Hanoi University of Science and Technology
School of Information and Communication Technology

* * *

PROJECT

NETWORK PROGRAMMING

FILE SHARING 2

Class : 139387
Instructor : Dr. Tống Văn Vạn
Group number : Group 2

Name	Student ID
Phạm Hải Đăng	20194736
Phạm Thành Biên	20194731
Hoàng Thanh Lâm	20194786

Hanoi, 7/2023

Table of Contents

Table of Figures

Table of Tables

Introduction

In today's interconnected world, the ability to share files across networks has become an integral part of our daily lives. From sharing documents with colleagues to exchanging multimedia files with friends and family, file sharing has transformed the way we collaborate and communicate. Network programming plays a crucial role in enabling efficient and secure file sharing among devices connected to a network.

Network programming refers to the development of software applications that facilitate communication and data exchange between devices connected over a network. It involves designing protocols, implementing network sockets, and utilizing various network communication technologies to establish seamless connections between devices. File sharing, as a subset of network programming, focuses specifically on the efficient and reliable transfer of files between different systems.

The advantages of file sharing through network programming are numerous. It allows users to access files stored on remote systems without the need for physically transferring storage media. This capability is especially valuable in scenarios where geographically dispersed individuals need to collaborate on a project or access shared resources. Additionally, file sharing reduces redundancy by eliminating the need for multiple copies of the same file, improving storage efficiency.

However, with the convenience of file sharing comes the need for robust security measures. Network programming for file sharing must address concerns such as data privacy, authentication, and access control. Encryption protocols and secure communication channels are essential to protect sensitive information from unauthorized access or interception.

In this project, we will delve into the realm of network programming for file sharing. We will explore various concepts and techniques to design and develop an efficient and secure file sharing system. By understanding the underlying principles and implementing practical solutions, we aim to enhance the file sharing experience while maintaining the integrity and confidentiality of shared data.

Through this project, we hope to gain insights into the intricate world of network programming and its application to file sharing. By examining different protocols, algorithms, and security mechanisms, we strive to contribute to the growing field of network programming while addressing the evolving needs of modern file sharing.

Due to limited time and capabilities, it is inevitable to make mistakes. Therefore, our team hopes to receive straightforward feedback from you and our classmates.

Finally, Group 2 would like to sincerely thank Mr. Tống Văn Vạn for guiding us throughout the time we worked on this report.

Sincerely

Group 2

Contribution

Name	ID	Contribution	Evaluation
Phạm Thành Biên	**20194731**	- Design program architecture - Design user interface - Program functionalities: - Create, join group - View, accept join request(admin) - View group list, files/folder list	Successfully completed the assigned task
Phạm Hải Đăng	20194736	- Program user interface - Program functionalities: - Upload, download file - Create, delete folder - Document preparation	Successfully completed the assigned task
Hoàng Thanh Lâm	20194786	- Design user interface - Program functionalities: - Sign in - Sign up - Document preparation	Successfully completed the assigned task

Table 1: Contribution table

Chapter 2. Project Introduction

1.1. State the problem

The problem addressed in this network programming project is the need for efficient and secure file sharing across networks. While file sharing has become essential for collaboration and communication, there are challenges associated with transferring files between devices over a network. These challenges include ensuring reliable data transfer, maintaining data integrity, optimizing file sharing performance, and ensuring data security.

Reliable data transfer: One of the key challenges in file sharing is ensuring that files are transferred accurately and without loss or corruption. Network disruptions, packet loss, and latency can all affect the reliability of data transfer, leading to incomplete or corrupted files.

Data integrity: Maintaining the integrity of files during the transfer process is crucial to ensure that the received files are identical to the original ones. Any alteration or corruption of data during the transfer can render the shared files useless or compromised.

Performance optimization: Efficient file sharing requires optimizing the performance of the network and the file transfer protocols. Large file sizes, limited bandwidth, and varying network conditions can all impact the speed and efficiency of file transfers. Ensuring optimal performance is vital to reduce transfer times and enhance the overall user experience.

Data security: File sharing over networks raises concerns about data security. Unauthorized access, interception, or tampering of shared files can lead to the exposure of sensitive information. Implementing robust security measures, such as encryption, authentication, and access control, is crucial to protect the confidentiality and integrity of shared files.

In this network programming project, we aim to address these challenges and develop a file sharing system that ensures reliable and secure file transfers across networks. By implementing efficient protocols, optimizing performance, and integrating robust security measures, we seek to enhance the file sharing experience while mitigating the risks associated with data transfer over networks.

1.2. Objectives and Scope of the Project

The objectives of this network programming project on file sharing are as follows:

Develop a reliable file sharing system: The primary goal is to design and implement a file sharing system that ensures the reliable transfer of files between devices connected over a network. This includes addressing issues such as network disruptions, packet loss, and latency to minimize the chances of data loss or corruption during file transfers.

Enhance file sharing performance: The project aims to optimize the performance of the file sharing system, considering factors such as file size, network bandwidth, and varying network conditions. By implementing efficient transfer protocols and employing techniques like data compression and parallel processing, we aim to reduce transfer times and enhance the overall speed and efficiency of file sharing.

Ensure data integrity: The project focuses on implementing mechanisms to maintain the integrity of shared files during the transfer process. By incorporating error detection and correction techniques, checksums, and validation mechanisms, we aim to ensure that the received files are identical to the original ones, with no alterations or corruptions.

Implement robust security measures: Data security is a critical aspect of file sharing. The project aims to incorporate secure communication channels, encryption protocols, authentication mechanisms, and access control measures to protect shared files from unauthorized access, interception, and tampering.

The scope of this project encompasses the development and implementation of a file sharing system with the mentioned objectives. The project will involve designing and coding the necessary protocols, algorithms, and interfaces required for file transfer. The focus will be on developing a system that can be tested and evaluated using simulated network environments.

1.3. Solution Approach

To address the challenges and achieve the objectives of efficient and secure file sharing in network programming, we propose the following solution approach:

Network Protocol Selection: We use TCP/IP protocol and evaluate and select appropriate network protocols for file sharing, considering factors such as reliability, performance, and security

Chapter 2. Survey and Analysis of Requirements

2.1. Overview of Functionality

The application will have the following functions:

- Login: Users must be login to use this app.
- Register: Users can register account if they don’t have any yet.
- Logout: When a user want to logout from the system.
- When login to the app, main screen have many function:
    - Create a group. The user who created the group is the group leader.
    - Each group has its own directory, which contains files shared within that group.
    - Any member in each group can upload files and create subdirectories within the group’s directory.
    - Only group leader can delete file, subdirectories.
    - Any user can request to join a group.
    - Download file from own directory group.

