VAULT

A shared distributed and redundant storage solution

Project ID: 19-002

IT 16106420 – T.R.N.R. Peiris

Structure of the SRS
(Software Requirements Specification)

B.Sc. Special (Honors) Degree in Information Technology

DECLARATION

I, T.R.N.R. Peiris declare that this is my own work and this software requirement specification document does not incorporate with acknowledge any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

……………………………………..
T.R.N. R Peiris

Table of Contents

Table of Figures

List of Tables

1 Introduction

1.1 Purpose

The purpose of this SRS document is to describe the requirements and the process related to redundancy handling and hardware monitoring module. The document will explain the purpose, features, functional and non-functional requirements, design constraint, project approach, constraint under which above mentioned module must operate and how the module will interact with the other modules and the external applications. This document is designed not only with the intention of proposing the solution to a customer, in order to get the approval but also to give an idea to the developers and the other stakeholders about what are the functions available, in what order they needed to be done and the boundaries within which they need to work.

1.2 Scope

The component which will be discussed in this document is called “Redundancy handling and hardware monitoring module”. The component mainly contains two sperate processes, redundancy handling process and hardware monitoring process. Main goal of the redundancy handling process is to use minimum storage space when storing files in a distributed manner while maintaining a higher amount of redundancy without needing to waste any extra storage space for backups, which make the final solution as a unique product when comparing with the existing file systems. To achieve that goal file system uses the concept of erasure coding. When a single file is segmented, M number of parity files will be generated with the other N number of data segments, while maintaining $M = N/2$, which allows the system to recreate the original file even M number of nodes in the system are down at the recreation time. Main goal of the hardware monitoring process is to allows administrative users of the system to check whether what are the nodes that are active/inactive at a certain time, and to check the disk health of every node to maintain consistency and finally present the relevant details to the administrative users in an interactive manner.

1.3 Definitions, Acronyms, and Abbreviations

Table 1:Definitions and Abbreviations

OS	Operating System
SMART	Self-Monitoring, Analysis, and Reporting Technology
API	Application Programming Interface

1.4 Overview

Remainder of this document mainly can be divided into two sections plus appendix. First section is called overall description and it provides readers an understanding about the overall functionalities of the component, and the interactions of the component with the other components. Future more this section describes functional and nonfunctional requirements, design constraints which includes user interfaces, system interfaces, hardware interfaces and system constraints.

Second section which called specific requirements provides requirements specifications in a detailed manner. Specify requirements clearly for different audiences to understand by using various kinds of specification techniques. Future more software system attributes and performance requirements will be discussed in this section.

2 Overall Descriptions

Redundancy handling and hardware monitoring module

As mentioned before mainly two functions will be discussed in this document called redundancy handling and hardware monitoring. Redundancy function allows users to store their data in a redundant and distributed manner, where each and every file will be divided in to, N number of data segments plus M number of parity segments, which will be generated automatically using erasure coding concept, where M=N/2. Altogether K (K= N + N/2) number of files will be output from the system and they will be scattered among K number of nodes available in the network. Advantage of the above-mentioned process is that even though N/2 number of nodes are down from the K number of nodes at the downloading time, the original file can be regenerated. Hardware monitoring function assist in the process of regenerating the original file to check whether a certain device is online or not and also give an overview about hard disk health to the administrative users, which will help them in the process of maintain consistency.

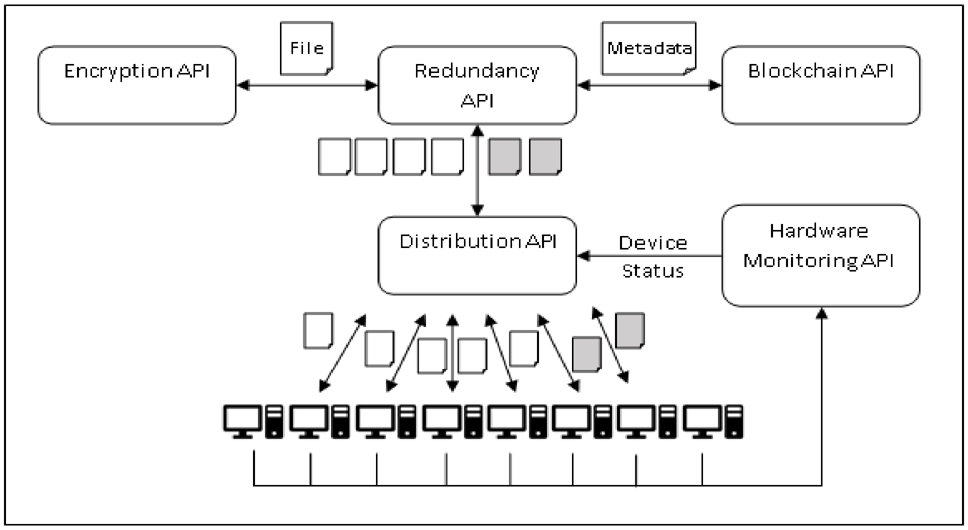


Figure 1: Redundancy handling and hardware monitoring component overview

2.1 Product perspective

The table below describes the similarities and the differences of the final product, not only the components mentioned in this document, with the existing products in the market.

