing redundant servers, load balancing, and failover mechanisms to mitigate the impact of hardware failures or maintenance.

5) Privacy

Users' privacy should be respected, and personal data should be handled in accordance with relevant privacy regulations. This involves obtaining user consent for data collection and processing, as well as implementing privacy-enhancing features such as anonymization and data minimization.

6) Security

Data security is critical to protect user information and prevent unauthorized access to sensitive data. The system should implement encryption, secure authentication protocols, and access control mechanisms to safeguard user data and prevent data breaches.

7) Compatibility

The application should be compatible with a wide range of mobile devices and operating systems to maximize accessibility for users. It should be tested on different platforms to ensure consistent performance and user experience across devices.

8) Usability

The user interface should be intuitive and easy to navigate, with clear instructions and feedback to guide users through the process of uploading images and interpreting search results. Accessibility features should also be implemented to accommodate users with disabilities.

9) Maintainability

The system should be designed with maintainability in mind, with clean and well-documented code that is easy to understand and modify. This facilitates future updates and enhancements to the application without introducing unnecessary complexity or risk.

10) Regulatory Compliance

The system should comply with relevant laws and regulations governing data protection, privacy, and security. This includes adherence to industry standards and best practices for handling sensitive information and ensuring transparency in data processing practices.

2.3. Requirements Prioritization and Classification

In software development, requirements prioritization is essential for navigating limited resources and tight timelines. This phase involves evaluating and ranking identified requirements based on their importance and impact on project goals. By prioritizing requirements, teams can focus on delivering critical functionalities early, maximizing value and minimizing risks. Effective prioritization ensures alignment with stakeholder needs, facilitates informed decision-making, and sets the stage for efficient development, ultimately delivering solutions that drive business outcomes.

There are several methods of prioritizing requirements such as the *MoSCow method, the Kano model, weighted scoring, Relative Ranking, cost-benefit analysis, value vs effort matrix, Impact vs Effort matrix*. But for our mobile-base retrieval and archival of lost items system, **weighted scoring** was used to compare the functional requirements and they were later on classified using the **MoSCoW** technique.

2.3.1. Weighted Scoring

For doing the weight scoring, we judged the or prioritized our functional requirements based on 3 main criteria

- **Impact**: How vital the functionality is to the success of the project
- **Urgency**: How pressing the completion of the functionality. (maybe it has to be completed for other functionalities to be possible)
- **Feasibility**: If the functionality is technically possible and if it can be completed in the timeframe given

2.3.2. MoSCoW classification

Based on the scored obtained from the weighted scoring the requirements were classified under the priorities must have, should have, could have and won't have. These classification helps in clearly outlining the order that has to be followed in the execution of the requirements in order to meet the schedule and satisfy users.

2.3.3. Prioritization and classification table

Requirement (1-5)	Impact (1-5)	Urgency (1-5)	Feasibility (1-5)	Weighted score (1-5)	Priority (MoSCoW)
Image recognition and Real-time image processing	5	4	3	4	Must have
User Authentication and Security	5	5	4	4.6	Must have
Database Integration	4	4	4	4	Must have
User Interface Design	4	4	5	4.3	Must have
Notification System	4	5	3	4	Must have
Image uploading	5	4	4	4.3	Must have
Customizable Searches (adding descriptions)	3	3	4	3.3	Should have
Feedback Loop	3	3	4	3.3	Should have
Location-based Services	4	3	3	3.3	Should have
Image Capture integrated in the application	3	2	4	3	Should have
Offline Mode	3	2	2	2.3	Could have
Reward system	4	2	4	3.3	Should have
Texting and calling service integration	4	2	2	2.6	Could have
Live location between finder and owner	3	3	3	3	Should have

Free access and availability to all	2	4	4	3.3	Should have
Tab to fill in info and address	2	3	3	2.6	Could have
Place for reporting missing objects	3	3	2	2.6	Could have
Agency for keeping missing items	5	2	1	2.6	Could have
Community for reporting missing and found items	2	3	3	2.6	Could have

From the Table above it can be seen that each criteria had a weight of 5 units and the weighted score is gotten by multiplying the ratio of all the scores to the total ratio, then multiplying by 5

$$weighted\ score = \frac{sum\ of\ criterion\ scores}{maximum\ score(15)} \times 5$$

