



**VIT<sup>®</sup>**

**Vellore Institute of Technology**

(Deemed to be University under section 3 of UGC Act, 1956)

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**B. Tech., Fall Semester 2021-2022**

**File Sharing System**

**Course Faculty: Dr. Braveen M.**

**Batch Members**

1. Pranav Barathwaj P(20BCE1351)
2. Panav Sinha(20BCE1640)

## **Table of Contents**

<b>S. NO.</b>	<b>TITLE</b>	<b>PAGE NUMBER</b>
1	Abstract	3
2	Introduction	4
3	Requirement Gathering	5
4	Structure of the Project	7
5	Planning and Scheduling	9
6	UML Diagrams	10
7	Architecture	13
8	Graphical User Interface	15
9	Source Code (Sample)	19
10	Verification and Validation - Test Case Document	21
11	Conclusion	22
12	References	23

# **Abstract**

This project is concerned with file sharing systems. Very often, people find it difficult to transfer files across different systems that run on different architecture. But with the presence of technology, computers, networks, and the internet, the process of communication became very easy. The required information, files and programs can be found, downloaded easily and effortlessly.

Based on these facts, this project aims at creating a file sharing system for the people who want to share files within a speculated private network. It can be a school, a university or even a house. This system allows users within the network to download and upload files; they can search for specific files uploaded by the other members too.

This application is going to be managed by the administrator or a power user who is going to coordinate and classify the different files and clients using the system.

# Introduction

The purpose of this project is to create a simple file sharing system that shares any format of files between any number of systems. A system can be a laptop, mobile phone etc. Basic feature involved in this system is the secure connection between a server system and various client systems, through which transfer of files can take place. There are three subsystems present and they include:

- The Client System
- The Server System
- The Data Transfer

The main feature includes file sharing across a central server present in a network, consisting of appropriate security protocols, to protect it from hackers. The system is developed for flexible use, that is the ability to operate and transfer files across any OS. The FSS uses a GUI.

The scope of the Server System is to call on different devices within the network and establish itself as a unique server. This enables other client systems to connect to the server system.

The scope of the Client System is to call on different devices and search for an appropriate server. By following appropriate security protocols, the client can establish a connection and send files across the server to any other clients present within the same client-server architecture.

The scope of the Data Transfer is the transfer of files across different client systems present within the network. The transfer of files follows HTTPS protocol, which is encrypted and secure. The concept of a shared clipboard also persists.

# Requirements Gathering

## Functional Requirements

### **-Server Module:**

- The user should be able to create a server with a name and password.
- This server system should establish its uniqueness within the network by sending get requests across the network. If the server finds itself not unique, an alert is sent to the genuine user.
- The server system should be able to receive and connect with the client systems to transfer data. This can be done using HTTPS (Hyper Transfer Text Protocol Secure).
- The user should be able to shut them down the server system, thus shutting down the client-server network.

### **-Client Module:**

- The user should be able to create a client system.
- The user should be able to enter an appropriate server name and password in order to send files across that particular server.
- The user should be able to copy data files onto the system, which can be transferred to other client systems.
- This module should be able to send and receive data from the server. This can be done using HTTPS (Hyper Text Transfer Protocol Secure).

### **-Data Transfer Module:**

- This module should be able to follow the protocols mentioned and transfer data from client to client using a server.
- The user should be able to access shared clipboards in order to ease transfer of files.

## **Non-Functional Requirements**

### **Performance Requirements**

The software should have high performance and low failure rates. The hardware and software should be able to transmit/receive data from different systems with high efficiency. Machines should have all recent Windows/Linux updates installed. Machines must have firewalls installed and active virus scanning software in usage. Data receiving/transmitting should be done using high security transmission.

### **Safety Requirements**

The software must be used by the User using application configured based on the OS and architecture. This will provide high level of security and the User will get the best experience while using them. Also, the User must not share their private credentials which can be used to log into their networks. The User is strongly advised to go through the User Manual first.

### **Maintenance Requirements**

The software must be made in such a manner that at a later point of time, software system or component can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment. This means that all new features must be easily incorporated to the pre-existing software system without hassle and providing discomfort to the User, through updates. The system must be able to handle a growing amount of work by adding resources to the system

### **Security Requirements**

The software should have secure connection that prevents any form of malicious attacks on data transferred. This is achieved by using an appropriate protocol. The presence of a power user, where he/she can see the logs of the system enhances the security of the system. The identification of a unique server is also an important feature that comes along.

### **Reliability**

The software must have 24 X 7 availability to the user, provided the user has a stable network connection. The software uses TCP to encapsulate the data, which further uses HTTPS in application layer to provide secure connection. Therefore, there is no loss of data, thus making it reliable.

# Structure of the Project

## **Server Module**

This module is the most important because of its core action of being able to facilitate the sharing of data, software and hardware resources. The server system's main role is to establish a secure connection to prevent hackers from exploiting the system. This is done by establishing its uniqueness by sending get requests across the network, thus identifying other devices, and its uniqueness. If it is not unique, the module identifies the presence of another server with similar name or a malicious user, thus sending an alert to the genuine user. This feature enhances the security of the system. After having established connection with different clients, the server acts a median for data communication, using HTTPS.

## **Client Module**

The client system module acts as the two ends to the system. This module also focusses on security. This is achieved by making sure the client system is connected to a genuine server. This can be established by sending get requests to different devices and thus finding the suitable server system. This module is acts as a source for data transfer. In the presence of a server system, this module can send data files to another system having the same module. This is achieved only when modules of systems are connected to the server and takes place using HTTPS.

## **Data Transfer Module**

This module comprises of the data transfer across the two client systems governed by the server system. This secure transfer of data is achieved by using HTTPS (Hyper Text Transfer Protocol). The data upon encapsulating uses the TCP/IP model in Transport Layer in order to initiate the communication between the hardware and software interface. The module also comprises of shared clipboards.

HTTPS uses an encryption protocol to encrypt communications. The protocol is called Transport Layer Security (TLS), although formerly it was known as Secure Sockets Layer (SSL). This protocol secures communications by using what's known as an asymmetric public key infrastructure.

This type of security system uses two different keys to encrypt communications between two parties:

1. The private key - this key is controlled by the owner of a website and it's kept, as the reader may have speculated, private. This key lives on a web server and is used to decrypt information encrypted by the public key.
2. The public key - this key is available to everyone who wants to interact with the server in a way that's secure. Information that's encrypted by the public key can only be decrypted by the private key.

The Structure of the system requires three users: Server User, Client User and Power User.

