**High Availability (HA) Distributed File Storage**

# Document: Software Requirements Specification

**Version 1.2**

**Publication date: 11-November-2017**

**Group name: Gryffindor**

**Group Members:**

* JITENDRA NEELAM
* BHAMIDI SIRISHA MANASWINI
* CHINNA BALAJI YALLA
* NITISH NAGABHAIRAVA
* RAGHUVINAYAK RAO MEDISETTI

Contents:

1. Preface

Release v1.2 on 2017-11-10

Release v1.1 on 2016-04-25 Release v1.0 on 2016-04-18

1. Glossary and abbreviations
2. System Architecture
   1. Frontend
   2. Database
   3. Backend
3. Requirements

Requirement: Identification string

Creation date: Date when this requirement was created Change date: Date when this requirement was changed Module: Architecture module it belongs to

Type: Functional/Non-Functional

Dependencies: What other requirements does this depend on Test: Test case identification string

Assignee: Team member responsible for the requirement being fulfilled Description: Detailed information, including graphs, tables or UML diagrams Comment: Reason why this requirement was changed.

* 1. User requirements

1. References

# Preface

The main concept of this project is to develop a secure file storage to the company SecuriFile in the form of a distributed file storage system with high availability to the customers.

In this we are creating replicas for the servers and when a user uploads a file, the file is stored in a randomly chosen server. we use file transfer protocol for the transfer of data. **Service Developer:** Gryffindor

**Customer:** Dragos llie

In this document, we defined the technical terms and a short note on them, system architecture, user and system requirements and references.

* **Release v1.2 on 2017-11-10**

Folder option added

* **Release v1.1 on 2016-04-25**

Preface changed

* **Release v1.0 on 2016-04-18**

Initial Release

# Glossary and abbreviations

* **HTTP: Hyper Text Transfer Protocol**

It is a transfer of version data formats between server and client EX: plain txt, hyper txt, video and sound

* **FTPS: File Transfer Protocol Security**

It is an extension for commonly used file transfer protocol (FTP) that adds support for the transfer layer security (TLS) and secure sockets layer (SSL).

* **Message digest: SHA-1**

IT is a crypto graphic hash function which is consider practically impossible to invert that is to recreate the input data from its hash value alone.

SHA-1: secure Hash algorithm.SHA-1 produces a 160bit (20 byte) hash value known as a message digest. SHA-1 advancements are SHA-2 and SHA-3.

* **GUI: Graphical User Interface**

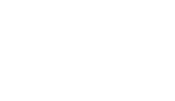
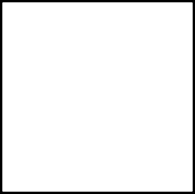
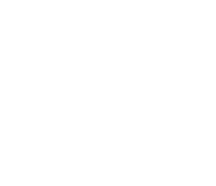
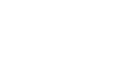
It is a type of interface which helps in interaction with electronic devices through graphical icon and visual indicators.

* **SQL Server: Structured Query Language Server**

SQL is used to store, query and manipulate data. It is used for managing data in a relational data base.

# System architecture

In this section of the proposal we provide the system architecture, which determines the working of the system. Initially the system can be determined in three sections they are the front end, database and the back end of the system



Module 1: Front End

Module 2: Data Base

Module 3: Back End

USER

Replicas

MySQL

ssh

connection

Web GUI

Origin Server

Graphs on uptime of each service.

Lib MySQL

Lib MySQL

Backend Tool Continuously monitors

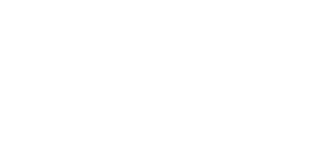
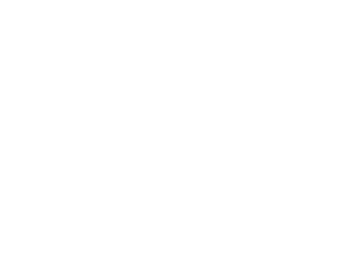
ssh Connection



**Fig 1: System Architecture**

### Frontend

At first the user is asked to login, he gets his access from database and this is done through a series of process through HTML and CSS. We create a login page to login into the server but first he needs to register into the admin server, the admin server provides a verification mail to the user and later conformation of the mail is done and the connections for the web pages are done through PHP. Now a separate account is allocated to the customer through which user can store files. Through the login page the customer logins into the user’s account and user can upload the file and make modifications to the existing file. The storage capacity is limited to the customer and users can’t exceed the given storage capacity, later he can logout from the page.