Table 2:Feature Comparison

Features	**MooseFS** [7]	**IBM Spectrum Scale** [6]	**OCFS 2** [4]	**OrangeFS** [5]	**BWFS** [1]	**Minio** [3]	**Ceph** [2]	**VAULT**
High Availability	✓	✓	✓	✓		✓	✓	✓
Scalability	✓	✓	✓	✓	✓	✓	✓	✓
Minimal Investment	✓			✓		✓	✓	✓
Big Data Support	✓	✓	✓	✓		✓	✓	✓
Data encryption		✓				✓	✓	✓
Data Recovery	✓	✓		✓	✓	✓	✓	✓
Platform independent	✓	✓						✓
Security		✓	✓	✓		✓	✓	✓
Data Redundancy					✓	✓		✓
Minimum additional storage for backup								✓
Blockchain integration								✓
Easy to setup and run								✓
File Sharing								✓

2.1.1 System interfaces

Since the system will be designed and developed according to the module-based approach each research component will be designed, developed and tested out individually as modules which then can be imported and assemble the complete software. Since JavaScript will be used as the developing language application can be run inside any Linux, MacOS and Windows environments (Platform independent). Since the final product and the component mentioned in this document both acts as standalone solutions final product, and the component mentioned in this document won't be interacting with the other existing applications other than the web browser which will be needed to display the user interfaces of the web-application. Although hardware monitoring function will be interacting with the OS build in features such as SMART to gather hardware health related data.

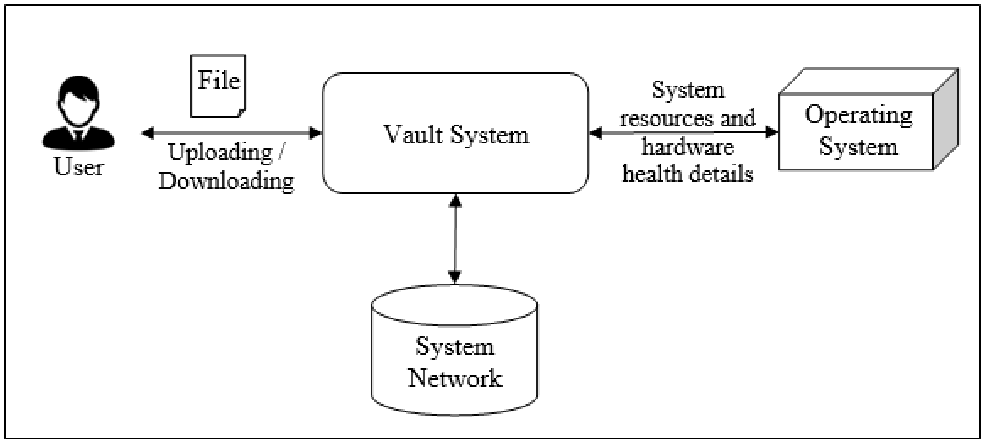
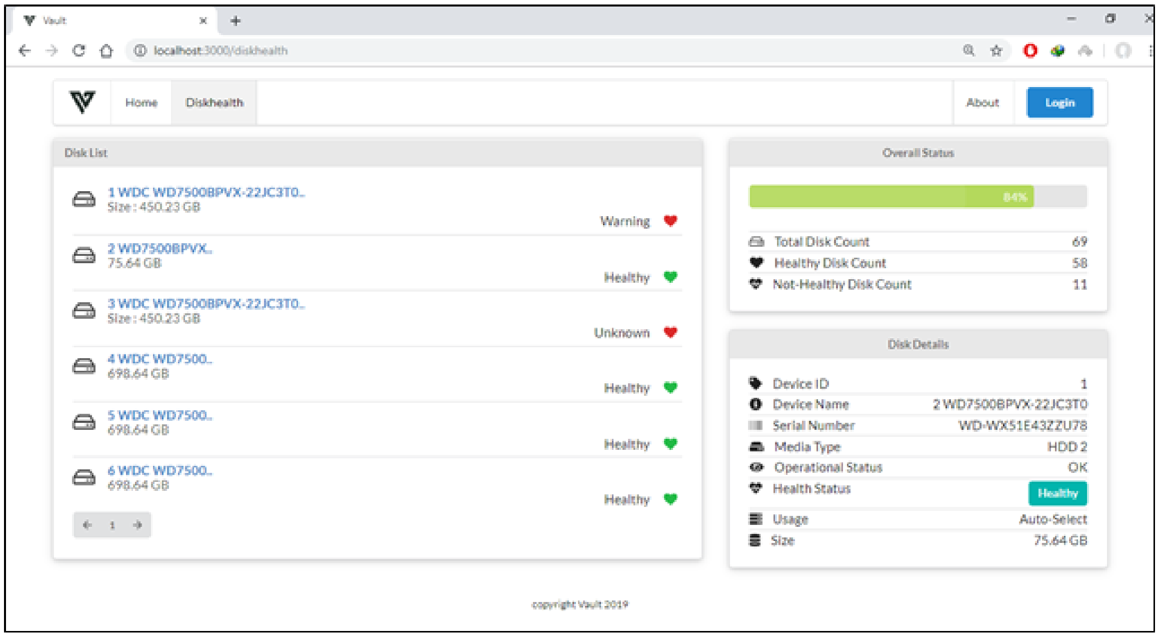


Figure 2:System Overview

2.1.2 <u>User interfaces</u>

Research component described in this document primarily can be divided into two specific functions redundancy function and hardware monitoring function, as mentioned before in this document. Since the redundancy function is a core module which acts in the middle of the file uploading and downloading process it doesn't requires any specific interface, it takes/gives data from/to encryption module and then send/receive them to/from distribution module, hence the interface which is responsible for triggering the redundancy function or else the interface which will be used by the users to upload and download files will be included as the redundancy function interface and will be mention in the section 3.