Server User:

- This user operates over the server domain, having the ability to initiate the presence of a server in a network.
- This server can send get requests to other machines with the network to create a client-server architecture.
- The user can set the server's name and password.
- The user can be involved in data transfer apart from the clients.
- The user can choose to shut the server down.

Client user:

- This user operates the client system, having the ability to act as a client in a given client-server network.
- The client can enter the server's name and associated and code word in order connect with the server.
- The client can send data files to be transferred across the system.

Power user:

- This user can be either a server user or a client user.
- If the user is a server, he/she can get access to all the logs, thus viewing real-time and past actions occurring in various systems present within the client-system architecture.
- If the user is a client, he/she can get access to all the logs, thus viewing real-time and past actions only within its system.

All access and data transfers/receiving is logged, in order to maintain a level of transparency, in order to prevent abuse of the system, and in order to hunt down any unauthorized users or hackers.



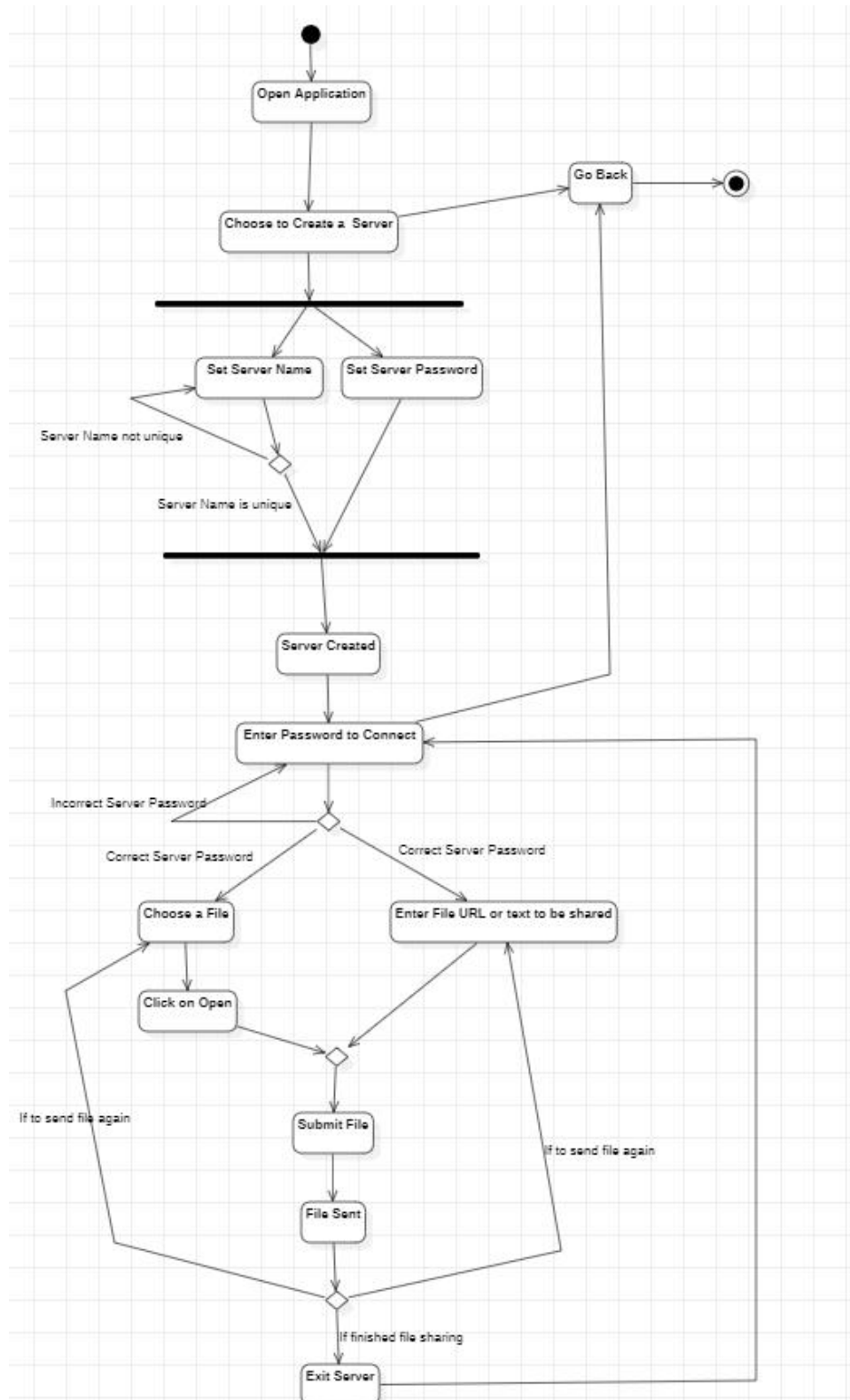
