Software Requirements Specification

for

CLOUDPI

CLOUD BASED STORAGE SYSTEM USING RASPBERRY PI

Version 1.0

Prepared by

A Anas
Aswathy Krishnan
Jayalekshmi
Nandakumar

Team- CloudPi

7 March 2023

Table of Contents

1. Introduction

1.1 Purpose

The purpose of the Software Requirement Specification is to provide a clear, documented model of the requirements for Cloud based storage using Raspberry Pi, this create a cloud-based storage system that allows users to upload their files to a remote server or cloud storage platform, and then automatically copy those files to a personal disk attached to a Raspberry Pi.

1.2 Document Conventions

The standards, typographical conventions or abbreviations that were followed when writing this SRS:

● “CloudPi” - It refers to a cloud based storage using Raspberry pi

2. Intended Audience and Reading Suggestions

People may find it hard to trust online methods and storage services sometimes to permanently store their highly confidential data. That is why there are huge customers for hard disks to store their huge files permanently. So this system helps the users to store and access their data remotely with better safety.

The document is intended to be read by the faculties of cse-tkmce,developers,staff users and documentation writers. Entire document is organized into

1.Introduction
2. Product description
3.Requirements
4. Appendix.

2.1 Product Scope

1. Hardware setup: The product would involve setting up a Raspberry Pi device with the necessary components, such as a hard drive, network connection, and power supply. The hardware setup will enable the Raspberry Pi to function as a cloud-based storage device.
2. Cloud-based storage software: The product would also require the installation and configuration of cloud-based storage software, such as Nextcloud or OwnCloud, on the Raspberry Pi device. This software would allow users to access and store their files and data on the Raspberry Pi device from any location with an internet connection.
3. User management: The cloud-based storage software would also need to include user management capabilities to control access to the stored files and data. This would involve creating user accounts, assigning permissions, and managing user activity.

4. Security: The product should also ensure that the stored files and data are secure. This would require implementing security measures such as encryption, secure authentication, and access controls to prevent unauthorized access.
5. Scalability: The product should be scalable, which means it should be capable of expanding its storage capacity as the demand for storage increases. This would require the ability to add new hard drives or increase the size of existing drives as needed.
6. User interface: The product should also include a user-friendly interface that makes it easy for users to upload, download, and manage their files and data.
7. Support and maintenance: The product should provide reliable support and maintenance services to ensure that the system runs smoothly and is always available to users

2.2 References

https://www.ljnrd.org/papers/IJNRDA001020.pdf
Private Cloud using Raspberry Pi by Bhavani.G, Akshaye J, Christ Annson M, Cyril j wilson

2. Overall Description

2.1 Product Perspective

CloudPi is a cost-effective and flexible way to store and access data remotely. With CloudPi, users can store and access files, photos, music, and videos from anywhere with an internet connection. This can be useful for personal use, or for businesses looking to securely store and share data with remote employees or clients.

2.2 Product Functions

1. File storage and synchronization
2. Web-based user interface
3. User authentication and access control

2.3 User Classes and Characteristics

User classes refer to groups of users who will interact with the system in different ways. For a cloud-based Raspberry Pi storage system, the following user classes can be identified:

System Administrators: These are users who will have complete control over the system. They will be responsible for setting up the system, managing user accounts, configuring the system, and monitoring its performance.

End Users: These are the users who will use the storage system to store and retrieve their data. They may be individual users or members of an organization.

Developers: These are users who will be involved in the development of the system. They may be responsible for developing and maintaining the software that runs on the Raspberry Pi, or for integrating the storage system with other applications.

As for user classifications, they refer to different types of users within each user class. For example, within the end user class, we can have different classifications such as:

Basic Users: These are users who will use the system for simple file storage and retrieval.

Power Users: These are users who will use the system for more advanced tasks such as sharing files with other users, setting permissions, and creating backups.

Business Users: These are users who will use the system for business purposes such as storing and sharing documents, collaborating with team members, and accessing files remotely.

By identifying user classes and classifications, we can better understand the different requirements and needs of each user group, which can then be used to inform the design and development of the system.

2.4 Operating Environment

The operating environment refers to the environment in which the system will be installed and run. For a cloud-based Raspberry Pi storage system, the operating environment can be described as follows:

Hardware: The system will run on a Raspberry Pi, which is a small single-board computer. The Raspberry Pi will be connected to the internet via a network connection. The storage system may also use external hard drives or other storage devices to increase the storage capacity.

Software: The software required for the system will include the operating system (e.g., Raspbian), web server software, database software, and the storage system software itself.

Network: The system will be connected to the internet via a network connection, which may be wired or wireless. The system will need to be configured to work with the network, including setting up firewall rules and configuring network settings.

Power: The Raspberry Pi will need a stable power supply to operate properly. This may be provided by a wall adapter or a battery backup.