Hardware monitoring function gather data from the Operating Systems through the network and displayed them in an interactive manner to the administrative users.



Disk List

1 WDC WD7500BPVX-22JC3T0.. Size : 450.23 GB	Warning ♥
2 WD7500BPVX.. 75.64 GB	Healthy ♥
3 WDC WD7500BPVX-22JC3T0.. Size : 450.23 GB	Unknown ♥
4 WDC WD7500.. 698.64 GB	Healthy ♥
5 WDC WD7500.. 698.64 GB	Healthy ♥
6 WDC WD7500.. 698.64 GB	Healthy ♥

← 1 →

Overall Status

84%

Total Disk Count	69
Healthy Disk Count	58
Not-Healthy Disk Count	11

Disk Details

Device ID	1
Device Name	2 WD7500BPVX-22JC3T0
Serial Number	WD-WX51E43ZZU78
Media Type	HDD 2
Operational Status	OK
Health Status	Healthy
Usage	Auto-Select
Size	75.64 GB

copyright Vault 2019

Figure 3: Disk health monitoring interface

2.1.3 Hardware interfaces

Since the software does not directly interact with Hardware level components and doesn't rely on how storage devices handle data, there is no requirement to document hardware interfaces.

2.1.4 Software interfaces

Since the application developed using module-based approach, the component mentioned in this document was mainly developed as two modules, Redundancy API and the Hardware monitoring API. Also, both of those mentioned APIs act as middle layer processes, hence they won't be storing any permanent data or else won't be connecting with external applications other than the OS build in features such as SMART to gather disk health related information. Only requirement needs to run the component successfully is a physical machine with NodeJS installed on them (Platform Independent).

2.1.5 Communication interfaces

Both APIs as mentioned in the above section act as middle layer processes, hence they won't be needing any external communication interfaces other than the interfaces needed to communicate with the other APIs within the application. Redundancy handling module will be needing two specific internal interfaces to connects with the encryption module and the distribution module. Hardware monitoring module will be mainly needing two communication interfaces as well, to communicate with the internal module called distribution module and another interface to communicate with the OS.

2.1.6 Memory constraints

Since the application is engineered to use disk space more than Memory, a machine with at least 2GB is sufficient.

2.1.7 Operations

Prior to use, Docker application will be deployed among the users by using the internal network or else physically. Application basically have two types of user accounts called administrator accounts and normal accounts.

All the user account related activities will be carried out by the administrative users using the web application through the web browsers. Administrators can connect/ disconnect workstations form/to the system, can view available workstations at a certain time and can view the disk health information related to all the workstations connected. Administrators most important task is to assign storage space from the workstations to create the storage pool where all the data will be stored.

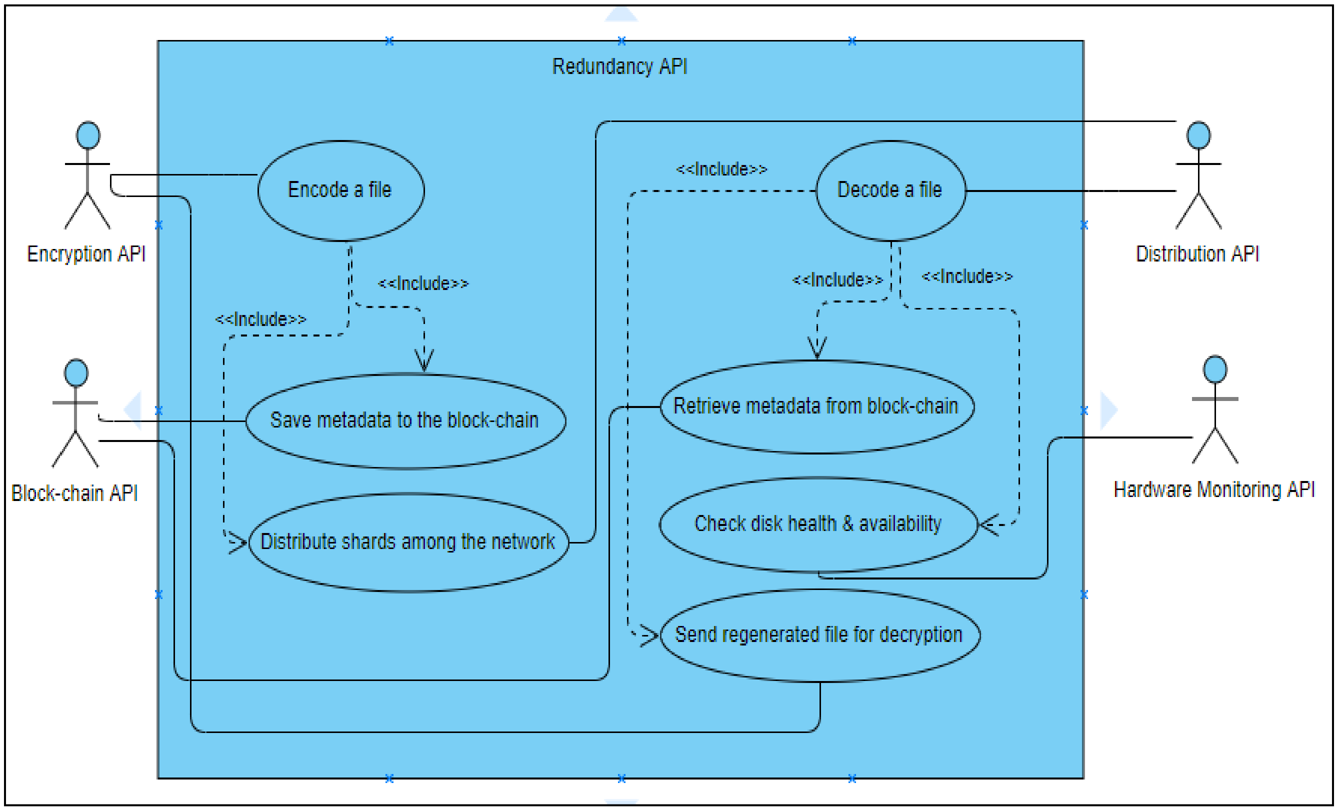
When a normal user finally can access the web application from the web browser a logging page will be displayed to the user, and users can use the logging credentials provided by the administrators to logged in to the application. When a normal user can login they can upload, download files to the system which will be stored among the network in a secure and redundant manner.