Base on the weighted scored obtained, MoSCoW priorities were assigned as follows

- Weighted score > 4: Must have
- Weighted score > 3: Should have
- Weighted score < 3: Could have
- Weighted score < 1: Won't have

2.3.4. Final Prioritized Requirements

The summery of our prioritized requirements is hence as presented below:

1) Must have:

- Image recognition and Real-time image processing
- User Authentication and Security
- Database Integration
- User Interface Design
- Notification System
- Image uploading

2) Should have:

- Reward system
- Customizable Searches (adding descriptions)
- Feedback Loop
- Location-based Services
- Image Capture integrated in the application
- Display owner and finder Location
- Free access and availability to all

3) Could have:

- Offline mode
- Texting and calling integration
- Tab to fill in info and address
- Place for reporting missing objects
- Agency for keeping missing items
- Community for reporting missing and found items

For this project, as far as functional requirements are concerned, we shall implement all the Must have requirements and as many should have's as possible, keeping aside the could haves till all the completion all of all the should have's. For the non-functional requirements they will serve as a guide to how well our system performs hence shall all be implemented for improved user satisfaction.

2.4. System Requirements Specifications (SRS)

The Software Requirements Specification (SRS) is a foundational document that outlines the objectives, features, and constraints of a software project. Acting as a communication tool between stakeholders and development teams, it ensures a shared understanding of project requirements and serves as a reference throughout the development process. By documenting clear requirements, the SRS facilitates effective project management and ensures the delivery of a high-quality software solution.

2.4.1. Functional Requirements:

Functional requirements define specific features and functionalities of the application. For our system , they include tasks such as image recognition, user authentication, database integration, user interface design, notification system, feedback loop, location-based services, image capture, offline mode, reward system, texting and calling service integration, live location sharing, free access to communication updates, tap to fill in info, reporting missing objects, agency for keeping missing items, image uploading, and customizable searches, and a community for reporting missing and found items. Each requirement should have been described in detail and prioritized in the previous sections of this document.

2.4.2. Non-functional Requirements:

Non-functional requirements define how the system should perform, including aspects such as security, performance, usability, reliability, and scalability. For example, the application should ensure data privacy and security through robust user authentication mechanisms. It should also be user-friendly with intuitive navigation and responsive design for various screen sizes. Additionally, it should be reliable,

able to handle a large volume of image data efficiently, and scalable to accommodate future growth. Like the Functional requirements, the non-functional requirements have also been discussed in detail a previous chapter of this document.

2.4.3. System Interfaces or Integration requirements:

System interfaces specify how the application interacts with external systems, including databases, APIs, and third-party services. For example, the application should integrate with a database for storing images and metadata, as well as with external services for sending notifications and handling communication between users while Integration requirements are crucial for ensuring the smooth functionality by defining interfaces, protocols, and data exchange mechanisms to facilitate communication between different components and external systems. The integration requirements include:

i. Interface Specification: Defining interfaces between the application's frontend, backend, and external services, ensuring compatibility for seamless interaction and data exchange.

ii. External Database Integration: Identifying external databases for storing archived objects and metadata, determining requirements for querying and synchronizing data.

iii. Image Recognition Service Integration: Selecting image recognition services for matching and scoring, defining requirements for processing results and handling communication failures.

iv. Data Exchange Mechanisms: Specifying mechanisms for data exchange with external systems, including RESTful APIs and authentication protocols for data security.

v. Compatibility Testing: Conducting tests to ensure compatibility with various devices, operating systems, and network environments, validating interface and protocol compatibility.

vi. Performance Optimization: Optimizing integration processes to minimize latency, implementing caching mechanisms, and monitoring system performance metrics for optimal responsiveness.

vii. Documentation and Maintenance: Documenting interface specifications, service dependencies, and data exchange protocols for developers and administrators, updating documentation regularly to reflect changes.

Addressing these integration requirements ensures seamless communication with external systems, enhancing the application's functionality for effectively locating missing items. These requirements will be processed in the analysis phase to determine the best integration tools and methods for the application's proper functioning.