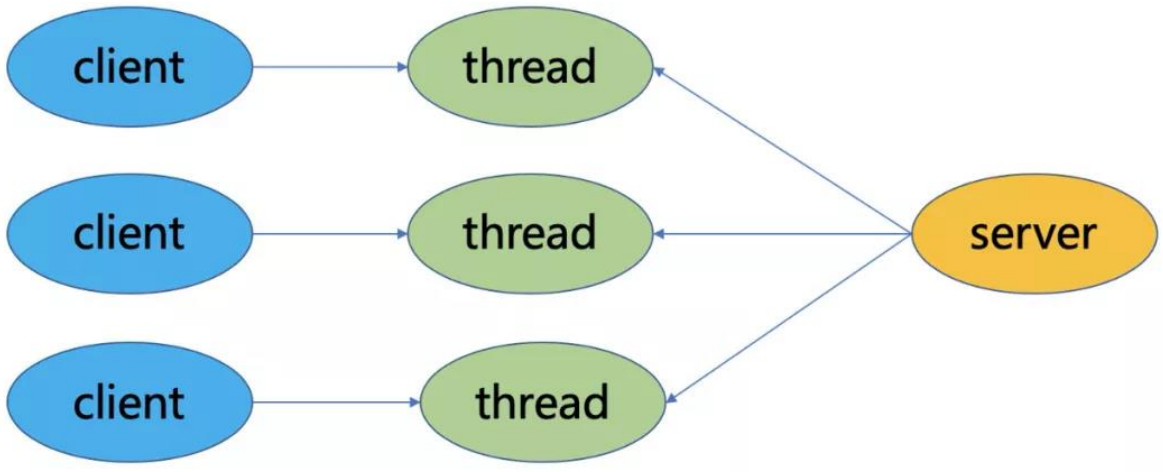


Figure 1: System architecture

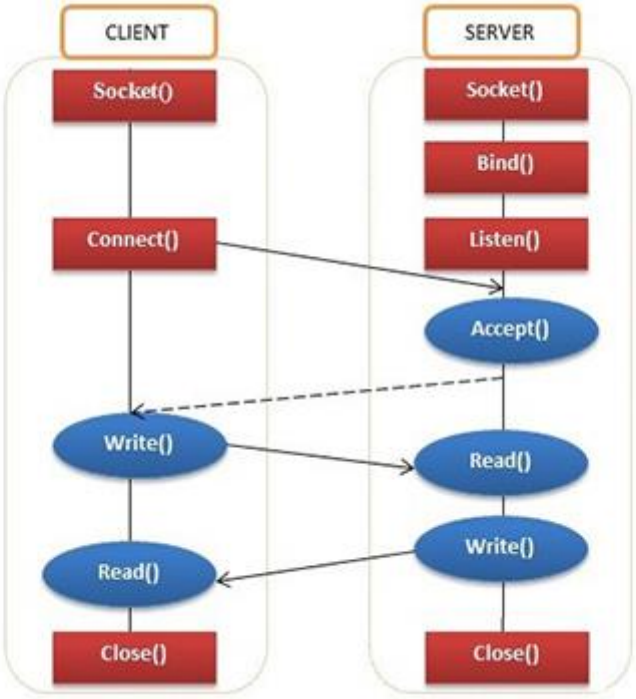


Figure 2: Communication between client -server

2.2. Function overview

2.2.1. Usecase diagram

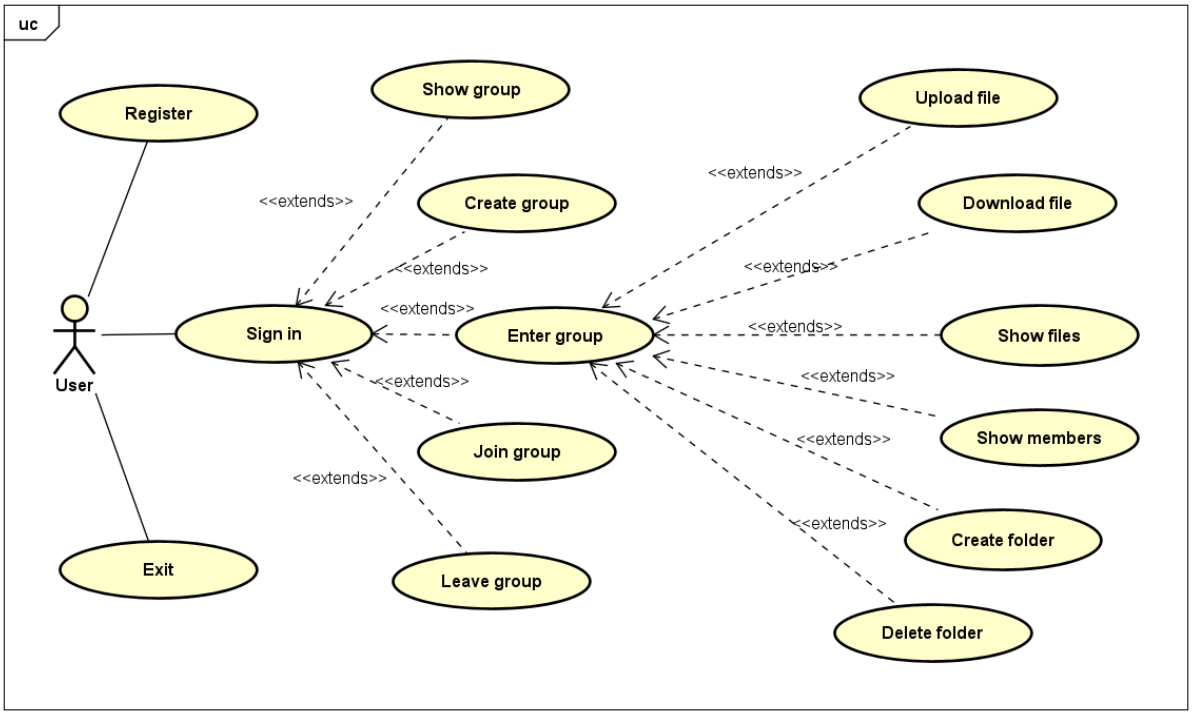


Figure 3: Usecase diagram

2.2.2. Detailed usecase diagram

- Actors : Users
- Use cases:
  - Register
  - Sign in
  - Exit
  - Show group
  - Create group
  - Enter group
  - Join group
  - Leave group
  - Upload file
  - Download file
  - Show files
  - Show members
  - Create folder
  - Delete folder
- Relation: Show group, Create group, Enter group, Join group, Leave group extends from login. Upload file, Download file, Show files, Show members, Create folder , Delete folder extends from Enter group.

2.2.3. Business processes

- Login/ Register/ Logout process:

  Users can choose sign in if they already have account.

  Users choose register account if they don't have any account yet. Users need to enter user name and password to check and registration process.

  When not thing to do, users can logout system to quit.

- Share file process:
  After login, user must be join in group or create a group to share file.
  User can choose show group to know which group they are in.
  If user don't have any group, they can send a request to group leader and wait for accept. Another function in main screen is that they can leave group.
  When user enter a group:
  + If they are group leader they can upload file, download file, show member in the group, create a folder ang delete a folder.
  + If they are a usual member they only have right to upload, download file, show file.

2.3. Function specification

2.3.1. Description of use case “Sign in”

Brief description:

- This use case describes flow user sign in to app.
  Actor: User

Basic flow of Events:

Step 1. User open app.

Step 2. User choose to login.

Step 3. User enter username.

Step 4. System for checking the username already exists..

Step 5. User enter password.

Step 6. System check password correct or not.

Step 7. User login successfully.

Alternative flows

No.	At step	Conditions	Actor	Action	Cont. at step
1	1-3		User	Username doesn’t exist.	
			System	Notify User to input again	
2	6		User	User enter wrong password.	
			System	Notify User to input again	
3	6	Wrong password 3 times	System	Block account	2

Table 2: Usecase Signin alternative flows

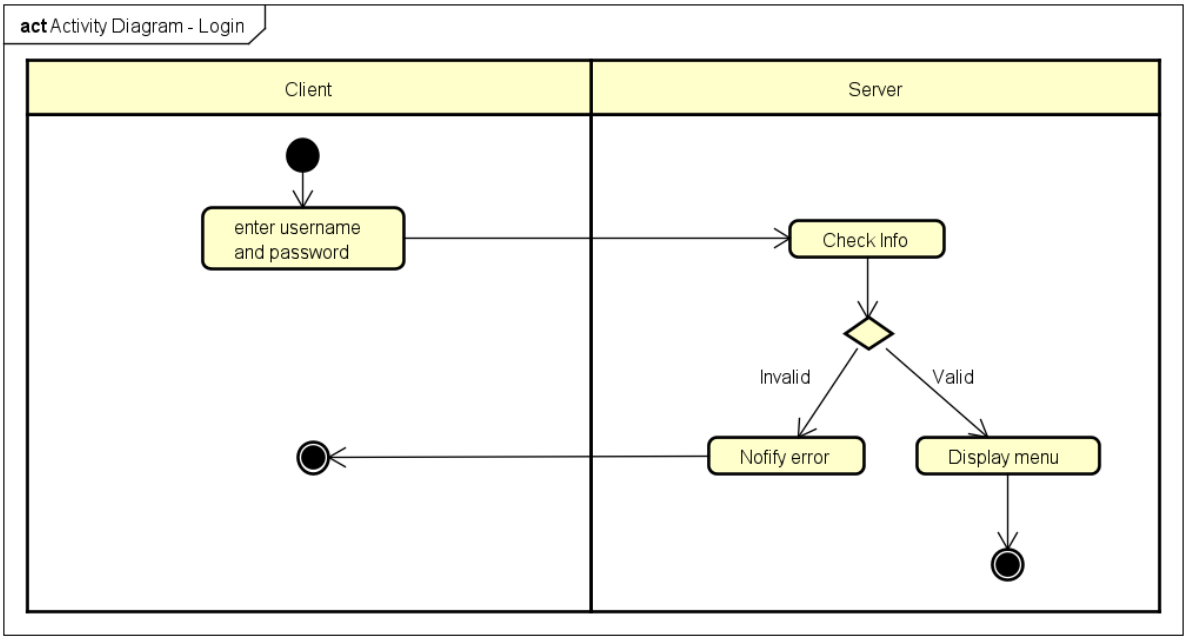


Figure 4: Signin activity diagram

2.3.2. Description of use case "Register account"

Brief description:

- This use case describes flow user register an new account.