2.1.8 Site adaptation requirements

In order to run the application users must complete few tasks in a certain order which will be presented below in the order that users must carry them out. When users accessing the application, they will be guided using English language.

1. NodeJS must be installed
2. Docker application which contains the system must be deployed
3. The Docker image must contain Rethink-DB
4. All the workstations must connect to the local network.
5. At least there must be 6-10 users that can contribute 10 GB storage space by each
6. Web browser must be installed
7. Users must have login credentials to access the web application

2.2 Product functions



Redundancy API
Encryption API
Block-chain API
Distribution API
Hardware Monitoring API
Encode a file
Decode a file
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
Save metadata to the block-chain
Distribute shards among the network
Retrieve metadata from block-chain
Check disk health & availability
Send regenerated file for decryption

Figure 4:Redundancy API use case diagram

Encode a file

Table 3:Encode a file use case scenario

Use Case No	01
Use Case	Encode a file
Actors	Encryption API
Pre-Conditions	
Flow of Event	1. Get the file details and send metadata to the blockchain 2. Scatter the file 3. Generate parity files 4. Distribute files among the network
Post Conditions	Trigger Distribution API
Alternatives	1. Return error message if file is not available 4. Return error message if Distribution API is not available

Save metadata to blockchain

Table 4: Save metadata to block-chain use case scenario

Use Case No	02
Use Case	Save metadata to block-chain
Actors	Redundancy API
Pre-Conditions	File metadata must be collected and validated
Flow of Event	1. Call Blockchain API 2. Send metadata to the blockchain API
Post Conditions	Trigger distribution API
Alternatives	1. Return an error message if blockchain API is not available

Distribute shard among the network

Table 5: Distribute shards among the network use case scenario

Use Case No	03
Use Case	Distribute shards among the network
Actors	Redundancy API
Pre-Conditions	File must be scattered in a redundant manner
Flow of Event	1. Scatter file in to small data shards 2. Generate parity shards 3. Trigger Distribution API 4. Distribute scattered files among the network

Post Conditions	Scatter distributed files among the network
Alternatives	3. Return an error message if Distribution API is not available

Decode a file

Table 6:Decode a file use case scenario

Use Case No	04
Use Case	Decode a file
Actors	Encryption API
Pre-Conditions	Shards of the requested file must be scattered among the network
Flow of Event	1. Retrieve metadata from the blockchain 2. Trigger distribution API 3. Check the available files 4. Regenerate the original file 5. Send original file to the encryption API
Post Conditions	Trigger Encryption API
Alternatives	1. Return error message if the blockchain API is not available 2. Return error message if the distribution API is not available 3. If the files aren't available return an error message 4. Return an error message if the encryption API is not available

Retrieve metadata from blockchain

Table 7: Retrieve metadata from blockchain use case scenario

Use Case No	05
Use Case	Retrieve metadata from blockchain
Actors	Redundancy API
Pre-Conditions	Details of the requested file must be stored inside the blockchain
Flow of Event	1. Trigger Blockchain API 2. Request file metadata
Post Conditions	Trigger hardware monitoring API
Alternatives	1. Return error message if blockchain API is not available

Check disk health & availability

Table 8:Check disk health & availability use case scenario

Use Case No	06
Use Case	Check disk health & availability
Actors	Redundancy API
Pre-Conditions	Workstation must be physically connected to the network
Flow of Event	1. Trigger disk monitoring API 2. Request disk related details from the disk monitoring API
Post Conditions	Disk health API must call Distribution API
Alternatives	1. Return an error message if disk monitoring API is not available

Send regenerated file for decryption

Table 9: Send regenerated file for decryption use case scenario

Use Case No	07
Use Case	Send regenerated file for decryption
Actors	Redundancy API
Pre-Conditions	File can be regenerated
Flow of Event	1. Generate the original file 2. Trigger encryption API 3. Verify hash 4. Send the file
Post Conditions	Send the file for decryption
Alternatives	1. If the original file cannot be regenerated return an error message 2. Return an error message if the Encryption API is not available Return an error message if the hash verification fails

