Software Requirements Specification

for

**Secure Data Sharing using Blockchain Technology**

Version 1.0

Prepared by **Team Sigma**

**University of North Texas**

\*Akhila Pam (11711224)

\*Akshara Reddy Bathula (11713259)

\*Jyothi Anjan Manini (11715079)

\*Lakshmichatura Medidi (11682526)

\*Manoj Kumar Bandari (11711378)

\*Nimitha Bangalore Sathyanarayana (11649788)

\*Nitin Reddy Balaiahgari (11698724)

\*Satya Laxman Pranav Vadlamani (11701928)

\*Sumuk Reddy Kalagiri (11702970)

Table of Contents

1. Introduction 3

1.1 Purpose 3

1.2 Document Conventions 3

1.3 Intended Audience and Reading Suggestions 3

1.4 Project Scope 3

1.5 References 3

2. Overall Description 4

2.1 Description of the System 4

2.2 Data Owner Module 4

2.3 Data User Module 4

2.4 Proxy Server Module 4

2.5 Trusted Authority (Blockchain) Module 4

2.6 Cloud Service Provider Module 5

3. Dataflow Diagram Description 5

3.1 Use Case Diagram 5

3.2 Flow Chart 7

4. Functional Requirements 8

4.1 Data Owner Requirements 8

4.2 Data User Requirements 8

4.3 Proxy Server Requirements 8

4.4 Trusted Authority Requirements 9

4.5 Cloud Service Provider Requirements 9

5. Nonfunctional Requirements 9

5.1 Security Requirements 10

5.2 Scalability Requirements 10

5.3 Reliability & Availability Requirements 10

5.4 Performance Requirements 10

5.5 Usability Requirements 10

5.6 Maintainability Requirements 10

6. External Interface Requirements

11

6.1 User Interfaces 11

6.2 Hardware Interfaces 11

6.3 Software Interfaces 11

6.4 Communications Interfaces 11

6.5 APIs Interfaces 11

6.6 Database Interfaces 11

7. Implementation Plan 12

7.1 Development Phase 1: Laying the Foundations 12

7.2 Development Phase 2: Advanced Functionalities & Security Enhancement 12

7.3 Development Phase 3: Refinement & Optimization 13

8. Conclusion 13

Revision History

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

# **Introduction:**

## **Purpose**

The primary goal of this project is to provide a secure data exchange in cloud environments using proxy intermediate key encryption. The project ensures that only authorized users will have access to data on the cloud by integrating identity based proxy encryption and re-encryption between data users and sharers which is managed by proxy servers and cloud service providers.

## **Document Convention**

The entire document is written in Times New Roman with the following font sizes.

Title Dimensions: 12

Other heading size: 11

Content Dimensions: 11

## **Intended Audience and Reading Suggestions**

There are Seven main sections in this document: Introduction, Overall description, Data flow diagram description, functional, Non-functional and External interface requirements, hardware interfaces and Implementation plan. Each section is further segmented into subsections. If the project undergoes any implementation changes in the future, this page can be utilized as a reference for any further clarification.

## **Project Scope**

Our system paradigm presents a PRE approach to data sharing based on blockchain technology. The application devices act as proxy points, re-encrypting data for the verified user(s). Edge devices provide consumers with excellent availability and performance as data is stored in the network’s cache. The users acquire the re-encryption key provided by the data owner, obtain the encrypted text from the CSP, and convert it depending on the identity of the individual or entity accessing it(i.e data user).

## **References**

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A survey on enabling technologies, protocols, and applications,” IEEE Commun. Surveys Tut., vol. 17, no. 4, pp. 2347–2376, Oct./Dec. 2015.

[2] M. Blaze, G. Bleumer, and M. Strauss, “Divertible protocols and atomic proxy cryptography,” in Proc. Int. Conf. Theory Appl. Cryptographic Techn., Springer, May 1998, pp. 127–144.

[3] A. Shamir, “Identity-based cryptosystems and signature schemes,” in Proc. Workshop Theory Appl. Cryptographic Techn., Springer, Aug. 1984, pp. 47–53.

[4] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” in Proc. Int. Conf. Theory Appl. Cryptographic Techn., Springer, May 2004, pp. 506–522.

# **Overall Description**

The system being developed is a sophisticated, multi-component solution intended to facilitate secure and efficient data management and access, leveraging technologies like encryption, blockchain, and cloud services. The system is designed to ensure that data owners can securely upload their data, data users can access the required data with proper permissions, and any interaction or transaction between different entities is securely logged and managed through a blockchain.

## **.Description of the System**

The system is meticulously crafted to integrate different modules seamlessly, each having a distinctive role, interacting harmoniously to form a secure, robust, and user-friendly environment. Below is a detailed description of each module and its respective functionalities within the system.