Actor: User

Basic flow of Events:

Step 1. User chooses register account.

Step 2. User enter username.

Step 3. User enter password.

Step 4. User re-enter password.

Step 5. System display successful registration and save to database.

Alternative flows:

No.	At step	Conditions	Actor	Action	Cont. at step
1	1-2		User	Username is already exist.	
			System	Notify User to input again	
2	4		User	User enter wrong password compare to previous password	
			System	Notify User to input again	

Table 3: Usecase Register account alternative flow

2.3.3. Description of use case "Create group"

Brief description:

- This use case describes flow user create a group to share file.

Actor: User

Basic flow of Events:

Step 1. User chooses create group.

Step 2. User enter group name.

Step 3. System successful create a new group, mark user who create group to leader and save to database.

Alternative flows:

No.	At step	Conditions	Actor	Action	Cont. at step
1	2		User	Group name is already exist.	
			System	Notify error	

Table 4: Usecase Create group alternative flows

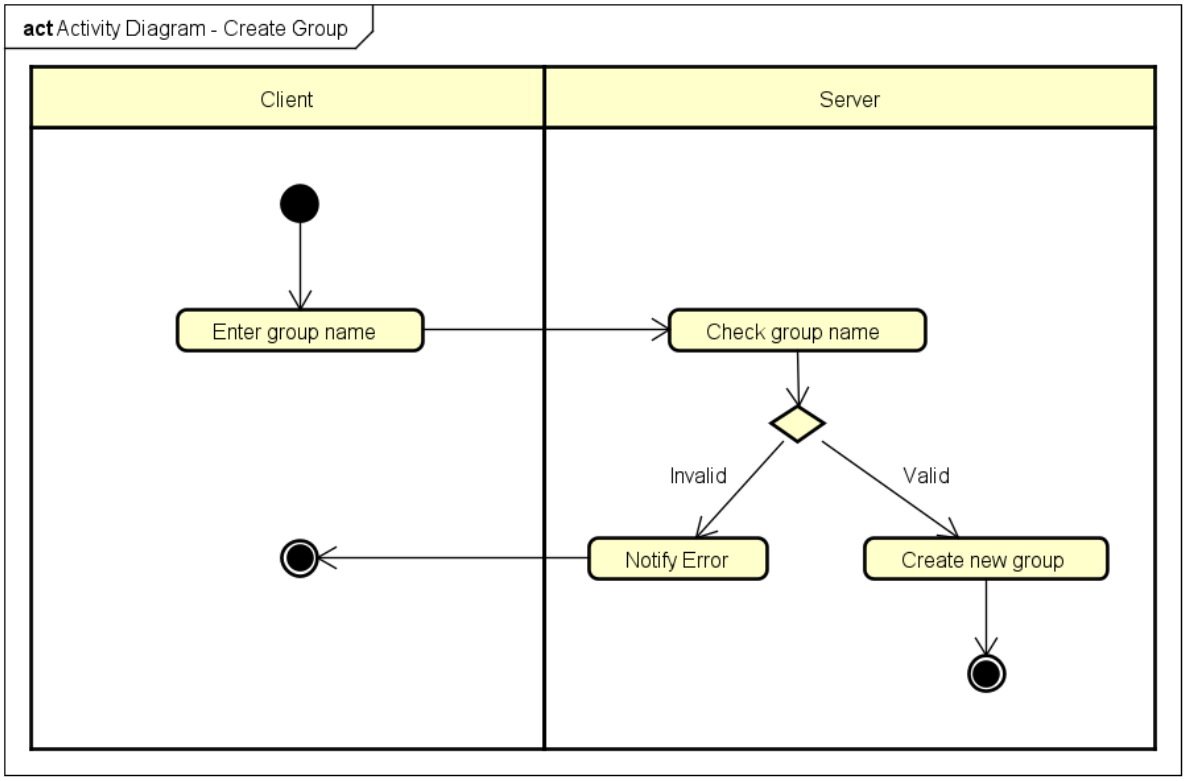


Figure 5: Create group activity diagram

2.3.4. Description of use case "Join group"

Brief description:

- This use case describes flow user join a group to share file.

Actor: User

Basic flow of Events:

Step 1. User chooses create group.

Step 2. User enter group name.

Step 3. System successful send a request to group leader

Step 4. Leader group accept to join.

Step 5. System add user to that group.

Alternative flows:

No.	At step	Conditions	Actor	Action	Cont. at step
1	3		User	Group name is not exist.	
			System	Notify error	
2	4		User	Group leader reject to user	
			System	Notify error	

Table 5: Usecase Join group alternative flows

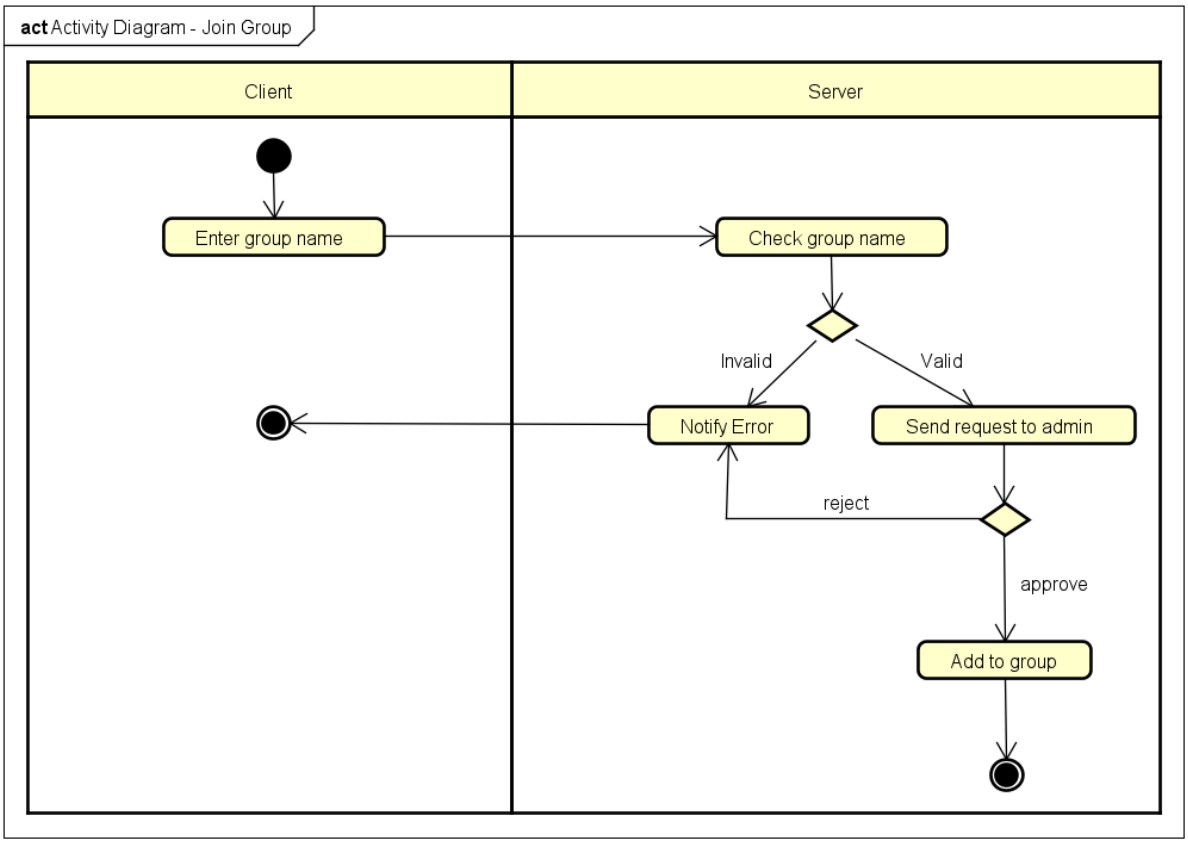


Figure 6: Join group activity diagram

2.3.5. Description of use case "Upload file"

Brief description:

- This use case describes flow user upload file.