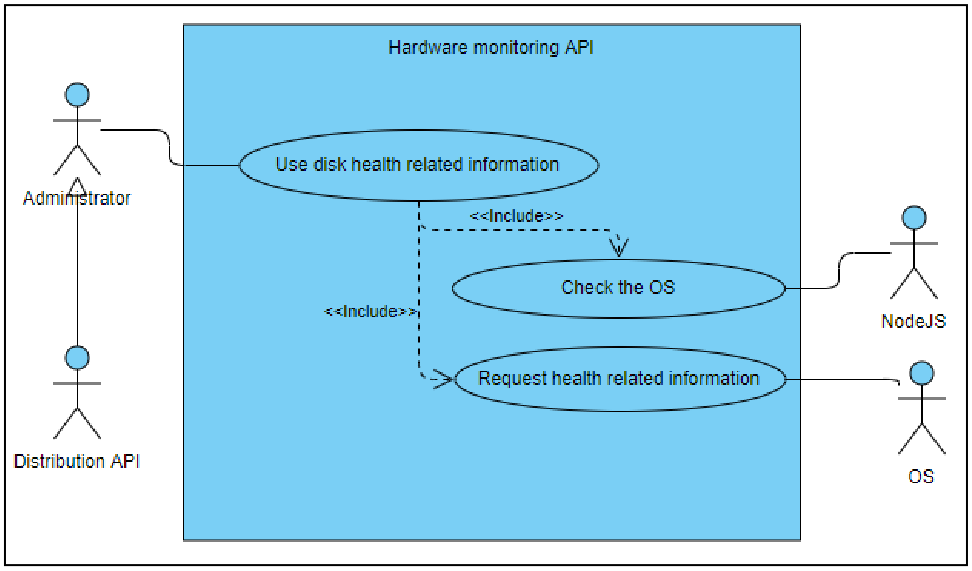


Figure 5:Hardware monitoring API use case diagram

Use disk health related information

Table 10: Use disk health information use case scenario

Use Case No	08
Use Case	Use disk health related information
Actors	Administrator, Distribution API
Pre-Conditions	Workstation must be physically connected to the system.
Flow of Event	1. Check the OS 2. Request health information from the OS 3. Analyze information 4. Send the information
Post Conditions	
Alternatives	1. Return error if the OS cannot be detected 2. Return an error if the health information is not available

Check the OS

Table 11: Check the OS use case scenario

Use Case No	09
Use Case	Check the OS
Actors	Administrator, Distribution API

Pre-Conditions	Workstation must be physically connected to the system.
Flow of Event	1. Use NodeJS to detect the OS
Post Conditions	
Alternatives	1. Return error if the OS cannot be detected

Request health related information

Table 12: Request health related information use case scenario

Use Case No	10
Use Case	Request health related information
Actors	Administrator, Distribution API
Pre-Conditions	Workstation must be physically connected to the system, and the OS must be identified properly.
Flow of Event	1. Request health information from the OS 2. Send the information to the requester
Post Conditions	
Alternatives	1. Return error if the information is not available

2.3 User characteristics

There can be mainly two types of users in the system administrators and normal users.

- Administrator- Software/hardware professional who will be configuring the system and maintain the consistency of the system.
- Normal Users- Anyone with the basic knowledge of computing who will be using the system to store their files and download them when needed.

2.4 Constraints

Redundancy handling module constraints

- Module to be run properly at least there must be 6 workstations
- NodeJS must be installed

Hardware monitoring module constraint

- NodeJS must be installed

2.5 Assumptions and dependencies

Yet there was no assumption made due to the reason both modules are platform independent and require no additional assist from third party applications or any additional hardware other than the assist get from the OS.