## **.Data Owner Module**

The Data Owner Module is crucial as it’s the point of interaction for the owners of the data. The owners register and log into the application, encrypt their data and upload it into the cloud. This module is pivotal for maintaining the integrity and security of the data, with functionalities that enable owners to view all uploads, and send re-encrypt requests to the proxy server with user identity, ensuring that the data is not only secure but also accessible as and when required.

## **.Data User Module**

The Data User Module is where registered users interact with the system to search for and request access to data. After successful login, the users can perform several operations such as sending requests to the blockchain, viewing responses, decrypting, and downloading files. The design of this module is user-centric, focusing on intuitive interfaces and responsive functionalities, ensuring a smooth user experience while maintaining the highest standards of data security.

## **.Proxy Server Module**

The Proxy Server is the intermediary module acting between data owners, data users, and the cloud service provider. This module receives re-encrypt requests from data owners, fetches the corresponding response (like URL) from the cloud, and uploads the necessary data to the blockchain. This server is paramount for mediating interactions and ensuring that every transaction and data exchange occurs in a secure and efficient manner.

## **.Trusted Authority (Blockchain) Module**

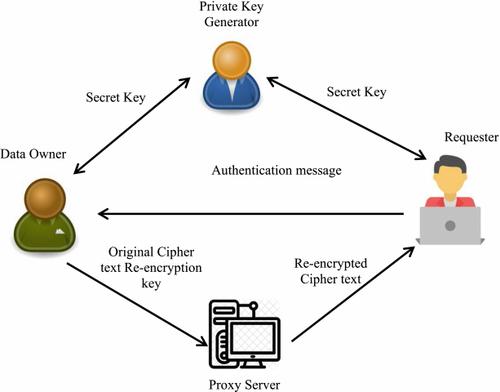
The Trusted Authority operates primarily through blockchain, ensuring the security and integrity of every interaction within the system. This module has the capability to log in directly and execute various operations, such as viewing registered users and owners, generating membership keys, and viewing all submitted data by the proxy server. The utilization of blockchain technology in this module ensures that every transaction and interaction is securely logged and verifiable, providing a transparent and immutable record of all activities within the system.

## **.Cloud Service Provider Module**

The Cloud Service Provider Module is the repository where all the encrypted files are stored and managed. It provides functionalities to view all files and interactions, monitor encryption, re-encryption, and decryption times, and observe all download graphs and attacked file graphs. This module is crucial for the secure and efficient storage of data, providing the necessary infrastructure to support the extensive functionalities of the system.

# **Data Flow Diagram Description**

The Data Flow Diagram (DFD) for this project represents a coherent visualization of how data traverses and is transformed within the system. It details the journey of encrypted data from Data Owners to the Cloud Service Provider and illustrates how data requests from Data Users are processed and fulfilled. The Proxy Server’s role as an intermediary in managing the flow of data and interfacing with the blockchain is highlighted, along with the Trusted Authority's responsibility in maintaining system integrity by verifying and logging transaction data. Through this diagram, the operational logic and the seamless flow of information between distinct modules of the system are concisely represented, providing a clear perspective on the system’s information management and processing mechanics.



## **.Use Case Diagram**

For this project, below diagram would illustrate the multifaceted interactions and functionalities within the proposed system. The actors—Data Owners, Data Users, Proxy Server, Trusted Authority (Blockchain), and Cloud Service Provider—are depicted with unique use cases demonstrating the spectrum of their interactions and responsibilities within the system. Data Owners and Data Users engage in uploading, requesting, and retrieving data, while the Proxy Server mediates, manages requests, and uploads data. The Trusted Authority maintains the system's integrity through logging and verifying transactions, and the Cloud Service Provider orchestrates the management of encrypted files, thus showcasing a coherent visual representation of functional relationships and interactions in the system.

A diagram of a network

Description automatically generated

## **.Flow Chart**

A flowchart is a sort of diagram that illustrates a process or activity. A diagrammatic description of an algorithm, or a step-by-step process for addressing a problem, is another definition of a flowchart. The flowchart displays the processes as various types of boxes and their order by joining the boxes together using arrows. A solution model to a specific problem is shown in this diagrammatic representation. In many different fields, flowcharts are used for process or program analysis, design, documentation, or management.

A diagram of a login check

Description automatically generated

# **Functional Requirements**

Each of the below requirements is imperative to achieve the intended functionality and seamless operation of the system, ensuring secure and efficient interactions between different components and users.

## **.Data Owner Requirements**

REQ-1: Registration & Login

The system should facilitate a secure registration and login functionality for the data owner, employing secure hash algorithms to protect passwords. New users should be able to create an account by providing a valid email address, username, and password.