Security: The system will need to be secured to prevent unauthorized access and protect user data. This may include using encryption, setting up secure user authentication, and implementing access controls.

By understanding the operating environment, we can identify any hardware or software dependencies, network requirements, power constraints, and security considerations that need to be taken into account when developing and deploying the system.

2.5 Design and Implementation Constraints

- Hardware constraints: The system needs to be designed to work with a specific hardware configuration, such as the Raspberry Pi. This could include limitations on processing power, memory, and storage capacity.
- Software constraints: The system needs to be designed to work with specific software components, such as the Python programming language and the Django web framework.
- Time constraints: There may be limitations on the amount of time available to complete the project, which could affect the design and implementation choices made.
- Security constraints: The system needs to be designed to protect against potential security risks, such as unauthorized access to sensitive data.
- Usability constraints: The system needs to be designed to be user-friendly and easy to use, with a clear and intuitive interface.
- Scalability constraints: The system needs to be designed to handle a potentially large number of users and files, and to scale up as necessary.
- Budget constraints: There may be limitations on the amount of resources available to implement the system, which could affect the design and development choices made.

2.6 User Documentation

- User manual: A step-by-step guide on how to use the application, including how to upload files, delete files, and download files.
-
- Troubleshooting guide: A list of common issues that users might encounter while using the application, along with instructions on how to resolve them.
-
- FAQ section: A list of frequently asked questions that users might have about the application, along with detailed answers.
-
- Contact information: Information on how users can contact the support team if they have any issues or questions that are not addressed in the user manual or troubleshooting guide.
-
- Glossary: A list of technical terms and definitions that are used in the application.

2.7 Assumptions and Dependencies

Assumptions:

- Users have access to a web browser and an internet connection to use the application.
- Users have basic knowledge of how to upload and delete files.
- The remote storage system is available and accessible to the application.
- The Raspberry Pi is properly set up and configured to receive and process the uploaded files.
- The uploaded files will not exceed the available storage capacity of the remote storage system or the Raspberry Pi.

Dependencies:

- The application relies on the Django web framework and its built-in functionality for handling file uploads and forms.
- The application depends on the remote storage system's API or SDK to programmatically upload and manage files.
- The Raspberry Pi must have the necessary software and libraries installed to process the uploaded files.
- The application may depend on third-party packages or modules for specific features or functionality.

3. External Interface Requirements

3.1 User Interfaces

- The design and layout of the user interface
- The navigation flow of the application
- The user input and output formats
- The use of multimedia elements such as images, videos, and animations
- The error messages and feedback mechanisms to guide the user
- The accessibility features such as support for assistive technologies and alternative input methods.

3.2 Hardware Interfaces

- Raspberry Pi model and specifications
- Any additional hardware components required for the project such as a camera module, sensors, etc.
- Operating system requirements for the Raspberry Pi
- Network interface requirements (e.g., Ethernet, Wi-Fi, Bluetooth)

3.3 Software Interfaces

- Operating system interfaces: Your project may need to interact with the operating system running on the Raspberry Pi to access files, manage network connections, and perform other tasks.

- Web framework interfaces: Your project is built using Django web framework, so you will need to ensure that your application interacts correctly with Django's software interfaces, such as its database ORM and HTTP request/response handling.
- Web server interfaces: Your project will be served using a web server such as Apache or Nginx, so you will need to ensure that your Django application interacts correctly with the web server software interfaces.
- Python library interfaces: Your project may use third-party Python libraries to perform various tasks, such as managing network connections or interacting with hardware components. You will need to ensure that your application interacts correctly with these libraries' software interfaces.

3.4 Communications Interfaces

- HTTP: The Django web application will communicate with the client's web browser over HTTP.
- TCP/IP: The Raspberry Pi will communicate with the Django web application over TCP/IP protocol.
- DNS: The project may require domain name system (DNS) to map the IP address to a domain name

4. System Features

4.1 File management:

4.1.1 Description and Priority

The file management feature allows users to upload, download, and delete files, and implements access control to restrict access to files based on user permissions.

4.1.2 Stimulus/Response Sequences

Stimulus: User selects a file to upload.
Response: System prompts user to select the location for the file upload and verifies that the file meets upload restrictions.
Stimulus: User confirms file upload.
Response: System saves the file to the selected location.
Stimulus: User attempts to download a file.
Response: System verifies that the user has permission to access the file and allows the user to download the file.
Stimulus: User attempts to delete a file.
Response: System verifies that the user has permission to delete the file and removes the file from the system.

4.1.3 Functional Requirements

Implement file upload restrictions, such as limiting the size and type of files that can be uploaded.
Implement access control to restrict access to files based on user permissions.
Implement file versioning to keep track of changes to files over time.