3 Specific requirements [(1)] (for “Object Oriented” products)

3.1 External interface requirements

3.1.1 User interfaces

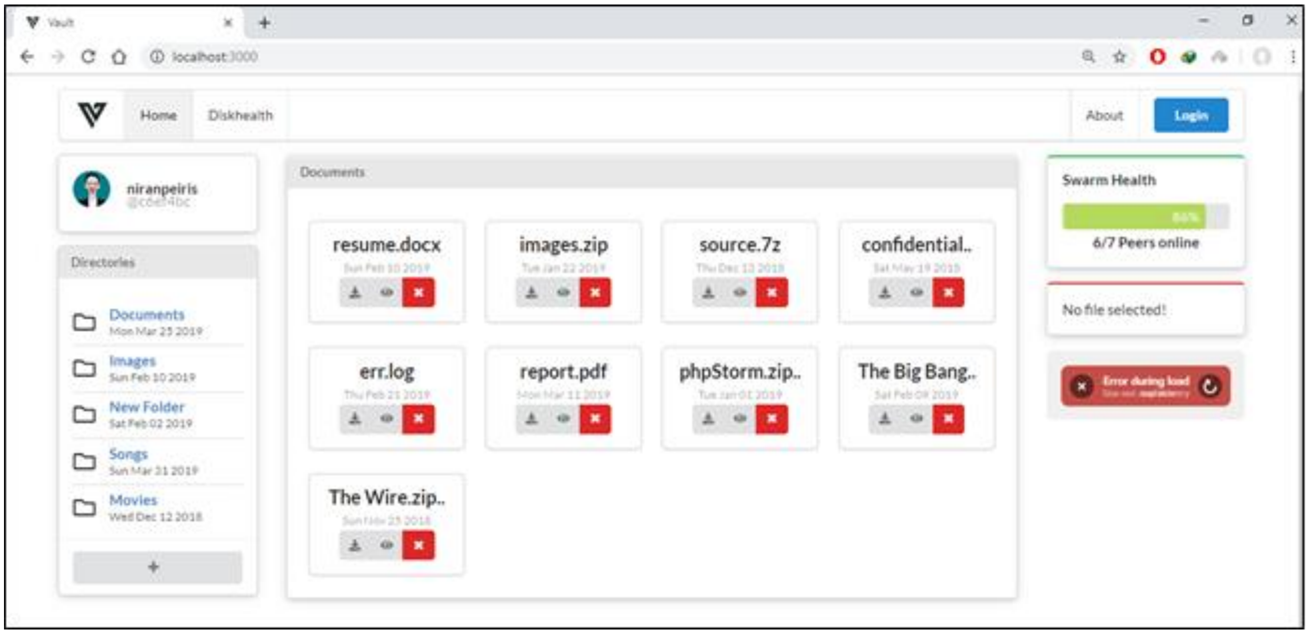
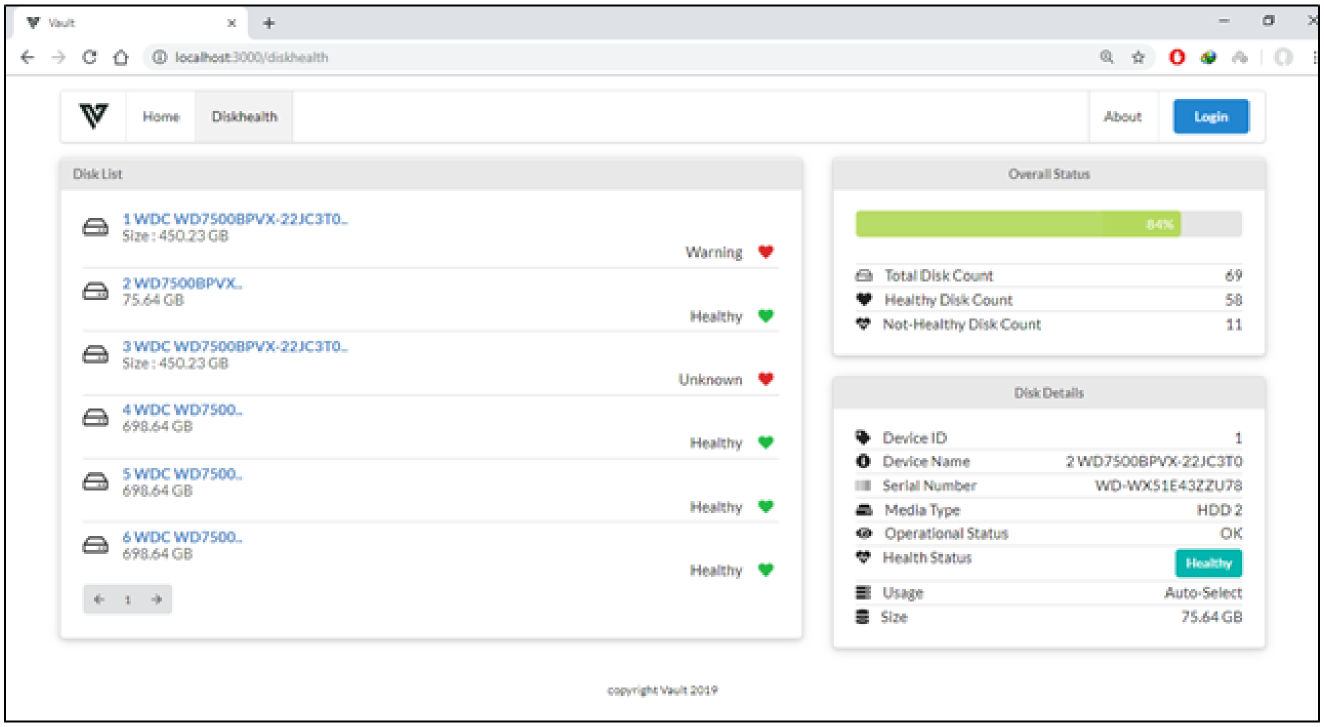


Figure 6: File Uploading/Downloading interface

Table 13:File uploading/downloading interface description

Name of item	File uploading/downloading interface (Redundancy handling)
Description of purpose	Input files needed to the encryption and redundancy modules which will be segmented by the redundancy module will be input to the system via this interface.
Source of input or destination of output	Any type of files
Valid range, accuracy and/or tolerance	100%
Units of measure	Bytes
Timing	-
Relationships to other inputs/outputs	Original file information such as name, size, extension, locations of the segments will be outputted to the blockchain API.
Screen formats/organization	Screen is organized in a monitor view
Window formats/organization	-
Data formats	Bytes



Disk List

1 WDC WD7500BPVX-22JC3T0..
Size : 450.23 GB
Warning

2 WD7500BPVX..
75.64 GB
Healthy

3 WDC WD7500BPVX-22JC3T0..
Size : 450.23 GB
Unknown

4 WDC WD7500..
698.64 GB
Healthy

5 WDC WD7500..
698.64 GB
Healthy

6 WDC WD7500..
698.64 GB
Healthy

Overall Status

84%

Total Disk Count	69
Healthy Disk Count	58
Not-Healthy Disk Count	11

Disk Details

Device ID	1
Device Name	2 WD7500BPVX-22JC3T0
Serial Number	WD-WX51E43ZZU78
Media Type	HDD 2
Operational Status	OK
Health Status	Healthy
Usage	Auto-Select
Size	75.64 GB

copyright Vault 2019

Figure 7:Hardware monitoring interface

Table 14:Hardware monitoring interface description

Name of item	Hardware monitoring interface
Description of purpose	Health and the availability of the each and every hard disk will be presented to the user by this interface.
Source of input or destination of output	Health related information from Operating system
Valid range, accuracy and/or tolerance	80%
Units of measure	-
Timing	-
Relationships to other inputs/outputs	Availability information will be outputted to the distribution API when segments of a file is recollecting.
Screen formats/organization	Screen is organized in a monitor view
Window formats/organization	-
Data formats	Alphanumeric

3.1.2 Hardware interfaces

Since the software does not directly interact with Hardware level components and doesn't rely on how storage devices handle data, there is no requirement to document hardware interfaces.