# Planning and Scheduling

*\*The WBS Document has been zipped along with this report.*

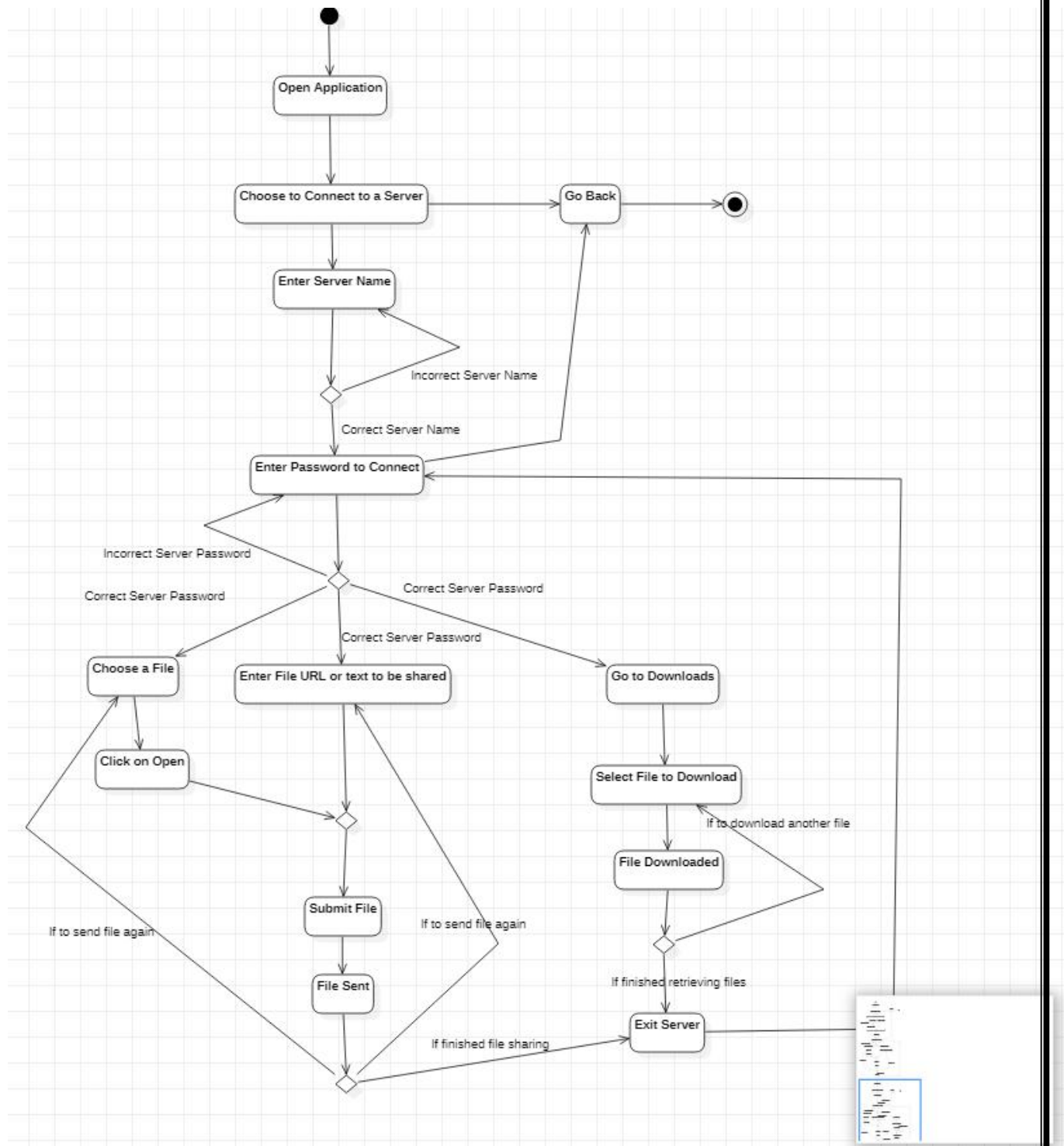
# UML Diagrams

## Activity Diagram:

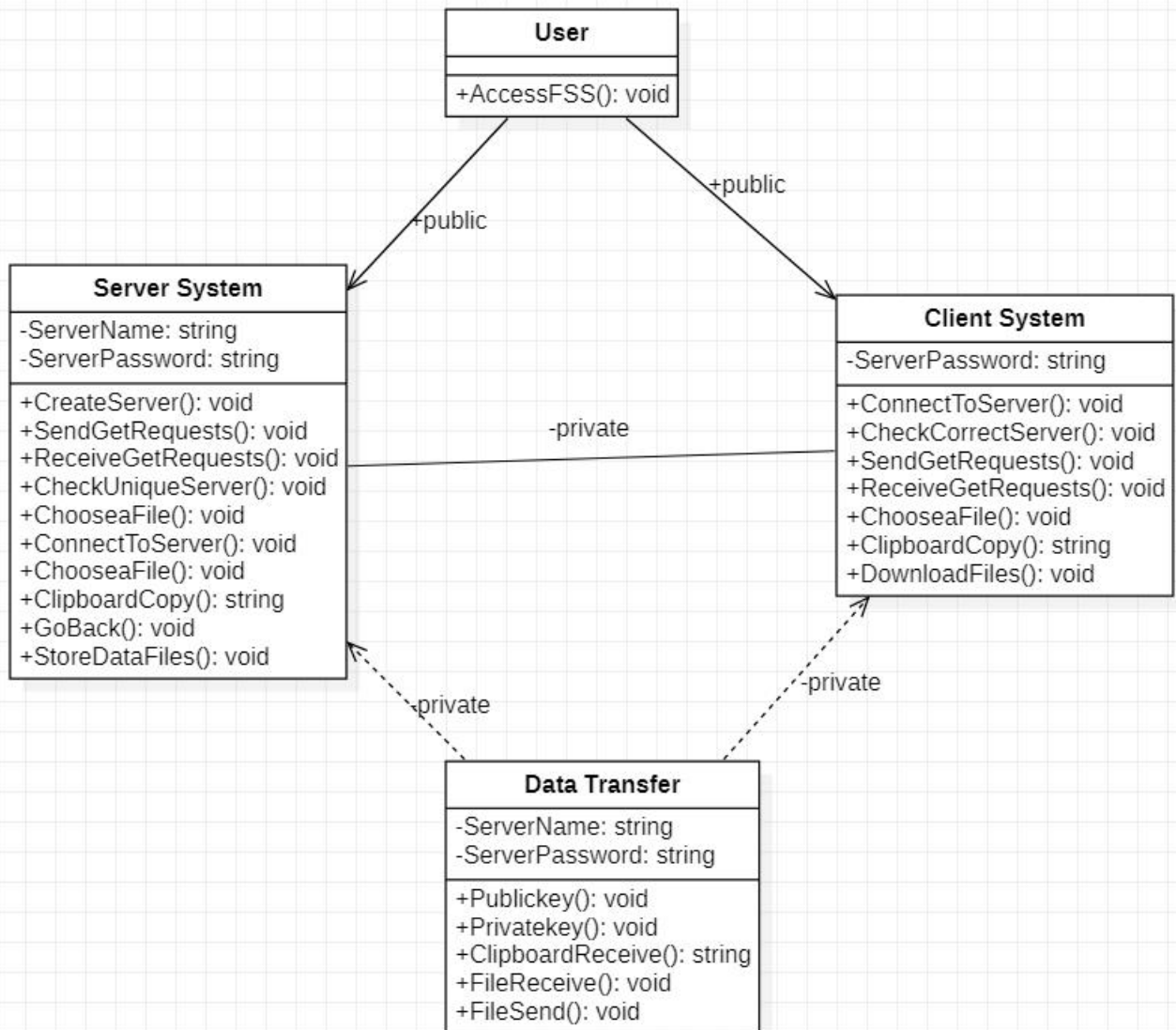
### Server Module:



## Client Module:



## Class Diagram:



# Architecture

The software needs the architectural design to represent the design of software. IEEE defines architectural design as “the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.” The software that is built for computer-based systems can exhibit one of these many architectural styles.

Of the many architectural styles that are present, our File Sharing System simply follows ‘Data Centric Architecture’.

What is Data Centric Architecture?

Data Centric Architecture is one of the many architectural styles in Software Engineering where the data is centralized and accessed frequently by other components, which modify data. The main purpose of this style is to achieve integrality of data. Data-centered architecture consists of different components that communicate through shared data repositories. The components access a shared data structure and are relatively independent, in that, they interact only through the data store.

There are two types of components –

A central data structure or data store or data repository, which is responsible for providing permanent data storage. It represents the current state.