2.4.4. User Interfaces and User Experience requirements:

User interfaces describe the visual elements and interactions of the application. This includes screens, menus, buttons, forms, and navigation flows. The user interface should be designed to be intuitive and

easy to use, with clear instructions and feedback for users. Some of the tools used and some outcomes for our project include:

User experience design (UX design) begins at the requirements stage and proceeds through all the stages of development, including the testing and post-release stages. The process of UX design includes research, prototyping, usability testing, and the actual designing part, during which lots of documentation and deliverables are produced. Several stages are involved in the planning of the UI/UX requirements

i. User Personas:

ser personas represent key characteristics of real users, focusing on behavior, thought patterns, and motivation. For our application, we were able to come up with one such personas for a simple user

Example:

- **Name**: Sarah
- **Background**: Sarah is a 35-year-old professional working in a busy corporate environment. She commutes to work daily using public transportation.
- **Characteristics:** Sarah is organized but occasionally forgetful, often misplacing her belongings in the rush of her daily routine. She values efficiency and convenience in the tools she uses.
- **Motivation**: Sarah's primary motivation is to quickly locate misplaced items, such as her wallet or keys, to avoid disruptions to her daily schedule.

ii. User Scenarios:

User scenarios describe the steps a user persona will take to accomplish a specific task. A simple scenario can be seen below

Scenario: Sarah uses the application to report her lost wallet while commuting to work.

Steps:

1. Sarah opens the application on her smartphone.
2. She selects the "Report Lost Item" option.
3. Sarah describes the lost item (wallet) and provides any relevant details.
4. She uploads an image of the lost wallet.
5. Sarah submits the report and receives a confirmation message.

iii. Scenario Map:

Scenario maps compile existing user scenarios into a single document, showing all possible scenarios available at a given moment for every function. Let us consider the simple scenario map for our system shown below:

Scenario Map:

Scenario 1: User reports a lost item.

Scenario 2: User searches for a lost item using image matching.

Scenario 3: User manages their profile and notification settings.

iv. User Story Map:

User story maps arrange user stories into future functions or parts of the application, denoting required functions for a certain sprint or flow. The user story map for our system is as shown below.

User Story Map:

Sprint 1: User Registration

Sprint 2: Lost Item Reporting

Sprint 3: Item Search

Sprint 4: Notification System

v. UX Style Guide:

The UX style guide includes design patterns and describes UI elements and content types, defining rules for arrangement and interaction. We came out with the following preliminary ux style guide

UX Style Guide:

Layout: Consistent header with logo, navigation bar, and search bar.

Color Scheme: Blue and white for a clean and professional look.

Typography: Sans-serif fonts for readability.

Button Styles: Rounded corners with contrasting colors for emphasis.

vi. Site Maps:

Site maps visually represent the connection between all pages and functions of the application. We considered the following site Map for our lost item archival and retrieval system

Site Map:

1. Home Screen
2. Item Search
3. Profile Management
4. Lost Item Reporting

vii. User Flow Schemes:

User flow schemes map the steps users take through the application, depicting the logic of user movement. A simple flow scheme for our system can be seen below:

User Flow Scheme:

Step 1: User opens the app.

Step 2: User selects "Report Lost Item" or "Search for Lost Item."

Step 3: User provides details or uploads an image.

Step 4: User submits the form and receives confirmation.

viii. Wireframes:

Wireframes serve as blueprints for the UI, focusing on functionality rather than visual design. Below is the textual description of our application's wireframes

Wireframe:

1. Home Screen: Header with logo, navigation buttons, and search bar.
2. Report Lost Item Screen: Form fields for item details and image upload.
3. Search Results Screen: Grid layout displaying potential matches.

ix. Usability Testing Reports:

Usability testing gathers feedback on the application's ease of use and functionality, with test results documented for improvement. Example of some reports that can be made include :

Report example

Findings: Users struggled to locate the "Report Lost Item" button on the home screen.

Recommendations: Move the button to a more prominent location for better visibility.

This gives the developing and design team direct knowledge as to where adjustments should be made

Throughout the UX design process, collaboration with product owners, UI designers, and engineers is crucial to ensure alignment with project goals and technical feasibility. Continuously updating and refining documentation ensures that the user experience remains a priority from requirements gathering to post-release iterations.

2.4.5. Data Requirements

Data requirements outline the types of data to be collected, stored, and processed by the software system. These requirements specify data formats, structures, and storage mechanisms, ensuring efficient handling and management of data. Considerations are made regarding data privacy and compliance with relevant regulations to safeguard users' data throughout the system's lifecycle.