3.1.3 Software interfaces

Other than NodeJS and OS the research component mentioned in this document won't be interacting with any other software application.

3.1.4 Communication interfaces

Both redundancy handling and hardware monitoring APIs act as middle layer processes, hence they won't be needing any external communication interfaces other than the interfaces needed to communicate with the other APIs within the application and the OS.

Table 15: Redundancy handling API communication interfaces

Interface	External Module	Connection details
Interface 1	Encryption Module	When user uploads a file, it will be encrypted by the encryption module and then encrypted file will be send to the redundancy API to be segmented. When user downloads a file, segments will be used to regenerate the original file and then it will be sent to encryption module to be decrypted.
Interface 2	Distribution Module	When a file is segmented, segments will be sent to the distribution function to scatter them across the network. When redundancy API needs segments to regenerate an original file a request will be made to the distribution function and it will collect segments from the network.

Table 16:Hardware monitoring API communication interfaces

Interface	External Module/ Feature	Connection details
Interface 1	Distribution Module	Gather disk health and availability information from all over the network through distribution function.
Interface 2	OS	Gather disk health related information from OS.

3.2 Classes/Objects

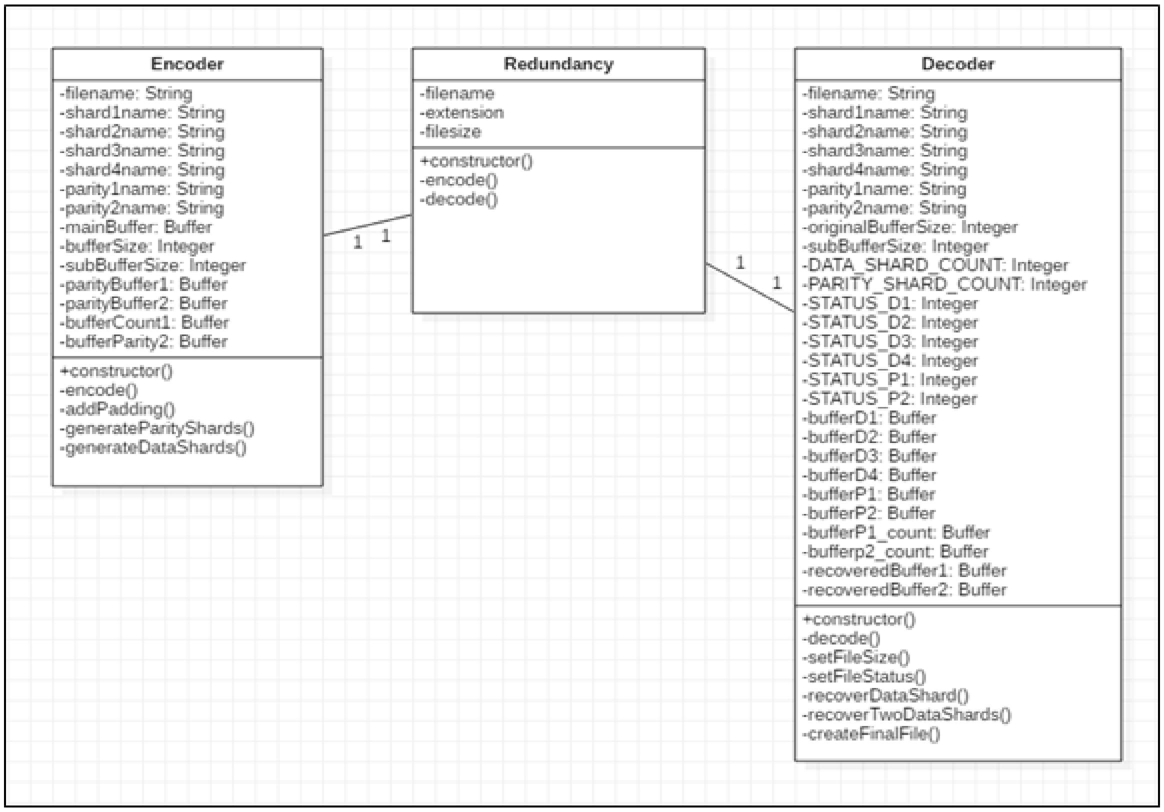


Figure 8: Redundancy handling API class diagram

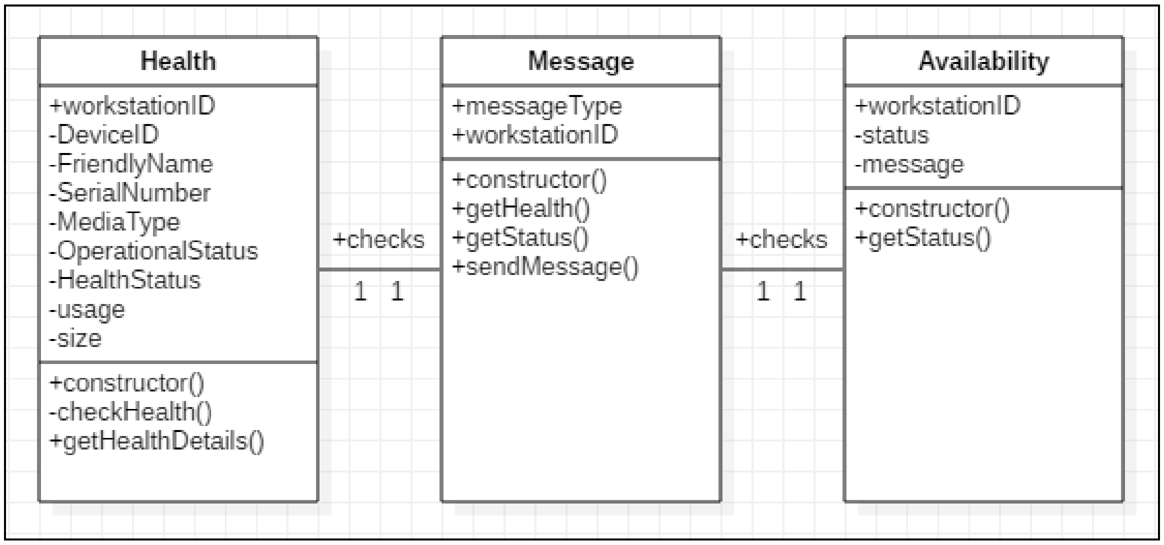


Figure 9: Hardware monitoring API class diagram

3.3 Performance requirements

Storage and network requirements can be identified for the blockchain and Messenger API facilitation. The host machine should have the necessary capacity to host storage for the Rethink DB server. The network should be reliable. Network speed/bandwidth is not a main concern since the amount of data throughput required by the module is very low.

3.4 Design constraints

There are no specific design constraints involves with the component which is mentioned above in this document, as long as the interfaces are easy to understand and use.

3.5 Software system attributes

In this section the features which will be offered to the customers will be described.

3.5.1 Reliability

Ability of a system to maintain its ordinary operation within given time in a given environment with minimum failures is called reliability. Purpose of the redundancy handling API and hardware monitoring API is to increase the redundancy.

As mentioned earlier redundancy function will segment the uploading file to a N number of data shards and generate M number of parity shards, where M = N/2. When a user downloads a certain file even though M number of workstations are down at the time, the original file can be regenerated by doing the segmentation and regeneration processes in a redundant manner which allows the system to increase the reliability of the system.

Hardware monitoring function increase the reliability by allowing administrative users to continuously monitor the system. By monitoring the disk health administrators can move the data inside one disk to another disk if that certain disk's health is not in a good condition, which allows them to stop future failure.