A data accessor or a collection of independent components that operate on the central data store, perform computations, and might put back the results.

Interactions or communication between the data accessors is only through the data store.

Similarly, our system consists of the Server System that acts as a central data structure or data store that stores all the files and data. It provides permanent data storage.

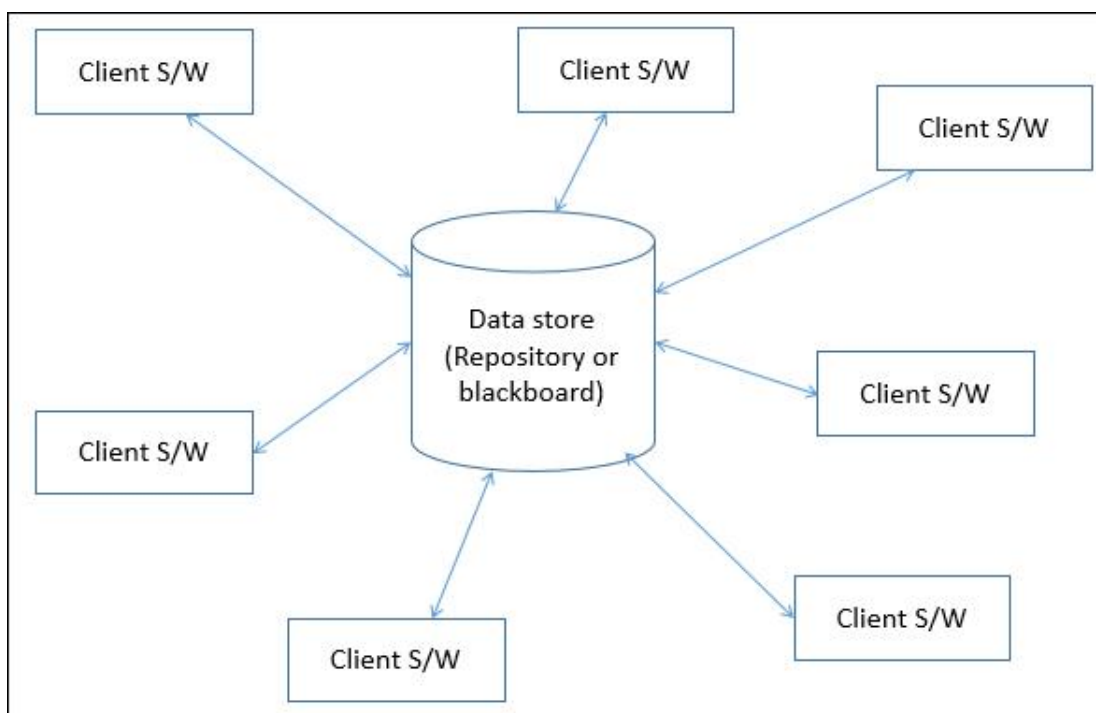
On the other hand, there are number of client systems that can connect to the Server System to initiate file transfer. These client systems send requests to the Server to establish connection and it is responsible for the logical flow. Therefore, the Client Systems in this case, act as a data accessor.

The File Sharing System works in the presence of multiple client systems connected to a single

Server over a small privatized network, where upload and download files can take place.

This occurs over a passive central Server, that receives requests from the clients to perform actions like add files, or download files. These computational processes are independent and triggered by incoming requests. Also, the existing components can be changed and new client components can be added to the architecture without the permission or concern of other clients.

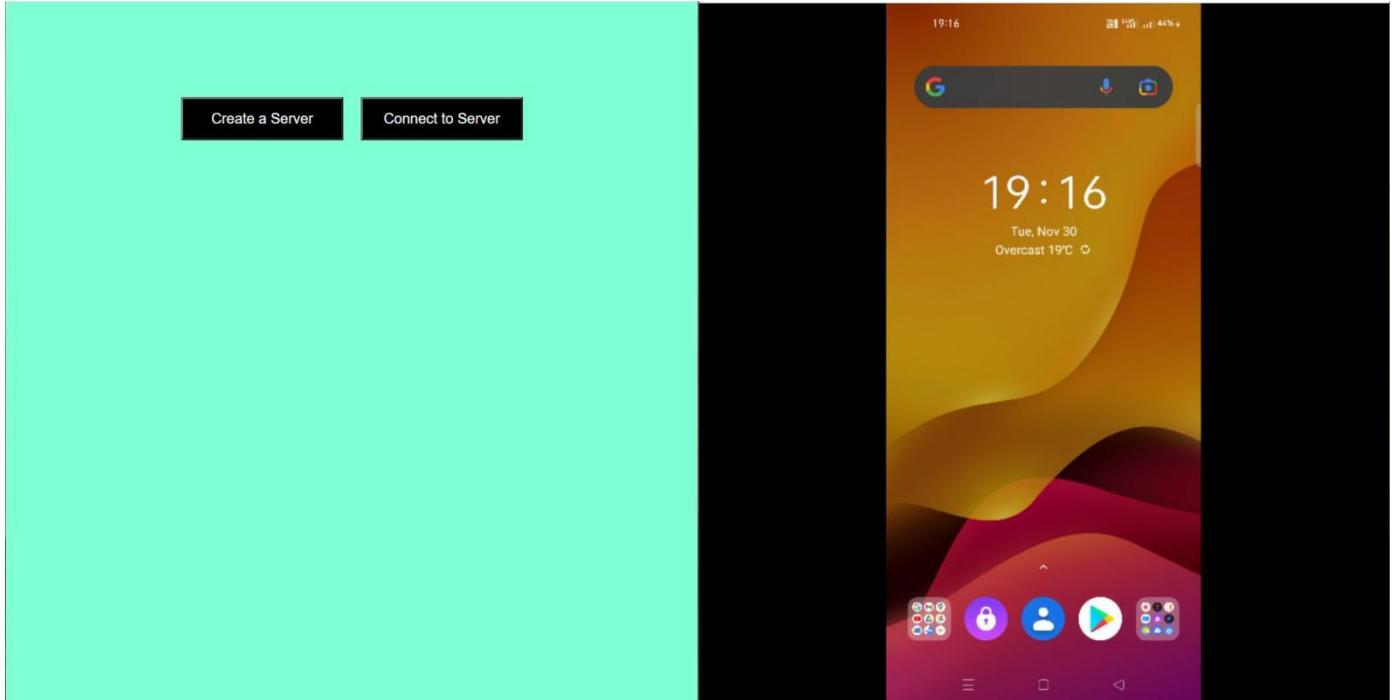
Therefore, the File Sharing System comes under the Repository Architecture Style.