Actor: User

Basic flow of Events:

Step 1. User enter file path name.

Step 2. System check file path.

Step 3. System upload the file.

Alternative flows:

No.	At step	Conditions	Actor	Action	Cont. at step
1	2		User	File path is wrong.	
			System	Notify error	

Table 6: Usecase Upload file alternative flows

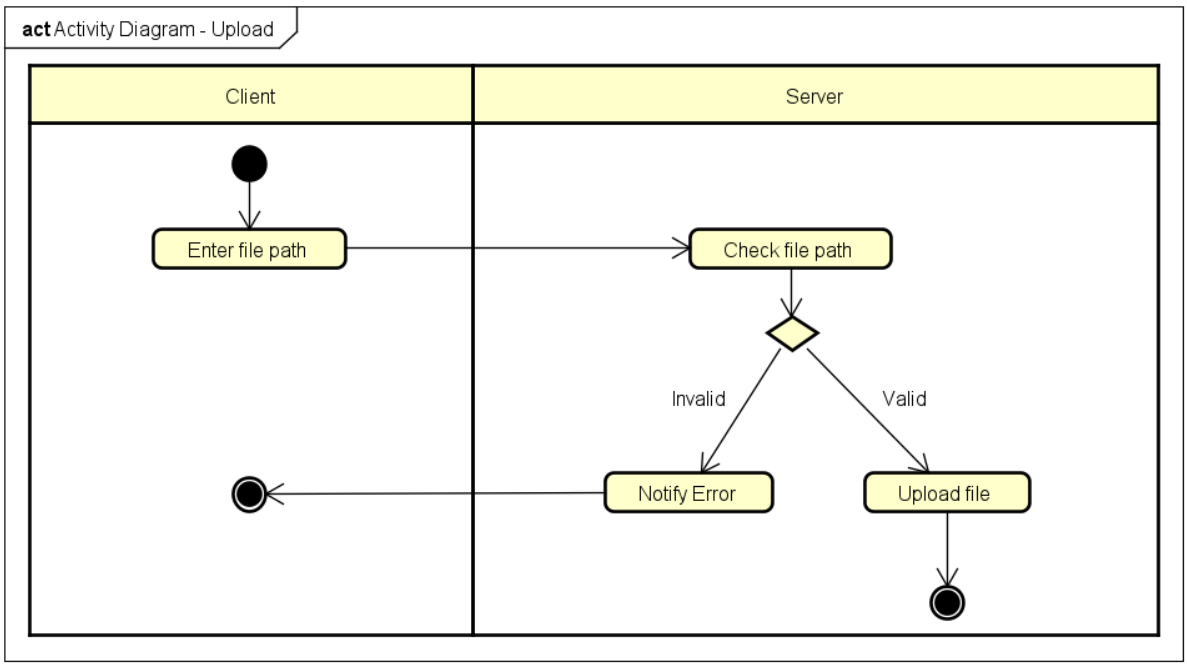


Figure 7: Upload file activity diagram

2.3.6. Description of use case “Download file”

Brief description:

- This use case describes flow user download file.

Actor: User

Basic flow of Events:

Step 1. User enter file path name.

Step 2. System check file path.

Step 3. System upload the file.

Alternative flows:

No.	At step	Conditions	Actor	Action	Cont. at step
1	2		User	File path is wrong.	
			System	Notify error	

Table 7: Usecase Download file alternative flows

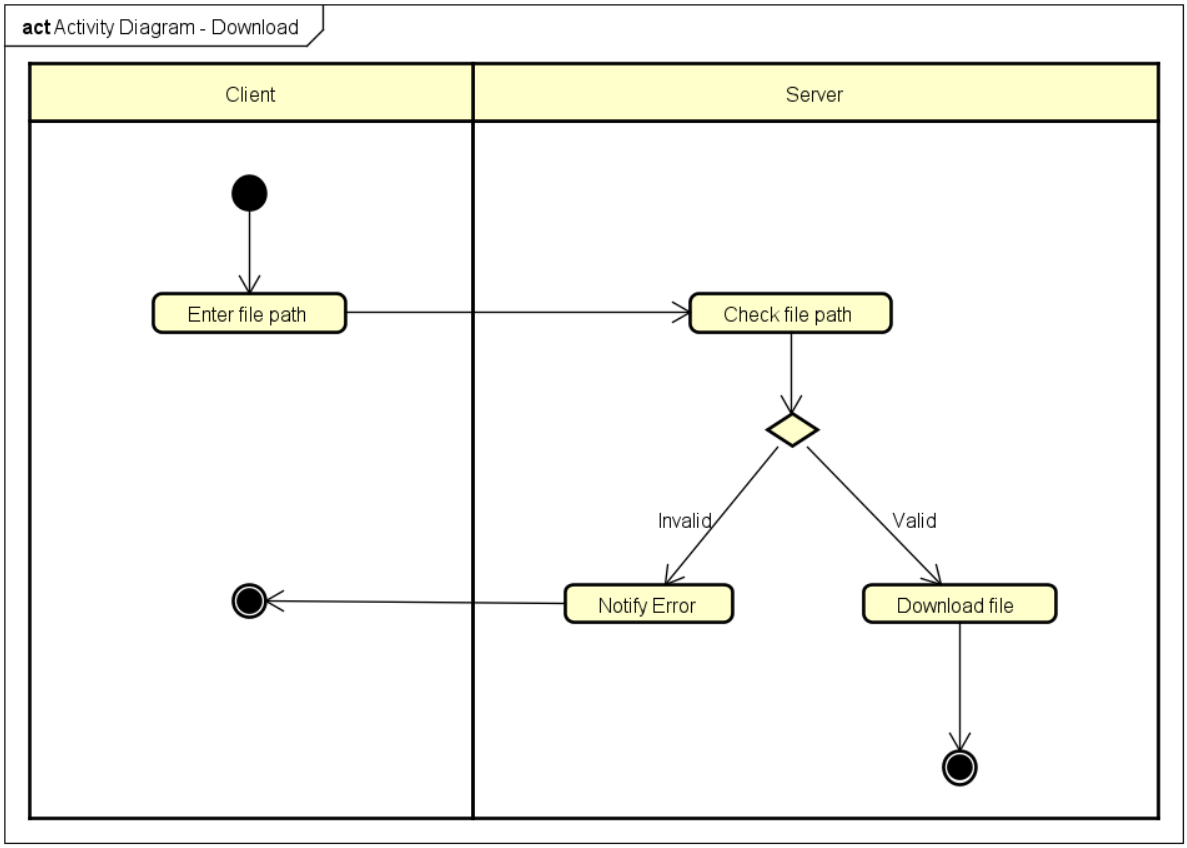


Figure 8: Download file activity diagram

Chapter 3. Methodology

In this project, our team use a new technique named as I/O Completion Ports (IOCP). It is a high-performance asynchronous I/O model, provides a scalable and efficient way to handle multiple I/O operations concurrently, making it well-suited for building high-performance network servers, file servers, and other I/O-intensive applications.