User

Dash Board

Admin Server

1.Registration

2.Storage of user files

3.Authentication through login page

4.Provides the service

5.Request for the file

6.Permits user to open the files and access it

MySQL

Data Base

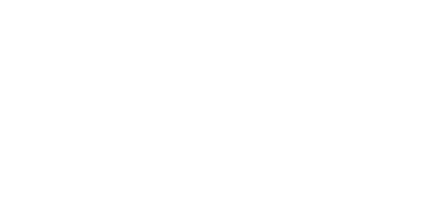
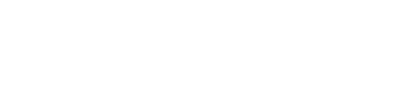
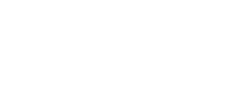


Fig 2: Front End

### Database

The MySQL database contains user’s information, data status and uptime information inserted into their respective service tables

In the MySQL database, we create a data slot for each customer. Through front end the user can access into user’s login account and through backend the user can access user’s files.



Request for restart service

Web GUI

Back End

Stores user files and retrieves files status and obtain information from the tables

MySQL

Data Base

Insert Status and uptime in the tables

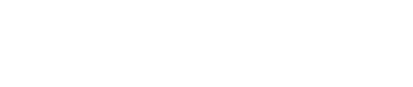
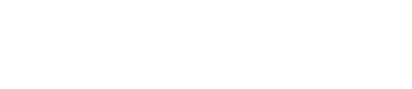
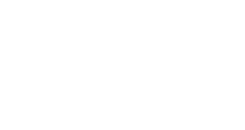
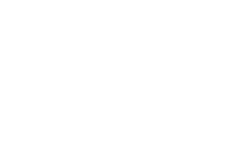


Fig 3: Data Bases

### Backend

The third section of the architecture is used to connect the entire architecture. For backend programming we use Python, which is used to connect the servers. Backend is also used to ping the servers constantly, if the ping is lost or server is down it sends a message to database and in the database, it is stored if the connection is lost.

It retrieves the status and stores it in the database in MySQL. Separate tables are allocated for each server.



Replicas

Back End

Inserts status and uptime into respective tables

MySQL

Establishes ssh connection

Monitors Replicas

Stores number of restarts

Retrieves status and uptime data

Restarts service upon demand



Fig 4: Back End

**4.Requirements:**

Requirements for a software development project is divided into two parts. They are:

**4.1 User Requirements:**

**The user requirement(s) specification** is a document usually that specifies what the user expects the software to be able to do.