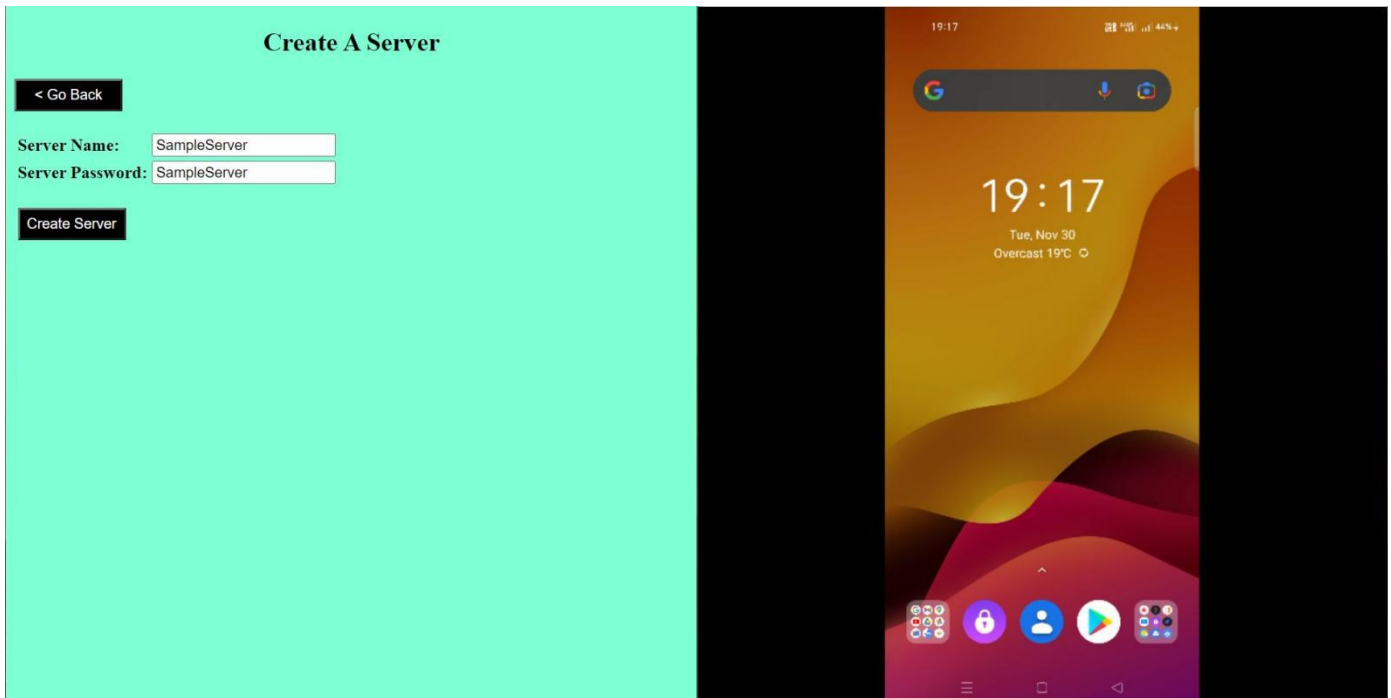
# Graphical User Interface (Screenshots)

The steps that the user must undergo are as follows:

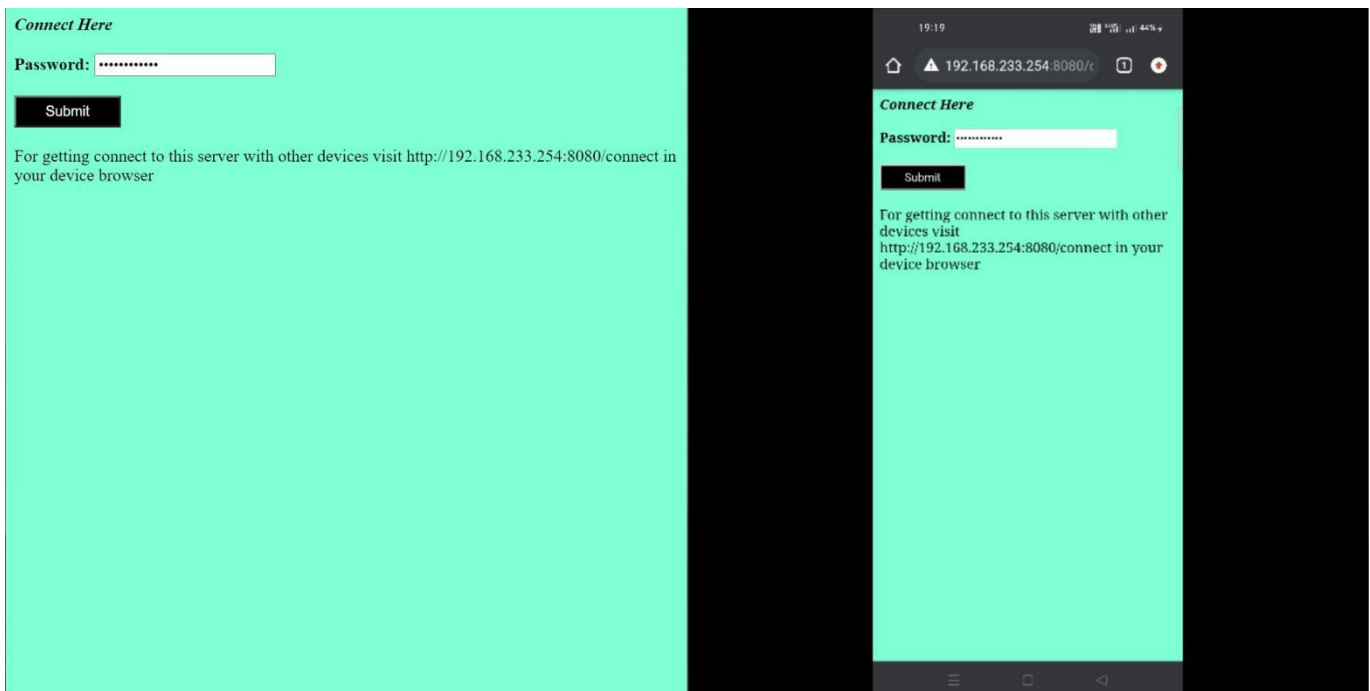
A. Determine the Server System and Client. Here, we use two devices to show the procedure. We click on 'Create a Server' to create a server on the device.



B. After clicking on 'Create a Server' the system navigates to a page where we have to enter the name of the Server that we want to create. Then, we also have to enter a password which will be used by all the clients that want to connect to this server.