Types of Data

- **Images of Lost Items:** The application will collect and store images of lost items captured by users using their mobile devices.
- **Metadata**: Each image entry in the database should include metadata such as timestamp, location, description, and any additional user-provided information.
- **User Authentication Data:** User authentication data, including usernames, passwords, and authentication tokens, should be securely stored to ensure the security of user accounts.
- **User history data:** Data about all searches and reports a user has made on the application
- **User feedback data:** User feedback data for potential app improvement or report of bugs

Data Structures and Schema

- **Centralized Database**: The application should utilize a centralized database to store and manage archived objects efficiently.
- **Relational Schema**: The database schema should include tables for storing images, metadata, user authentication data, and any other relevant information, following standard relational database design principles.

Data Privacy and Security

- **Compliance**: The application should comply with relevant data protection regulations, such as GDPR or CCPA, to ensure the privacy and security of user data.
- **Encryption**: Sensitive data, including user authentication credentials and personal information, should be encrypted both at rest and in transit to prevent unauthorized access.
- **Access Controls**: Role-based access controls should be implemented to restrict access to sensitive data and functionality based on user roles and permissions.

Data Retention Policies

- **Retention Period**: The application should define data retention policies specifying how long archived objects and associated data will be stored in the database.
- **Backup and Recovery**: Regular data backups should be performed to prevent data loss in case of

system failures, with robust recovery mechanisms in place to restore data in the event of a

2.4.6. Performance Requirements:

Performance requirements define the expected performance characteristics of the software system, including factors such as speed, responsiveness, scalability, and resource utilization. These requirements ensure that the system meets user expectations for efficiency and reliability under varying conditions. We can outline some of the following performance requirements for our project

Response Time: Specify the maximum acceptable response time for actions such as uploading images, searching for items, and sending notifications.

Scalability: Determine the system's ability to handle increasing loads of user interactions and data processing without degradation in performance.

Resource Utilization: Define limits for CPU, memory, and network usage to optimize resource allocation and prevent system overload.

Image Processing Speed: Specify performance benchmarks for image recognition and matching algorithms to ensure timely processing of user queries.

Notification Delivery Time: Define the maximum delay for sending notifications to users regarding potential matches or updates on lost items.

Database Query Performance: Set performance targets for database queries to ensure efficient retrieval and storage of image and metadata information.

Usability under Load: Test the application's usability and responsiveness under simulated load conditions to identify and address performance bottlenecks.

System Availability: Guarantee high availability of the application to users, minimizing downtime and ensuring uninterrupted access to critical functionalities.

Performance Monitoring: Implement monitoring mechanisms to track system performance metrics in real-time and identify areas for optimization and improvement.

2.4.7. Security Requirements:

Security requirements encompass measures implemented to protect the integrity, confidentiality, and availability of the software system and its data. These requirements mitigate potential threats and vulnerabilities, safeguarding user information and ensuring compliance with privacy regulations. For our project we could outline the following security measures to be taken

User Authentication: Implement secure authentication mechanisms, such as password-based authentication or biometric authentication, to verify users' identities and prevent unauthorized access.

Data Encryption: Utilize encryption techniques to protect sensitive data, including user credentials, uploaded images, and communication between the application and external services.

Access Control: Enforce access control policies to restrict user access to sensitive functionalities and data based on roles and permissions.

Secure Data Storage: Employ secure storage mechanisms to protect stored data from unauthorized access or tampering, including encryption at rest and secure database configurations.

Secure Communication: Use secure communication protocols, such as HTTPS, to encrypt data transmission between the application and external servers, ensuring data privacy and integrity.

Input Validation: Validate user inputs to prevent injection attacks, such as SQL injection or cross-site scripting (XSS), which can compromise the security of the application.

Security Updates and Patch Management: Regularly update and patch software components to address known security vulnerabilities and mitigate the risk of exploitation.

2.4.8. Constraints:

Constraints are limitations or restrictions that influence the design, development, and implementation of the software system. These constraints may arise from technical, budgetary, schedule, or regulatory factors and impact various aspects of the project's lifecycle. Here are some of the constraints limiting the completion or success of our project

Time Constraint: Limited timeframe for project completion due to academic deadlines or other commitments, necessitating efficient project management and prioritization of tasks.

Technical Constraint: Compatibility with mobile devices and operating systems, adherence to image recognition service APIs, and integration with external databases impose technical limitations on system design and functionality.

Resource Constraint: Limited availability of human resources, expertise, or infrastructure may affect the scalability and performance of the application.

Regulatory Constraint: Compliance with data privacy regulations, such as GDPR or HIPAA, imposes legal constraints on data handling and security measures.