REQ-2: Encrypt & Upload Files

Once logged in, the data owner should have the ability to upload files. The system should allow the uploading of text files of different sizes. Before uploading to the cloud, files must be encrypted using secure encryption algorithms to protect the information. [TBD: Whether files of different formats such as mp3 (audio/video)files, Image files (png, jpg) files can be uploaded or not.]

REQ-3: View Uploads

Data owners should have a clear and concise dashboard that lists all their uploaded files with metadata such as upload date, file size. They should also have the ability to filter and sort this list based on the metadata.

REQ-4: Send Re-encrypt Requests

Data owners should be able to send requests for re-encryption to the proxy server, providing adequate information such as user identity and file details. The process should be secured and authenticated, preventing unauthorized access and alterations.

## **.Data User Requirements**

REQ-1: Registration & Login

Similar to data owners, data users must have a secure and user-friendly interface for registration and login, ensuring the protection of user credentials through encryption.

REQ-2: File Search & Requests

Data users, post-login, should have the capability to search for files based on specific criteria and send requests for access. The interface should be intuitive, with clear indications of request statuses and responses.

REQ-3: View & Decrypt Files

Upon receiving access, data users should be able to view and decrypt files. The decryption should be secure, and users should have the option to download the decrypted file in its original format.

## **.Trusted Authority(Blockchain) Requirements**

REQ-1: Login

The trusted authority should be able to log in securely, accessing the system with enhanced privileges that are securely managed and restricted to authorized personnel.

REQ-2: View & Manage Data

It should have the capability to view, manage, and monitor user data, file data, and membership keys. This authority should also be able to view all the data submissions from the proxy server and manage them effectively.

## **.Proxy Server Requirements**

REQ-1: Login

The proxy server should have a secure login and should be capable of managing re-encryption requests, ensuring secure and authenticated interactions.

REQ-2: Manage Re-Encrypt Requests

It should receive, process, and manage re-encryption requests, interact securely with the cloud, manage responses, and upload the necessary information to the blockchain securely and efficiently.

## **.Cloud Service Provider Requirements**

REQ-1: Login

The cloud service provider must have secure login functionality, allowing authorized access to manage files, requests, and responses.

REQ-2: Manage Files & Requests

The CSP should have a detailed and secure interface to manage all uploaded files, view their details, and manage the requests and responses related to those files.

REQ-3: Provide Analytics

The CSP should provide robust analytical data on various operations like encryption, re-encryption, and decryption times, presented in an understandable and clear manner through graphs or charts.

# **Nonfunctional Requirements**

## **.Security Requirements**

REQ-1: Data Protection

The system must employ strong encryption algorithms to protect user data during transmission and while at rest. Any data breach attempt should be detected and reported immediately.

REQ-2: User Authentication

It should use secure and reliable methods for user authentication to prevent unauthorized access, possibly employing multi-factor authentication.

REQ-3: Access Controls

There must be strict access controls in place, ensuring that only authorized users can access certain parts of the system.

## **.Scalability Requirements**

REQ-1: Load Management

The system should be able to handle a large number of simultaneous users and file uploads/downloads without degrading performance.

REQ-2: Extension Capability

The architecture should allow for the addition of new features and components with minimal disruption to existing services.

## **.Reliability & Availability Requirements**

REQ-1: High Availability

The system should aim for the highest availability, minimizing downtime and ensuring that users can access the system when needed.

REQ-2: Data Backup & Recovery

Regular backups of the data should be taken, and there should be clear procedures in place for data recovery in case of any loss.

## **.Performance Requirements**

REQ-1: Response Time

The system must respond to user requests, whether for uploading, downloading, or processing data, within acceptable time frames, ensuring a smooth user experience.

REQ-2: Throughput

The system should be able to process a high volume of data efficiently and should be capable of managing multiple transactions simultaneously.

## **.Usability Requirements**

REQ-1: User- Friendly Interface

The UI should be intuitive and user-friendly, enabling users to easily navigate through the system and perform required operations without unnecessary complications.

REQ-2: Accessibility

The system should be accessible from various devices and browsers, ensuring a broad user base can access it.

## **.Maintainability Requirements**

REQ-1: Modularity

The system should be modular to allow for easier maintenance, updates, and the addition of new features.

REQ-2: Documentation

Comprehensive documentation should be maintained for every component and functionality of the system to facilitate maintenance and further development.

# **External Interface Requirements**

## **.User Interfaces**

The user interface is a critical component, acting as the interaction point between the user and the system. It needs to be intuitive and user-friendly. In this project, the user interface will likely consist of various components, such as forms for data input, buttons for triggering actions, and panels for displaying information and results. It will be designed to provide smooth navigation and access to all the functionalities of the system.