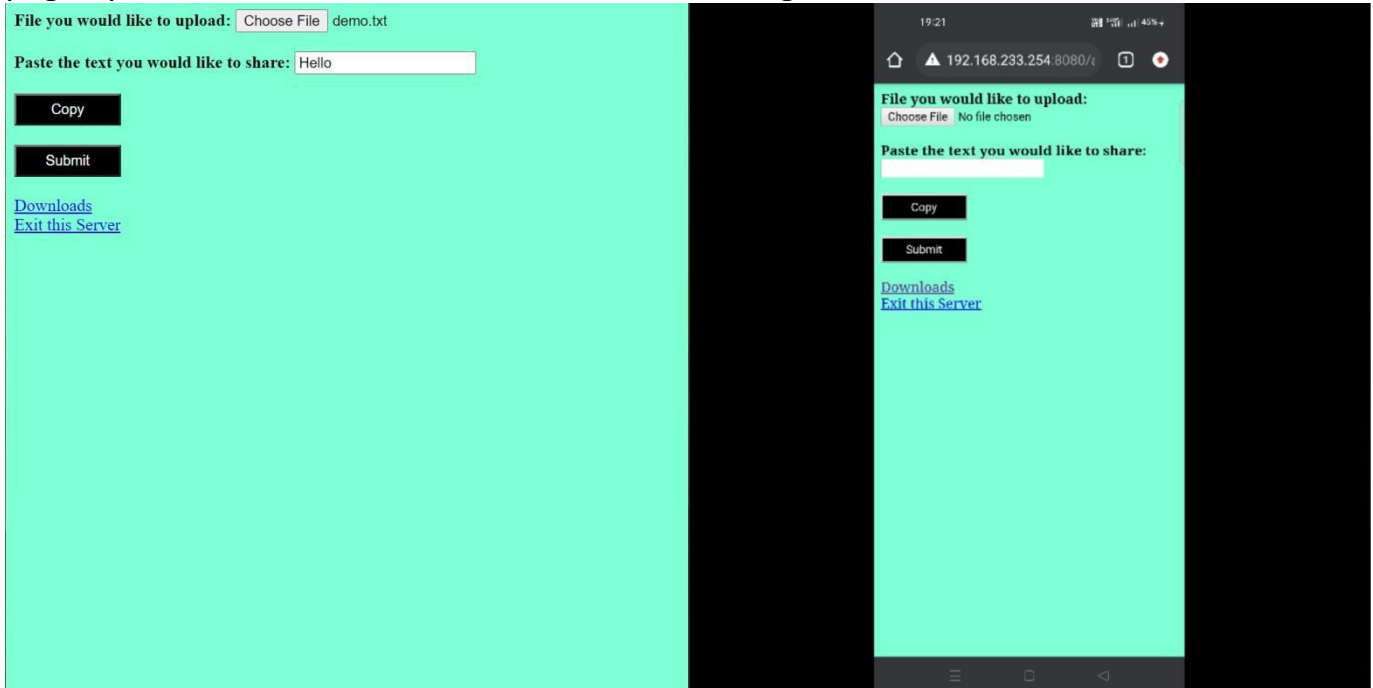


C. After successfully creating a server, the device that we created the server on will also become a client to the server. We will be navigated to a page where we have to enter a password to connect to the server. This is the same password that we entered while creating the server. We also get a URL that we use to open the system on other clients. We use this URL to open the File Sharing System on our second device, that is, our mobile phone. The second device is also a client to the server. We have to enter the password here as well. Then we click on 'Submit'.