3.5.2 Availability

Probability a system is functioning when its services are required by the users of the system is called availability of a system. Another purpose of the redundancy handling API and the hardware monitoring API is to maintain the availability of the system.

Redundancy handling function increase the availability by allowing users to download a file even though some of the workstations which holds some data shards are offline. Assume one file will be divided in to, N data shards and M parity shards, where M= N/2. Altogether there are 3N/2 files will be scattered among the network. Users doesn't have to wait until all the 3N/2 workstations are online by using the redundancy handling API the same original file can be regenerated using N number of available workstations.

Hardware monitoring function allows administrative users to continuously monitor the availability of the system, which allows them to increase the availability of the system by using policy implementation or else in some other relevant manner.

3.5.3 Security

Security of a system is the function which allows system to provide its services to its legitimate users, while resisting the other unauthored users from gaining access to the system or its data and resisting authorized users from performing unauthorized actions. Redundancy handling API and the hardware monitoring API assist in the security maintaining process in a different manner.

Redundancy handling API scatter the data and send them through the network in a secure manner, hence no one person can have files needed to regenerate the same file. Hardware monitoring API allows the system to maintain the security by giving administrative users the ability to monitor the system availability, which helps administrators to identify availability patterns to secure data and the system.

3.5.4 Maintainability

Maintainability is the ability to change the systems functionalities and increase the performance by applying system repairs and updates while maintaining systems availability, security and reliability. Apart from the details mentioned in this document major changes to the system cannot be expected in the hardware monitoring and redundancy handling APIs, but there can be few changes in the class structure and the interfaces which was depicts in this document and there also can be few improvements in the APIs as well.

4 References

[1] BlueWhale, "bwstor.com," BlueWhale, 03 2019. [Online]. Available: www.bwstor.com.cn/templates/T_product_EN/index.aspx?nodeid=150&page=ContentPage&contentid=402. [Accessed 02 03 2019].

[2] Ceph, "ceph.com," Ceph, 03 2019. [Online]. Available: https://ceph.com/ceph-storage/file-system/. [Accessed 01 03 2019].

[3] Minio, "minio," Minio, 02 2019. [Online]. Available: https://www.minio.io/. [Accessed 03 2019].

[4] Oracle, "oracle.com," Oracle, 03 2019. [Online]. Available: https://oss.oracle.com/projects/ocfs2/. [Accessed 03 2019].

[5] The OrangeFS Project, "orangefs," The OrangeFS Project, 03 2019. [Online]. Available: http://www.orangefs.org/. [Accessed 03 2019].

[6] IBM, "www.ibm.com," IBM, 03 2019. [Online]. Available: https://www.ibm.com/us-en/marketplace/scale-out-file-and-object-storage. [Accessed 03 2019].

[7] "moosefs.com," moosefs, [Online]. Available: https://moosefs.com/.