Following are the requirements provided for the user.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Serial number** | **Requirement** | **Identification String** | **Change date** | **Module** | **Type** | **Dependencies** | **Test** | **Description** |
| 1 | Giving sign-in, sign-up, login options to the user. | **UR1** | **10-06-2017** | Front-end | Functional |  |  | It is used ti describe the GUI for the users. |
| 2 | Error detected if email format is invalid. | **UR2** | **10-06-2017** | Front End | Functional |  |  | Error will be detected if email format is invalid without "@" and ".". |
| 3 | User Authentication fails incorrect details. | **UR3** | **10-06-2017** | Front End | Functional |  |  | Click on sign-in button and enter user credentials. Error occurs if the user enters the incorrect data. |
| 4 | The user details are stored in database. | **UR4** | **10-06-2017** | Front end | Functional |  |  | The user details are stored in corresponding columns in the secure database. |
| 5 | The user password stored in database is encrypted. | **UR5** | **10-06-2017** | Front End | Functional |  |  | The user credentials are stored in database GUI. The password is encrypted in database. |
| 6 | Existing user cannot sign up again. | **UR6** | **10-06-2017** | Front End | Functio nal |  |  | An error message is displayed with email already exists. |
| 7 | Existing user logs in to the dashboard with verified credentials. | **UR7** | **10-06-17** | Front-end | Functional |  |  | User logs in to the dashboard displays with valid credentials. |
| 8 | The dashboard displays the files owned by the user. | **UR8** | **16-10-2017** | Back End | Functio nal |  |  | The files uploaded by the user are displayed on the dashboard. |
| 9 | To store the list of files in the database. | **UR9** | **16-10-2017** | Front End | Function nil |  |  | Users file is displayed in the database table after uploading. |
| 10 | View folder option in Database | **UR10** | **16-10-2017** | Back End | Functional |  |  | The folder upload option is visible. |
| 11 | The database stores file size. | **UR11** | **16-10-2017** | Back End | Function nil |  |  | The file size, created is present in the corresponding table. |
| 12 | User should be able to upload files to the distributed file storage system. | **UR12** | **16-10-2017** | Backend | Functional |  |  | User can upload the required file into the system. |
| 13 | User should be able to download files from the distributed file storage system. | **UR13** | **16-10-2017** | Back end | Functional |  |  | User can download the required file. |
| 14 | User can create New Folder in Database | **UR14** | **19-10-2017** | Back End | Functional |  |  | User have access to create New Folder in Database |
| 15 | User can edit Folder in Database | **UR15** | **19-10-2017** | Front-end | Functional |  |  | User can modify folder. |
| 16 | User can delete Folder/File in Database | **UR16** | **19-10-2017** | Front-end | Functional |  |  | User can delete file. |
| 17 | User can upload new files into the folder in Database. | **UR17** | **19-10-2017** | Back-end | Functional |  |  | User can upload new files to the folder in Database |
| 18 | User can view file in the folder in Database. | **UR18** | **19-10-2017** | Front-end | Functional |  |  | User can see file in the folder in Database |
| 19 | User can copy files in the Folder in Database. | **UR19** | **19-10-2017** | Front-end | Functional |  |  | User can copy files into the Folder in Database |
| 20 | User can move files in the Folder in Database. | **UR20** | **19-10-2017** | Back-end | Functional |  |  | User can move files into Folder in Database. |
| 21 | User can delete files in the Folder in Database | **UR21** | **25-10-2017** | Front-end | Functional |  |  | User can delete files contained inside Folder in Database. |
| 22 | User can see the status of folder Active or De-Active. | **UR22** | **25-10-2017** | Back-end | Functional |  |  | User can view the status of folder i.e., Active or De-Active. |
| 23 | User can share Folder/File between users by clicking Can View button. | **UR23** | **25-10-2017** | Back End | Functional |  |  | User can share Folder/File in between users by just clicking on Can View button. |
| 24 | User can edit Folder/File between users by clicking Can Edit button. | **UR24** | **25-10-2017** | Back End | Functional |  |  | User can edit Folder/File in between users by just clicking Can Edit button. |
| 25 | To see the list of files shared to other users under sent files. | **UR25** | **25-10-2017** | Back End | Functional |  |  | User can see list of files sent to other users under sent files menu. |
| 26 | To see the list of files received by the other users. | **UR26** | **25-10-2017** | Back End | Functional |  |  | User can see list of files received by the other users under receive files. |
| 27. | Select servers randomly | **UR27** | **25-10-2017** | Back End | Functional |  |  | The selection of server done randomly. |
| 28 | If three ping messages continuously fail email sent to administrator. | **UR28** | **28-10-2017** | Back End | Functional |  |  | The email is sent to the administrator when the message fail continuously. |
| 29 | If one server fail it is redirected to another server. | **UR29** | **28-10-2017** | Back End | Functional |  |  | The user is redirected to another server if one server is fails. |

**5.References:**

[1] Software Documentation, <https://en.wikipedia.org/wiki/Softwaredocumentation>

[2]. Ian Somerville UU4 Software Engineering. 9th ed.