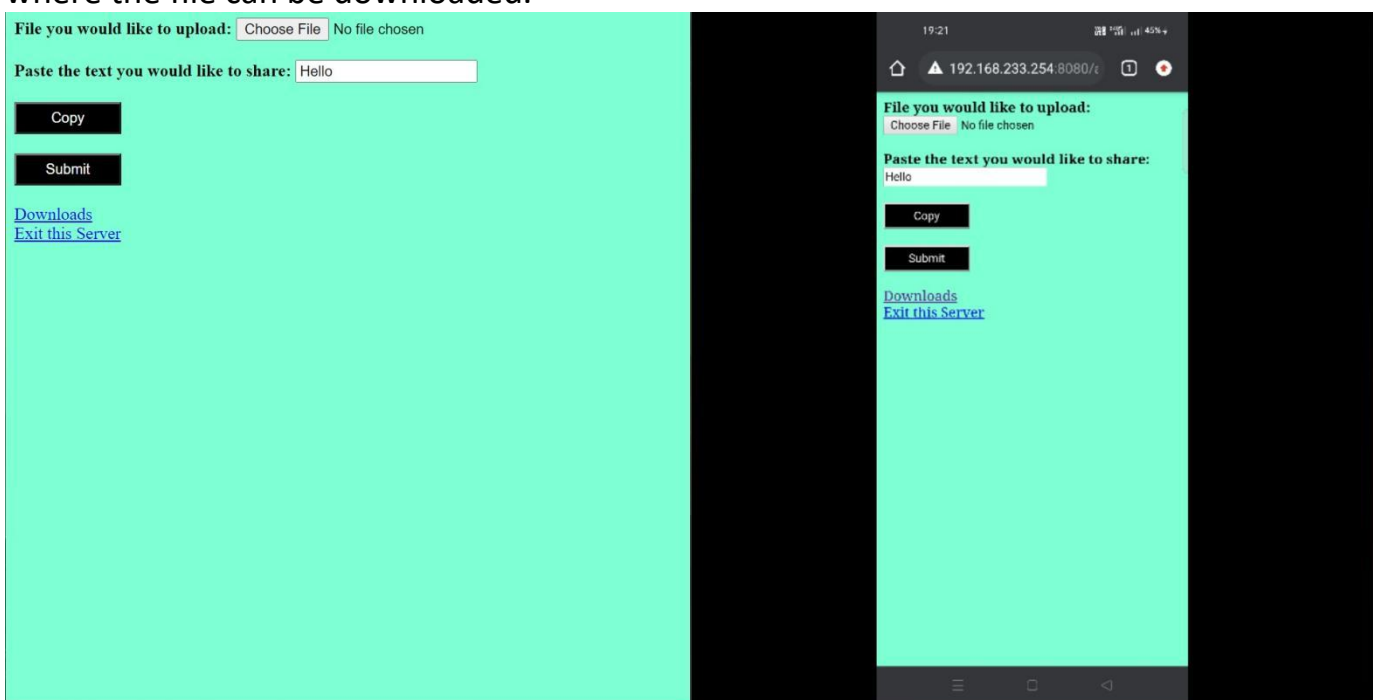




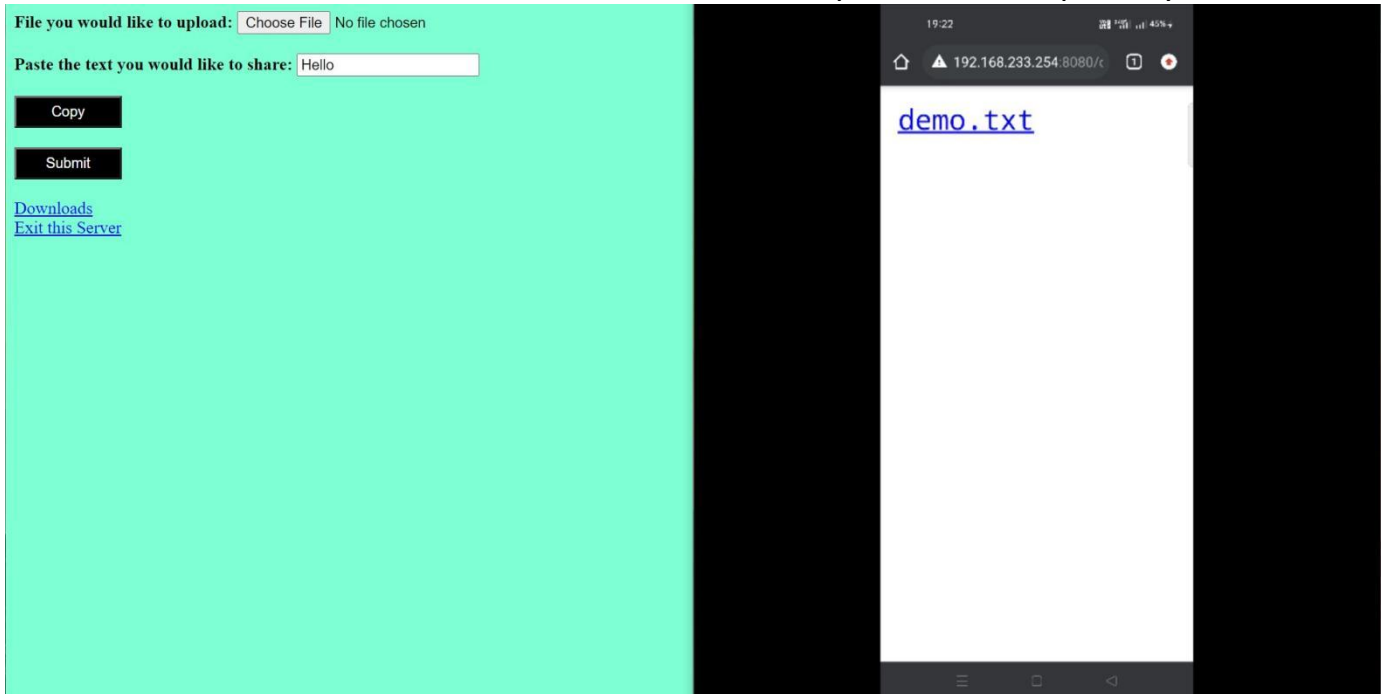
D. Then we are taken to a page where we can choose the files that we want to share. Here, we have taken the file 'demo.txt' for demonstration. One of our major features was the 'Shared Clipboard'. To share the clipboard, we type the text that we want to share. After selecting the file and typing the text to be shared, we click on submit. Then we refresh the page open on the second device to notice the changes.



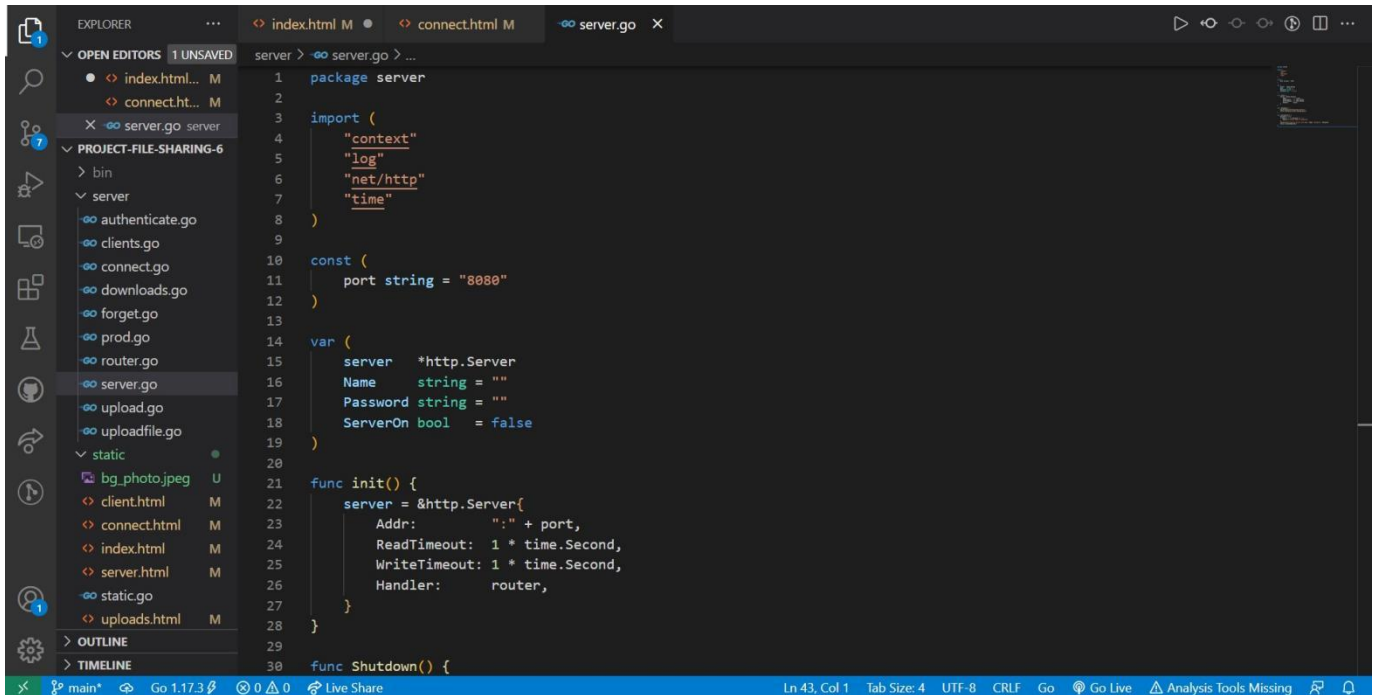
E. After refreshing, we can see that our shared clipboard ('Hello') is visible on the second device. To see the files that we had shared, we must click on the 'Downloads' button from where the file can be downloaded.