Dependency Constraint: Dependencies on external services, libraries, or third-party APIs may impact project timelines and introduce risks related to service availability or compatibility.

User Constraint: User preferences, expectations, and demographics may impose constraints on the user interface design, functionality, and usability of the application.

2.4.9. Assumptions and Dependencies:

Assumptions and dependencies are factors that are not under the direct control of the project team but may impact the development, implementation, and operation of the software system. These assumptions involve making educated guesses about certain conditions or outcomes, while dependencies involve relying on external entities or factors for successful project execution. Some assumptions and dependencies identified for our system include:

1) Assumptions

i) **Users have access to smartphones with internet connectivity to use the application effectively**.

Explanation: The assumption that users possess smartphones with internet access influences decisions regarding the design, functionality, and accessibility of the application. It impacts the choice of features and the level of network dependency for real-time interactions.

ii) Users trust the application to handle their personal data securely and responsibly.

Explanation: The assumption that users trust the application with their personal data influences decisions regarding data privacy, security measures, and transparency in data handling practices.

iii) Users are willing to provide necessary information and images of lost items for effective search and retrieval.

Explanation: The assumption that users are cooperative and willing to actively participate in the search and retrieval process influences the design of user interfaces, information prompts, and feedback mechanisms within the application.

iv) Users have basic literacy and technological competency to navigate and use the application effectively.

Explanation: The assumption that users possess basic literacy and technological skills influences decisions regarding user interface design, instructional materials, and support features to enhance usability and accessibility.

2) Dependencies

i) Integration with third-party image recognition services or libraries for image matching functionality.

Explanation: The application relies on external image recognition services or libraries for accurate matching of lost items based on uploaded images. Dependencies on these services affect system performance, reliability, and compatibility.

ii) Availability and reliability of external databases or repositories for storing archived objects and metadata.

Explanation: The application depends on external databases or repositories for storing and accessing archived objects and associated metadata. Dependencies on these external resources impact data availability, consistency, and security.

iii) Compliance with data privacy regulations, such as GDPR or CCPA, for handling and processing user data.

Explanation: The application is dependent on adherence to data privacy regulations to ensure legal compliance, protect user privacy rights, and avoid potential penalties or liabilities.

3. CONCLUSION

The requirements analysis phase serves as a crucial foundation for the successful development of the Mobile-Based Archival and Retrieval of Missing Objects Application. Beginning with a comprehensive review of the gathering phase, where user needs and expectations were identified, the process transitioned into the specification stage. Here, functional and non-functional requirements were meticulously outlined, ensuring clarity and alignment with project objectives.

Prioritization and classification of requirements further streamlined the focus, enabling the team to allocate resources efficiently and address high-priority needs first. Through weighted scoring and the MoSCoW method, critical functionalities were identified, ensuring that essential features are addressed within the project's constraints and limitations.

Subsequently, the creation of the Software Requirements Specification (SRS) document encapsulated the findings of the requirements analysis phase, providing a detailed blueprint for the development team. This comprehensive document outlines user personas, scenarios, system interfaces, and performance, security, and usability requirements, among others.

As we transition to the design and implementation phase, the insights gleaned from the requirements analysis phase will serve as guiding principles. They will inform design decisions, development tasks, and testing procedures, ensuring that the resulting application meets user needs effectively and efficiently.

In conclusion, the requirements analysis phase lays the groundwork for the entire software development lifecycle, providing a clear understanding of user needs, system functionalities, and constraints. By diligently analyzing, specifying, prioritizing, and documenting requirements, the project team sets the stage for successful design and implementation, ultimately delivering a robust and user-centric solution for mobile-based archival and retrieval of missing objects.

REFERENCES

1. https://www.lambdatest.com/learning-hub/requirement-analysis date: 12/05/2024

2. “Requirements Engineering in Software Engineering” geeksforgeeks, https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/ date: 12/05/2024

3. https://pm.stackexchange.com/questions/26929/prioritizing-both-functional-and-non-functional-requirements-in-agile date: 12/05/2024

4. https://www.seguetech.com/key-phases-software-development/ date: 12/05/2024

5. https://scand.com/company/blog/functional-vs-non-functional-requirements/ date: 12/05/2024

6. https://medium.com/@growsolutions/functional-and-non-functional-requirements-the-ultimate-checklist-with-examples-cde16aba33d7 date: 12/05/2024