# Team Project #2

Software Requirements Specification

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Description** | **Author** |
| 2/17/2022 | 1.0 | Initial Version | Quang Nguyen |
| 2/19/2022 | 1.1 | Revision Draft | Quang, Travis, Andrew, Michael |
| 2/21/2022 | 1.2 | Final Draft | Quang, Travis, Andrew, Michael |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

Team Project #2 1

1. Purpose iii
   1. Scope 4
   2. Definitions, Acronyms, Abbreviations 4
   3. References 4
   4. Overview 4
2. Overall Description 4
   1. Product Perspective 4
   2. Product Architecture 4
   3. Product Functionality/Features 5
   4. Constraints 5
   5. Assumptions and Dependencies 5
3. Specific Requirements 5
   1. Functional Requirements 5
      1. Common Requirements: 5
      2. Server Module Requirements: 5
      3. Client Module Requirements: 5
      4. Persistence Module Requirements: 6
   2. External Interface Requirements 6
   3. Internal Interface Requirements 6
4. Non-Functional Requirements 6
   1. Security and Privacy Requirements 6
   2. Environmental Requirements 6
   3. Performance Requirements 7

4.3.1 A client request must finish within 5 seconds. 7

# Purpose

This document outlines the requirements for the local distributed file storage system.

## Scope

This document will catalog the user, system, and hardware requirements for the file storage system. It will not, however, document how these requirements will be implemented.

## Definitions, Acronyms, Abbreviations

HTTPS: Hypertext Transfer Protocol, is a protocol for transmitting hypermedia such as

HTML.

HTML: HyperText Markup Language, is a language that made up the web.

GUI: Graphical User Interface, is a form of user interface that allows users to interact with the program.

JWT: JSON Web Token, is a compact URL-safe means to representing transfer between two parties.

## References

Use Case Specification Document

UML Use Case Diagrams Document

Class Diagrams

Sequence Diagrams

## Overview

The distributed file transfer system is designed to let user upload and request file as easily as possible. With a failsafe system and security feature that will make sure that the file is secured and people who are accessing the file have the right credential.

# Overall Description

## Product Perspective

The distributed file system is a cloud adjacent method of peer-to-peer file storage. It is a system used to self-contain company wide data; retaining data in house ensures privacy and security needs are met.

## Product Architecture

The system will be organized into 3 major modules: the server module, the client module, and the persistence module.

## Product Functionality/Features

Users log in to distributed file system via username/password combination. Users view text list of files stored within system and can access any available file and/or upload new files. Interactions between user, system and files are logged.

## Constraints

Location of files will be unknown to the user. Files uploaded by a given user will be stored in an unknown, separate, arbitrary location(s) within the company.

## Assumptions and Dependencies

It is assumed that company hardware is standardized; all users will run identical operating systems on corresponding hardware.

# Specific Requirements

## Functional Requirements

### Common Requirements:

3.1.1.1 Users should be allowed to log in using their issued id and pin, both of which are alphanumeric strings between 6 and 20 characters in length.

3.1.1.2 Users able to fetch file.

3.1.1.3 Able to process upload and distribute file to multiple nodes.

3.1.1.4 Server able to log every action and saved log for at least 30 days.

### Server Module Requirements:

3.1.2.1 Server should be able to keep a log of which users pushed a request.

3.1.2.2 Server should be able to send requested files to user

3.1.2.3 Files sent should not be able to be edited

3.1.2.4 Able to differentiate between user levels based on authentication provided

### Client Module Requirements:

3.1.3.1 Client able to push requests for files

3.1.3.2 Client able to look at list of available files and choose desired file from list

3.1.2.3 Client can upload their files through the server on their personal node

3.1.2.4 Client is not able to modify files that are not their own

3.1.2.5 Able to log in using id and pin to gain access to server

### Persistence Module Requirements:

3.1.4.1 Persist log to a file system.

3.1.4.2 Able to pull log.

## External Interface Requirements

3.2.1 There must be an easy to follow and easy to understand GUI for users.

3.2.2 GUI will be Java program.

## Internal Interface Requirements

3.3.1 Every action that server process will be log into Persistence module, with date and time stamp.

3.3.2 Any file type and any size will be accepted.

3.3.3 Any file that is uploaded will need to be saved to 2 or more nodes.

3.3.4 The log will be saved as a .txt file up to 30 days.

# Non-Functional Requirements

## Security and Privacy Requirements

4.1.1 Auto generated username.

4.1.2 Id and passwords change every year.

4.1.3. Local Network Only.

4.1.4 All files that are saved by the Server must be encrypted.

4.1.5 Authentication and Authorization must only be handled by the server

4.1.6 All request must pass a authorization token issued from the server.

## Environmental Requirements

4.2.1 Program with be written and operate in Java.

4.2.2. Systems in Java must use at least JDK 11

4.2.3 Server must use environment variables to configure IP and port.

## 

## Performance Requirements

## 4.3.1 A client request must finish within 5 seconds.