## **.Hardware Interfaces**

While this project is primarily software-based, it will interact with the hardware of the host machines. It will require access to the CPU, memory, and storage of the host machine for performing various operations like data processing and storage. The hardware interface will manage these interactions, ensuring optimal performance and resource utilization. The system will be designed to be adaptable to different hardware configurations, allowing it to run on various host machines with varying specifications.

## **.Software Interfaces**

The software interface manages interactions between different software components and layers of the system. It will be critical for integrating various modules, like the data processing module, storage module, and user interface module, allowing them to work together seamlessly. It will also manage interactions with the operating system and any external software components or libraries that the system relies on.

## **.Communications Interfaces**

The communication interface is responsible for managing the interactions between the system and external entities. It is crucial for networking and data exchange between the client and the server, or between different components of a distributed system. In this project, it will manage the secure transmission of data between the user and the system, ensuring data integrity and confidentiality. It will also handle any communication between different modules or subsystems of the project, enabling them to coordinate and work together efficiently.

## **.Application Programming Interfaces(APIs)**

APIs will be leveraged to facilitate interaction between different software components, enabling the exchange of data and functionality. They will be crucial for integrating third-party services or libraries, allowing the system to leverage external functionalities and data. The system will likely expose APIs to allow other systems or services to interact with it, expanding its usability and interoperability.

## **.Database Interfaces**

The database interface will manage interactions with the database, enabling the system to store and retrieve data efficiently. It will be designed to support various database operations like create, read, update, and delete (CRUD), allowing the system to manage user data, transaction data, and any other necessary information. It will also ensure the integrity and consistency of the stored data, preventing data corruption and loss.

# **Implementation Plan**

## **.Development Phase 1: Laying the Foundations**

**Priority**: Focus on creating a sturdy and reliable foundation, establishing core functionalities such as user registration, authentication, and basic file interactions, which are crucial for system operations.

**Task1**: Develop User Registration & Login Modules

* Develop separate registration and login modules for different user types, ensuring a seamless and user-friendly experience.
* Implement secure password storage and retrieval mechanisms, utilizing hashing and salting techniques to protect user credentials.

**Task2**: Implement Basic File Upload and Download Functionalities

* Create interfaces allowing Data Owners to upload files and Data Users to download them, ensuring a smooth and intuitive user interaction.
* Develop a file management system to organize and store uploaded files securely.

**Deliverables**:

* Functional User registration and login modules for all user types.
* Basic file upload and download functionalities.
* Secure and encrypted communication channels between system components.

## **.Development Phase 2: Advanced Functionalities & Security Enhancement**

**Priority**: Concentrate on the development of advanced functionalities such as file encryption and decryption and concentrate on secure file transactions while enhancing security protocols and addressing vulnerabilities.

**Task 1**: Develop File Encryption and Decryption Modules

* Implement advanced encryption algorithms to secure files during transit and storage.
* Develop decryption modules to allow authorized access and interaction with secured files.

**Task 2**: Implement Request and Response Management

* Develop sophisticated request management systems to handle file transactions, ensuring secure and efficient processing of user requests.
* Implement responsive mechanisms to handle transaction responses, providing users with timely and accurate feedback.

**Task 3**: Enhance Security Protocols

* Strengthen security protocols and conduct rigorous security assessments to identify and mitigate potential vulnerabilities.
* Develop contingency plans and implement countermeasures to deal with security breaches and data leaks.

**Deliverables**:

* Advanced file encryption and decryption modules.
* Efficient request and response management for file transactions.
* Enhanced security protocols and mitigation strategies against vulnerabilities.

## **.Development Phase 3: Refinement & Optimization**

**Priority**: Focus on refining the overall system, optimizing performance, enhancing user interactions, and conducting extensive testing to ensure the reliability and stability of the system.

**Task1**: Optimize System Performance

* Conduct comprehensive performance assessments to identify bottlenecks and optimize resource utilization and system responsiveness.
* Refine system components and algorithms for maximum efficiency and reliability.

**Task2**: Refine User Interfaces

* Enhance user interfaces, focusing on usability, accessibility, and user experience, making system interaction intuitive and pleasant.
* Collect user feedback and make necessary adjustments to meet user expectations and needs.

**Task 3**: Conduct Extensive Testing

* Perform rigorous testing on all system components, focusing on functionality, security, performance, and reliability.
* Identify and address any issues or bugs found during testing, ensuring the system is robust and ready for deployment.

**Deliverables**:

* An optimized, efficient, and reliable system.
* Refined and user-friendly interfaces.
* A well-tested, stable, and bug-free system, meeting all specified requirements and quality standards.

# **Conclusion**

The structured and incremental approach of this implementation plan will allow for a systematic and organized development process, enabling the project team to effectively address issues, refine functionalities, and assure the quality and reliability of the final system. Through clear prioritization and detailed task allocation, each development phase builds upon the previous one, resulting in a comprehensive, user-friendly, and secure system.