Advantages of I/O Completion Ports (IOCP):

1. Scalability: IOCP is designed for scalability and can efficiently handle a large number of concurrent connections. It utilizes a thread pool to manage I/O operations, allowing it to scale to thousands of connections with a small number of threads.
2. Asynchronous I/O: IOCP enables asynchronous I/O operations, meaning that I/O requests can be initiated without blocking the calling thread. When the I/O operation is completed, the IOCP notifies the appropriate thread from its thread pool, which can then handle the result.
3. Reduced Context Switching: IOCP significantly reduces the overhead of context switching compared to traditional synchronous I/O models. Since the I/O operations are performed asynchronously, there is less need for threads to wait for I/O to complete, leading to better overall performance.
4. Improved Throughput: By allowing simultaneous processing of multiple I/O requests, IOCP can achieve better throughput and reduce the time taken to handle I/O operations.
5. Support for Multiple I/O Types: IOCP can handle various I/O types, including network sockets, file I/O, and custom I/O devices, making it a versatile choice for I/O-intensive applications.
6. Proactive Resource Management: With IOCP, you can proactively manage resources, such as limiting the number of active connections, controlling buffer sizes, and handling graceful shutdowns effectively.

```
if ((completionPort = CreateIoCompletionPort(INVALID_HANDLE_VALUE, NULL, 0, 0)) == NULL) {
    cout << "CreateIoCompletionPort() failed with error " << GetLastError() << endl;
    return 1;
}

GetSystemInfo(&systemInfo);

for (int i = 0; i < (int)systemInfo.dwNumberOfProcessors * 2; i++) {
    if (_beginthreadex(0, 0, serverWorkerThread, (void*)completionPort, 0, 0) == 0) {
        cout << "Create thread failed with error " << GetLastError() << endl;
        return 1;
    }
}
```

Figure 9: Initiate IOCP

GUI: GTK library

GTK (GIMP Toolkit) is an open-source cross-platform library for creating graphical user interfaces (GUIs) in various programming languages. Originally developed for the GIMP (GNU Image Manipulation Program) graphics editor, GTK has evolved into a general-purpose toolkit widely used for building desktop applications on Linux, Windows, and macOS.

Our team choose GTK for its simplity, easy to implement. It also has some standard UI component as buttons, labels…. GTK provides a robust event handling system to manage user interactions with widgets and respond to events like button clicks, key presses, mouse movements, and more.

Chapter 4. Experiment and Evaluation

4.1. Application development

4.1.1. Libraries and tools

Purpose	Tool/Library	URL address
IDE	Visual Studio Code	https://code.visualstudio.com/
Programming language	C/C++	
UI	Gtk	https://www.gtk.org/

Table 8: Libraries and tools

4.1.2. Result

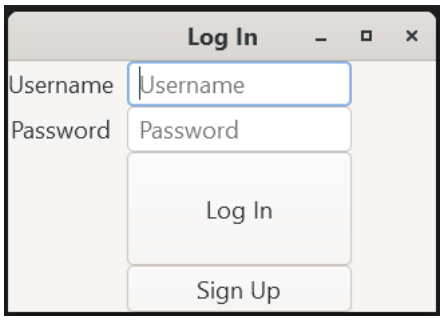
Our final product is a client-server application that allows users to share and interact with files. In detail, users can create new group, create folder, upload or download a file. If users are group leader, they can delete file or folder in that group.

Application information

Number of coding line	5640
Number of folder	2
Number of packet	14
Capacity of source code	20MB

Table 9: Application information

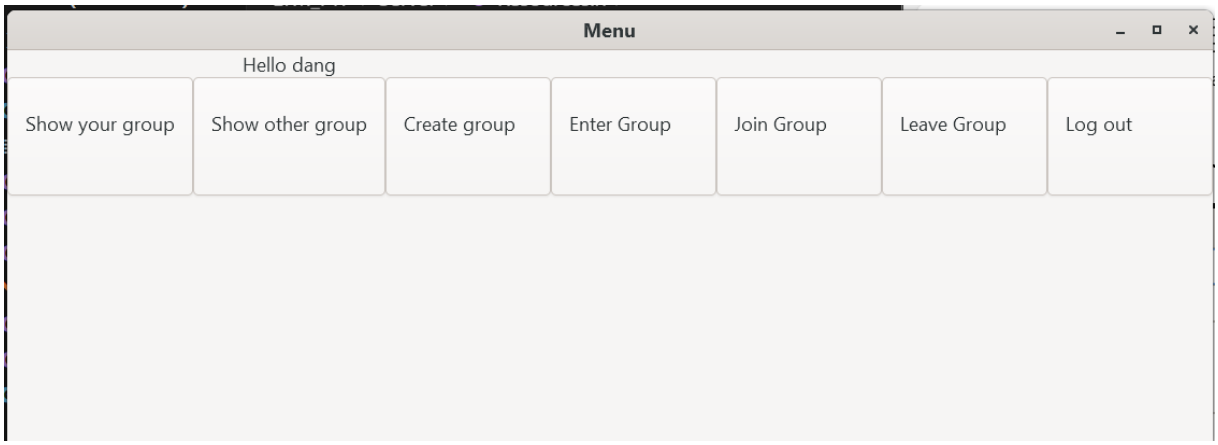
4.1.3. Demonstrate of main functionalities



After user open the application, login screen will open. User can choose login or signup function by clicking on button.

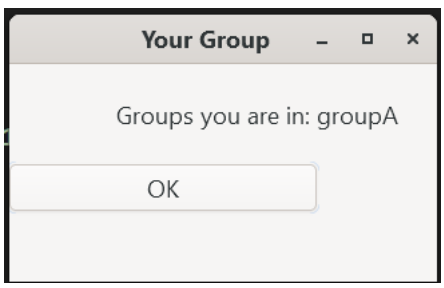
To login, user needs to fill up two fields: username and password

If user existed, system will display main menu as below

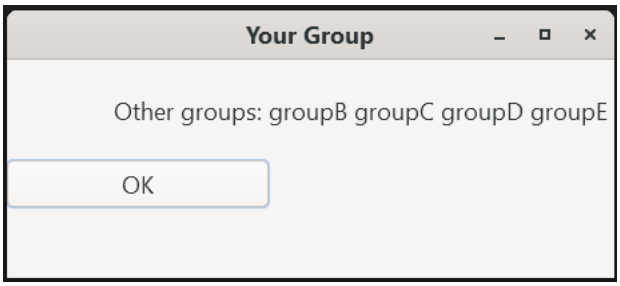


This is the main menu of user. User can show his/other groups, create a new group, enter joined group and register to join other group. User also can leave group and logout.

Now I choose ‘show your group’ from the menu screen

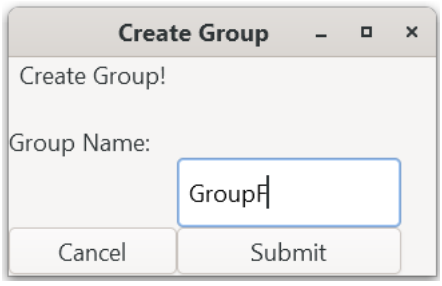


The system pop up a screen that notifies user’s group

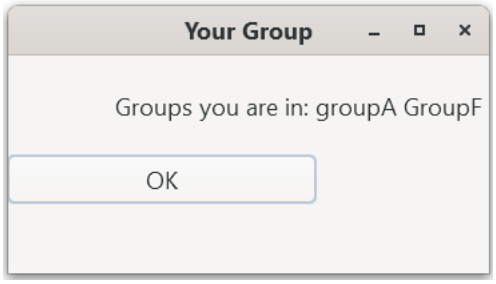


If user choose ‘create group’ function, the system will display a new screen.

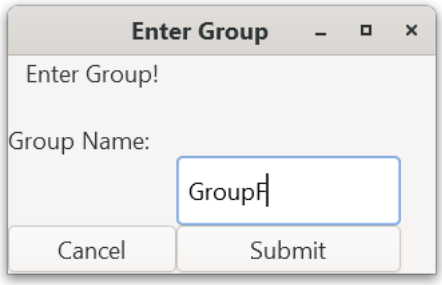
User needs to enter the group name and submit. If that group is not existed, a new group will be created and notifies to user