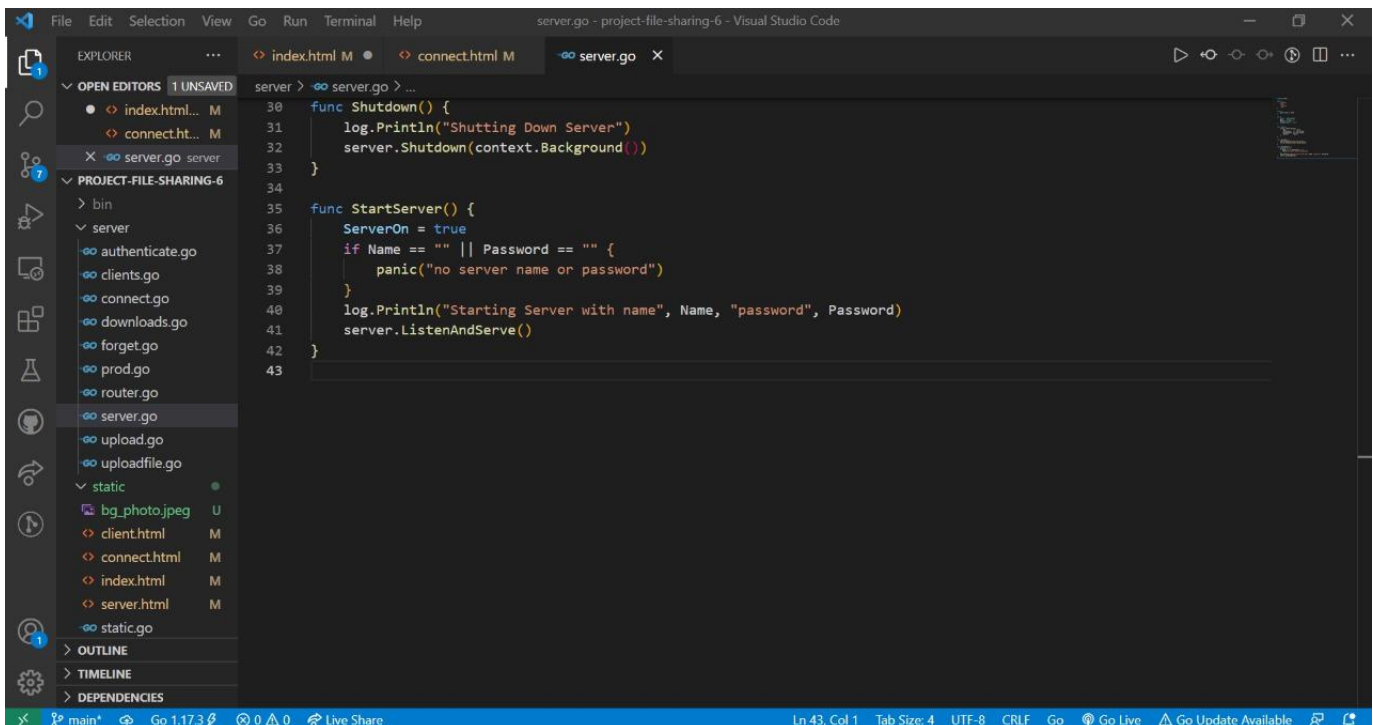
F. After clicking on 'Downloads' on the second device, we can clearly see that we have received the file that we had shared initially. We click on this file in order to save it on our device. The same file will be visible in a folder called 'Uploads' on the primary device.



# Source Code (Sample)



```
1 package server
2
3 import (
4     "context"
5     "log"
6     "net/http"
7     "time"
8 )
9
10 const (
11     port string = "8080"
12 )
13
14 var (
15     server *http.Server
16     Name    string = ""
17     Password string = ""
18     ServerOn bool = false
19 )
20
21 func init() {
22     server = &http.Server{
23         Addr:    ":",
24         ReadTimeout: 1 * time.Second,
25         WriteTimeout: 1 * time.Second,
26         Handler:    router,
27     }
28 }
29
30 func Shutdown() {
```



```
30 func Shutdown() {
31     log.Println("Shutting Down Server")
32     server.Shutdown(context.Background())
33 }
34
35 func StartServer() {
36     ServerOn = true
37     if Name == "" || Password == "" {
38         panic("no server name or password")
39     }
40     log.Println("Starting Server with name", Name, "password", Password)
41     server.ListenAndServe()
42 }
43
```

```
static > index.html > ...
You, seconds ago | 2 authors (You and others)
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>FSS</title>
9 </head>
10
11 <body>
12   <style>
13     body {
14       background-color: aquamarine;
15     }
16
17     .button_css {
18       background-color: black;
19       color: white;
20       width: 150px;
21       height: 40px;
22     }
23
24     .container {
25       height: 200px;
26       position: relative;
27     }
28
29     .center {
```

```
static > index.html > ...
29 .center {
30   display: flex;
31   justify-content: center;
32   align-items: center;
33   height: 200px;
34 }
35 </style>
36 <div class="container">
37   <div class="center">
38     <button style="margin-right:16px" class='button_css' onclick="createServerHTML()">Create a Server</button>
39     <button class='button_css' onclick="connectToServerHTML()">Connect to Server</button>
40   </div>
41 </div>
42 </body>
43 </html>
44
45
46
```

## Verification & Validation

*\*The Test Case Document has been zipped along with this report. Since ours is a desktop application, it was not possible for us to use Selenium for testing.*

# Conclusion

To conclude, our File Sharing System offers the basic features available in any application of this domain, at the same time, presents unique features, like unique security protocols and cross-platform availability. Incorporating file sharing between different Operating Systems is a necessity and enhances the customer experience as well.

The System was primarily implemented using Golang, to enable and amplify the Client-Server Model in our system. At the same time, to offer user-friendly interface, HTML and CSS were used for the Front-End. However, more focus was put into introducing new features that stand out from other File Sharing Systems.

The Overall effort and time put into this project was worth it and it allowed us to broaden our spectrum of knowledge. There is always more to discover and develop. Hence, as a team we are motivated to learn and upgrade our system.

Finally, we would like to thank VIT Chennai for providing us this opportunity and moreover, we would like to thank our mentor and teacher, Dr. Braveen M .

Thank you.

# References

HTTPS - <https://en.wikipedia.org/wiki/HTTPS>

UDP - [https://en.wikipedia.org/wiki/User\\_Datagram\\_Protocol](https://en.wikipedia.org/wiki/User_Datagram_Protocol)

GoLang GUI - <https://github.com/webview/webview>

GoLang Documentation - <https://golang.org/doc/>