File Storage System

Design Document

Megan Paffrath, Haiyang Sun, Priyanka Gopal,
Prasoon Shakya, Monicah Gikanga

Revision History

Date	Revision	Description	Author
07/7/2020	1.0	Initial Version	Megan Paffrath, Haiyang Sun, Priyanka Gopal, Prasoon Shakya, Monicah Gikanga
07/15/2020	1.1	Final Version	Megan Paffrath, Haiyang Sun, Priyanka Gopal, Prasoon Shakya, Monicah Gikanga

1. Introduction

1.1. Goals and Objectives

This document describes the server and client side of the system and how the programs will interact to provide full functionality to the user running the client software.

1.2. Statement of Scope

This document covers the ability for the system to be maintained, usable, efficient, and user friendly.

1.3. Software Context

The **ClientCommunicator** and **ServerCommunicator** will contain the functionality to send serialized messages back and forth to allow communication.

1.4. Major Constraints

Issue 1: How will the files be stored and organized?

Option 1.1: Files get saved to the server's system and are given unique object IDs to keep track of them.

Option 1.2: Files get stored on a server and are given unique object IDs to keep track of them and help find them.

Decision: For the beginning of this project, files will need to be stored on the server's system to prevent extra complexity for the time being.

2. Data Design

2.1. Client Side

- Serialized messages will be able to be sent to the server as well as received though the **ClientCommunicator**.
- Responses from the server (serialized messages) will be accepted by the **ClientCommunicator** and processed by **ClientHelper**
- **ClientHelper** will also help formulate the serialized messages that are to be sent to the server.

2.2. Server Side

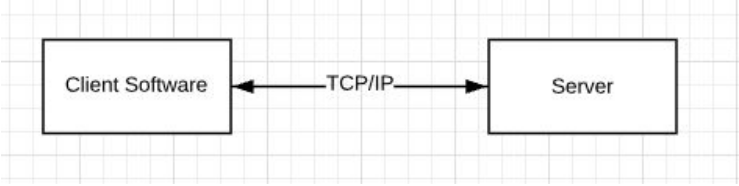
- The **ServerCommunicator** will accept and process serialized messages from the client.
- The **ServerHelper** methods take received messages as arguments to formulate responses messages for the client

3. Architectural and Component-level Design

3.1. Program Structure

This system runs as a client-server application in which the client sends messages to the server and the server sends responses to support functionality.

3.1.1. Architectural Diagram



3.2. Description of Client

3.2.1. Client processing narrative

The client will take various commands from the user that will specify what is to be done. These commands will call the functions within ClientHelper that will help create serialized messages to be sent to the server. These function help the user login, logout, sign up, view files, open folders, read

files, upload files, delete items, download, share, and change account settings.

3.2.2. Client interface description

The client side utilizes the **ClientCommunicator** to both send and receive serialized messages to and from the server.

3.2.3. Client processing detains

The client side **ClientCommunicator** will need to keep a list of the serialized messages that are both ingoing and outgoing to preserve the order in which things are sent.

3.3. Description of Server - Software Interface Description

3.3.1. External Interfaces

Later down the line, a server should be used to store files and file data.

3.3.2. Internal Interfaces

For now, the server program will store files and file data to the local server's system.

3.3.3. Human Interfaces

Each person will have their own main "folder" that will store all of their files, folders, and the files/folders that have been shared with them.

4. User Interface Design

The user will be able to click buttons to specify how they wish to interact with the program. These button clicks will determine the events that happen next. All events that will occur have been specified in the SRS document and diagramed with the supporting UML.

5. Restrictions, Limitations, and constraints

- Time constraints will need to be closely analysed to ensure best organization and productivity
- The restriction of not using a server will mean that the server program will need to save files and file data to it's local system
- Data is account based for the client, users need server permission to access, share, edit and delete.
- Multiple Clients are not allows to access the system under same IP address
- Same client can not login at the same time in more than one IP location.
- Client and server and not allowed to access data from each other unless permission is granted
- If users do not interact with the client program for long periods of time or become offline suddenly while accessing the account, they will be logged out

6. Testing Issues

- Testing the ability to send and receive messages would have to span over the network, this may mean large amounts of black box testing.
- All internal functionality for both server and client should be tested with JUnit with at least 80% coverage.
- Blocking clients from accessing the system in multiple locations at the same time. May be difficult to test.

7. Appendices

- Software Requirements Specification Document
- UML Class Diagram
- UML Use Case Specification Document