Now current user is in two group A and F



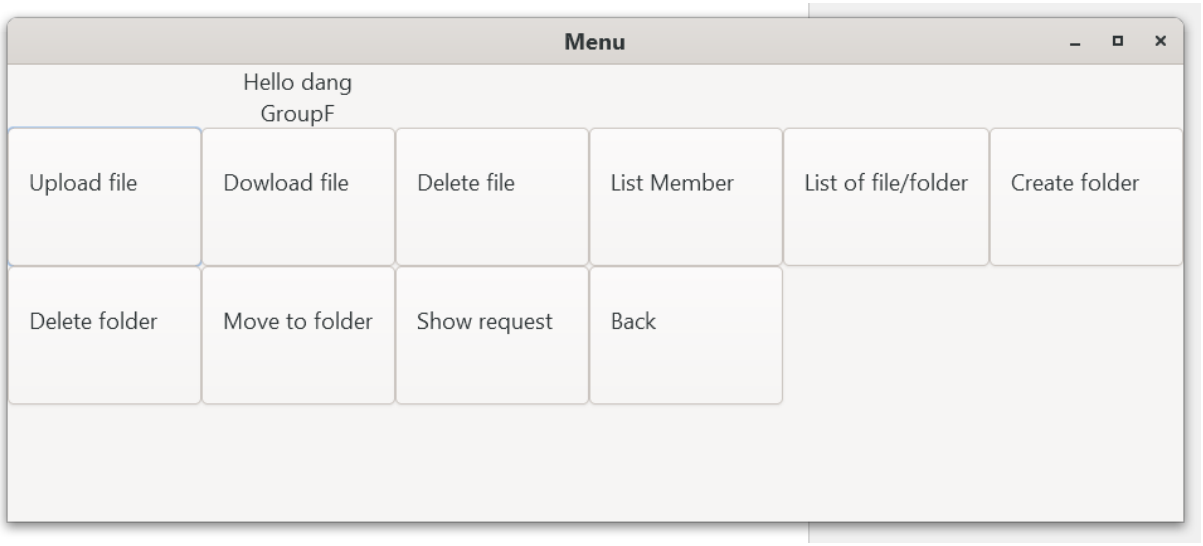
Now I will enter the group F that just been created, user choose ‘enter group’ function then enter the group name to enter



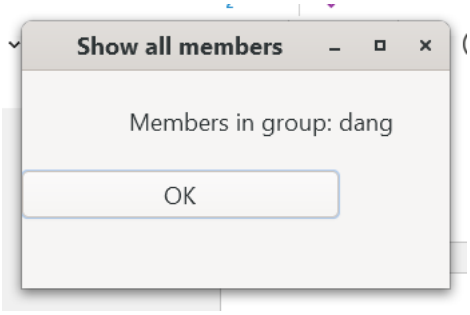
The system will check the input. If it is valid, the system will display the menu of that group function

The main functions here are upload/download file, list member/file/folder in that group and create folder…

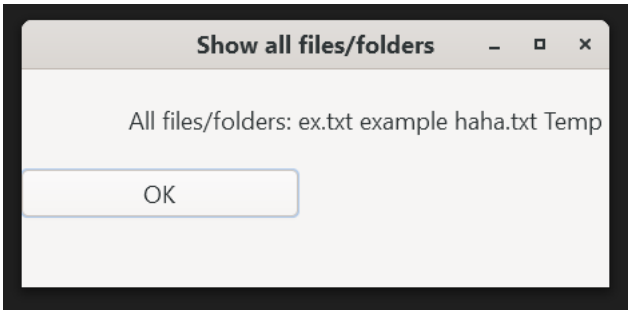
If user who creates that group, he has more function as delete folder and file



User lists the member of that group, and the system will display all the user's name

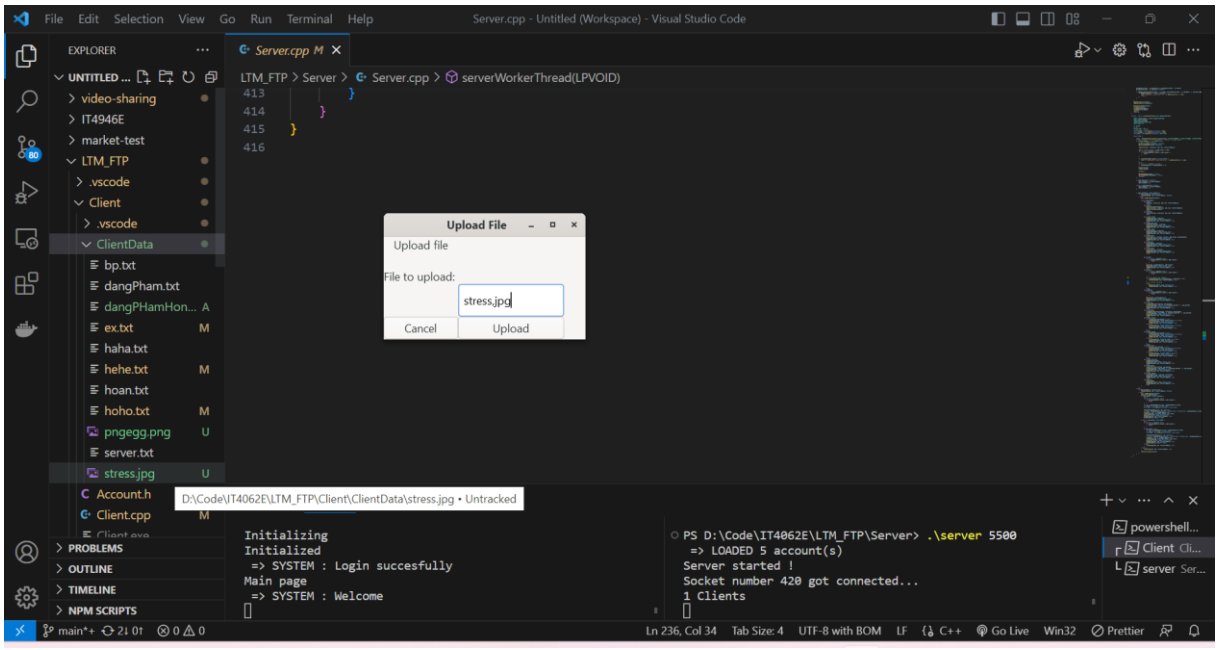


Following, user can see the folders and files existed in that group

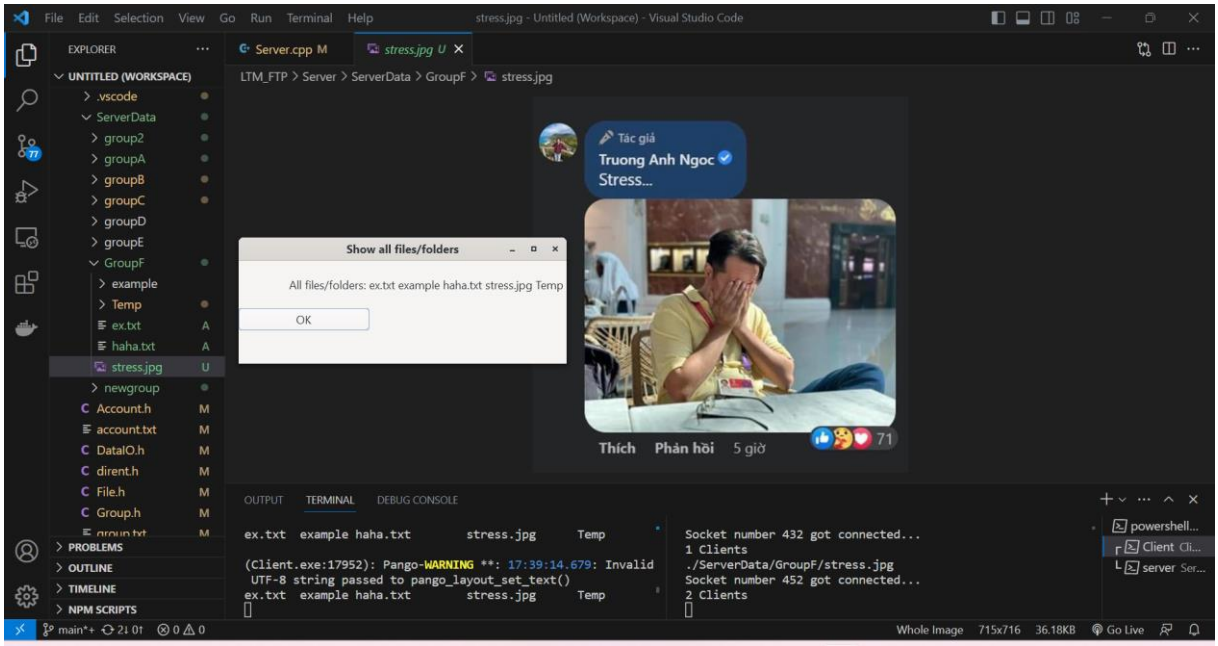


After entering a group, user can handle file in that group, for example, this is 'upload file' screen