4.2 User Authentication

4.2.1 Description and Priority

The user authentication feature ensures that only authorized users can access the system and perform actions such as uploading, downloading, or deleting files.
This is of high priority.

4.2.2 Stimulus/Response Sequences

Stimulus: User attempts to access the system or perform an action.
Response: System prompts user to enter a valid username and password.
Stimulus: User enters valid credentials.
Response: System grants user access to the system or allows user to perform the requested action.
Stimulus: User enters invalid credentials.
Response: System denies user access and displays an error message.

4.2.3 Functional Requirements

Implement a login page that prompts users to enter a username and password.
Implement a mechanism for creating and managing user accounts with different levels of access.
Ensure that all user passwords are stored securely and encrypted.

4.2 Encryption and Security

4.3.1 Description and Priority

The encryption and security feature implements mechanisms to protect data in transit and at rest, and to prevent and detect unauthorized access attempts.

4.3.2 Stimulus/Response Sequences

Stimulus: User attempts to access the system or perform an action.

Response: System verifies that the user is authorized and grants access or allows the action to be performed.
Stimulus: System detects an attempted unauthorized access.
Response: System logs the attempted access and alerts the system administrator.

4.3.3 Functional Requirements

Implement encryption mechanisms such as HTTPS, SSL, or TLS to protect data in transit.
Encrypt data stored on the Raspberry Pi to protect against unauthorized access.
Implement a firewall and intrusion detection system to prevent and detect unauthorized access attempts.
Regularly backup all data stored on the Raspberry Pi to prevent data loss in the event of hardware failure or other disasters.

4.4 User Interface

4.4.1 Description and Priority

The user interface feature provides an easy-to-use web interface for users to interact with the system and manage their files. This has less priority only.

4.4.2 Stimulus/Response Sequences

Stimulus: User logs into the system.
Response: System displays the user's dashboard, including their files and any notifications.
Stimulus: User searches for a file.
Response: System displays search results based on the user's query.
Stimulus: User selects a file to download or delete.
Response: System verifies that the user has permission to perform the selected action and allows the action to be performed.

4.4.3 Functional Requirements

Implement a web interface that is easy to use and navigate.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1. File transfers should be fast and reliable, with minimal data loss or corruption.
2. The system should be able to handle large files and file volumes without performance degradation.

3. The Raspberry Pi should be able to handle the processing and data storage requirements of the system without becoming overloaded or crashed.

5.2 Security Requirements

1. All user authentication should be done securely to prevent unauthorized access to the system.
2. Files and data should be encrypted for maximum security.
3. The system should be protected against malicious attacks and data breaches.

5.3 Safety Requirements

Some requirements related to safety for a cloudPi:

Authentication and access control: The system should require users to authenticate themselves before accessing any files or performing any actions. This requires implementing a user authentication system, such as a login page that prompts users to enter a username and password.

Encryption: Data stored on the Raspberry Pi should be encrypted to protect against unauthorized access or theft. This requires implementing encryption mechanisms.

Compliance with regulations: The system should comply with any relevant regulations or policies related to data privacy and security.

5.4 Software Quality Attributes

1. Reliability: The system should be reliable and available at all times with minimum downtime. The software should be designed to handle any failures or errors that may occur during operation.
2. Security: The system should have proper security measures in place to ensure data privacy and protection. The software should have strong authentication and encryption mechanisms to prevent unauthorized access.
3. Scalability: The system should be able to handle a large volume of data and users without any performance degradation. The software should be designed to handle multiple requests and load balancing.
4. Usability: The system should be easy to use and navigate. The software should have a simple and intuitive user interface that is easy to understand and use.
5. Performance: The system should be able to process data quickly and efficiently. The software should be optimized for performance to ensure fast response times.
6. Compatibility: The system should be compatible with various devices and platforms. The software should be designed to work with different operating systems, browsers, and devices.
7. Maintainability: The system should be easy to maintain and update. The software should be modular and well-documented to facilitate maintenance and updates.

8. Portability: The system should be portable and able to run on different hardware and software environments. The software should be designed to be platform-independent and easily deployable

5.5 Business Rules

Operating principles include

1. User authentication: Only authorized users should be able to access the files stored on the Raspberry Pi. This requires implementing user authentication, such as a login page that prompts users to enter a username and password. Users should have different levels of access depending on their role or permissions.

2. Data privacy and security: Users' data should be kept private and secure. This requires implementing security measures such as encryption, firewalls, and backups to protect against data breaches or loss.

3. Logging and monitoring: The system should log all user activity, including file uploads, downloads, and deletions. This allows for easy monitoring and auditing of user behavior, as well as detecting and responding to any security incidents.

These operating principles imply certain functional requirements, such as implementing a user authentication system, setting file upload and access restrictions, implementing encryption and other security measures, and implementing logging and monitoring functionality.