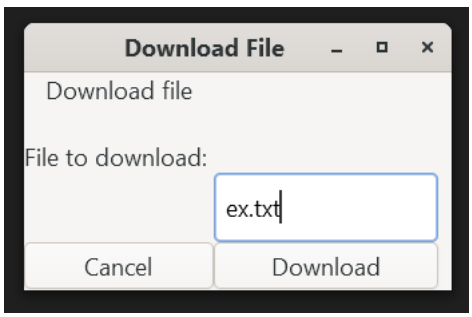
User enter the file path, for example, here we upload a image file named 'stress.jpg'



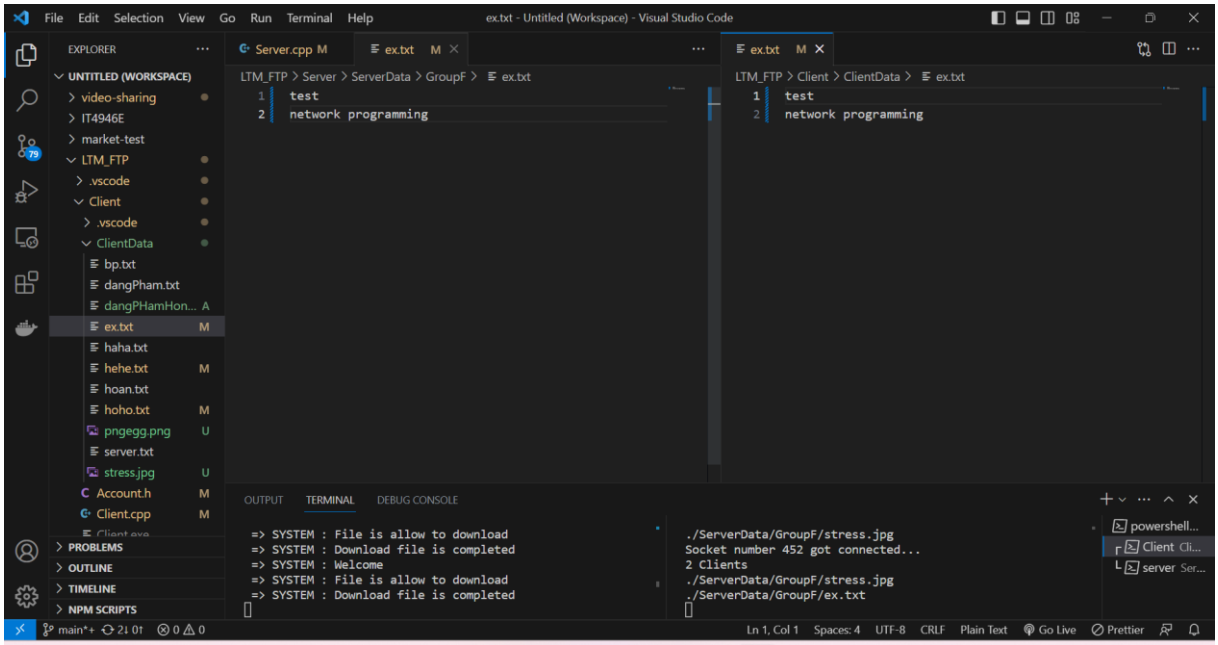
After processing, the system will store that file in group data folder and user can list the file to see



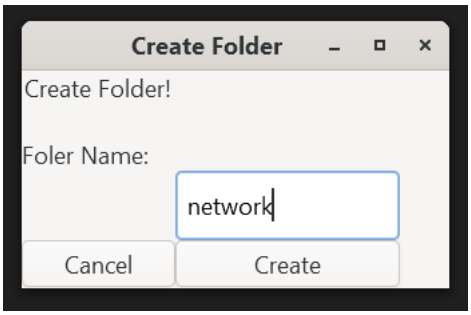
Same as above, user can download a file from server data as in picture



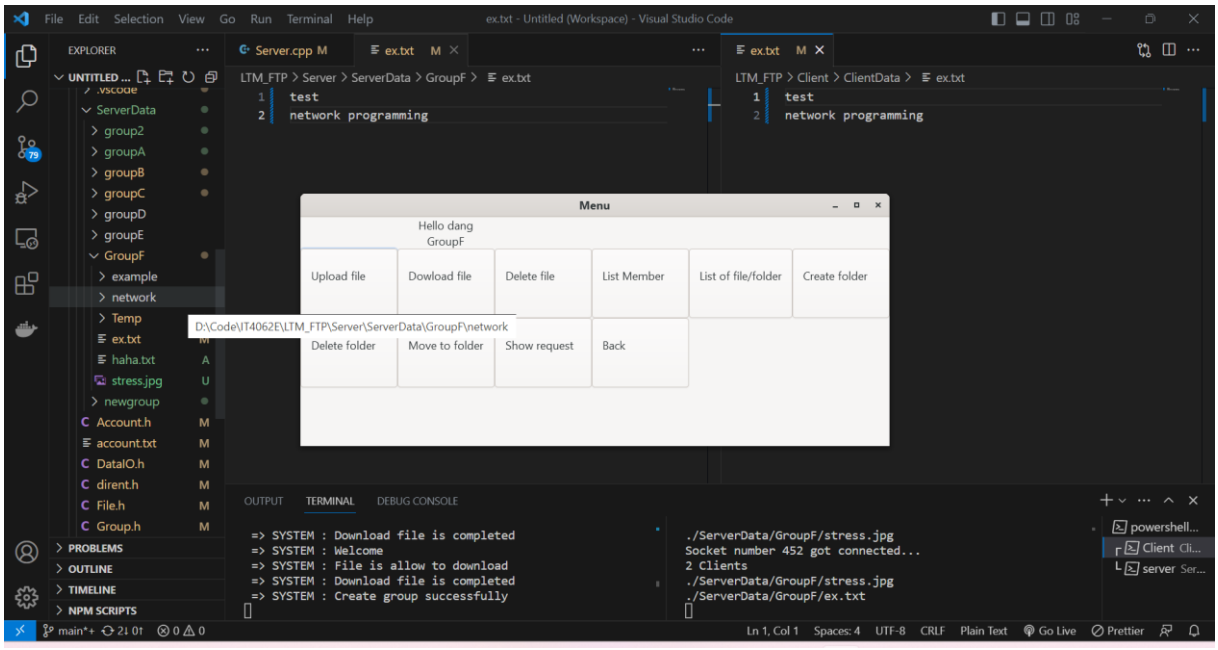
After downloading, now in client data storage we have a new ex.txt file, with the same content as server file



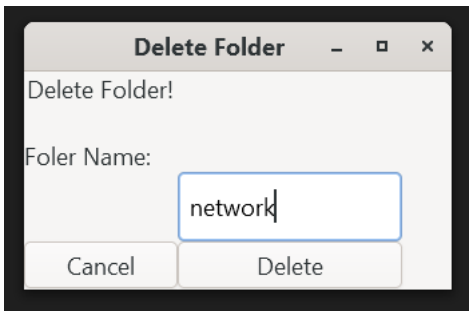
Now I will create a new folder in GroupF named as ‘network’



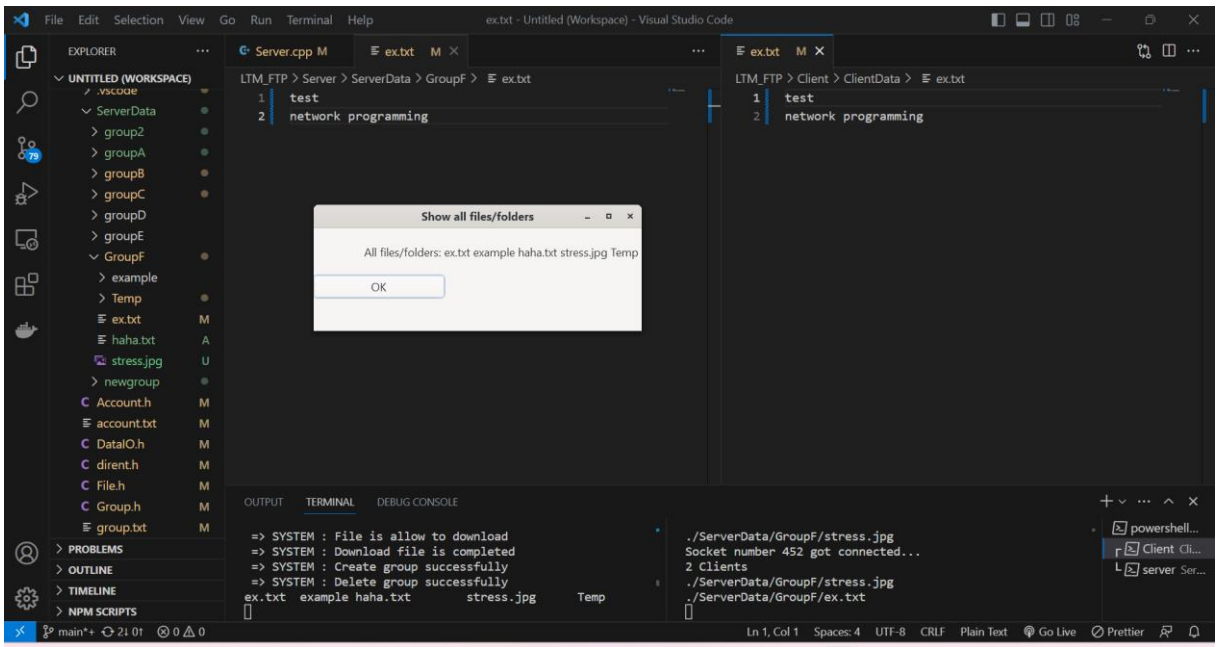
And now we have a folder named ‘network’ in GroupF



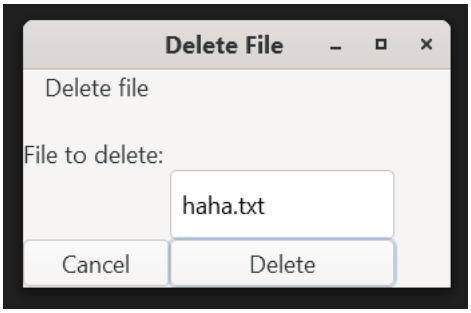
For group leader, we also provide delete folder/file function. User needs to enter folder name and click on Delete button

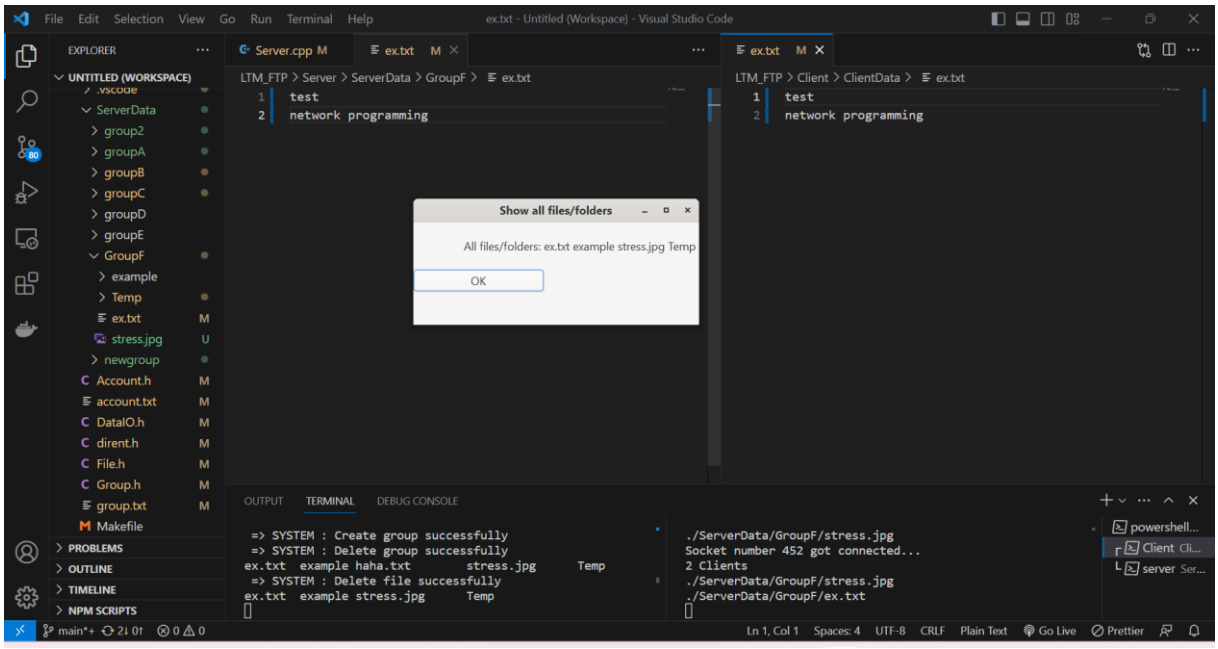


And user can choose ‘list folder’ function to see the result as below



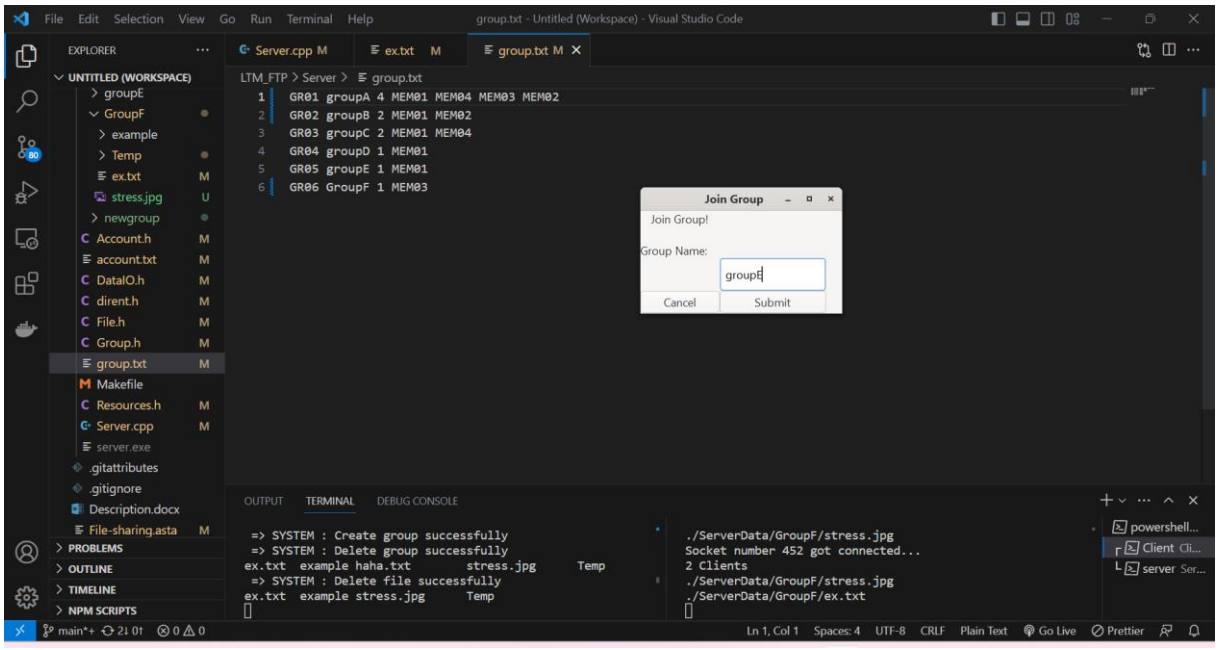
Same as delete file function



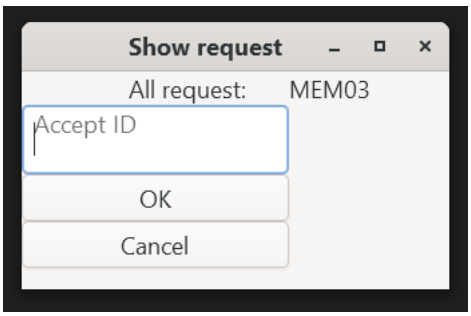


Beside above function, user also can join other group. User clicks on ‘join group’ button from the menu screen and fill in group name

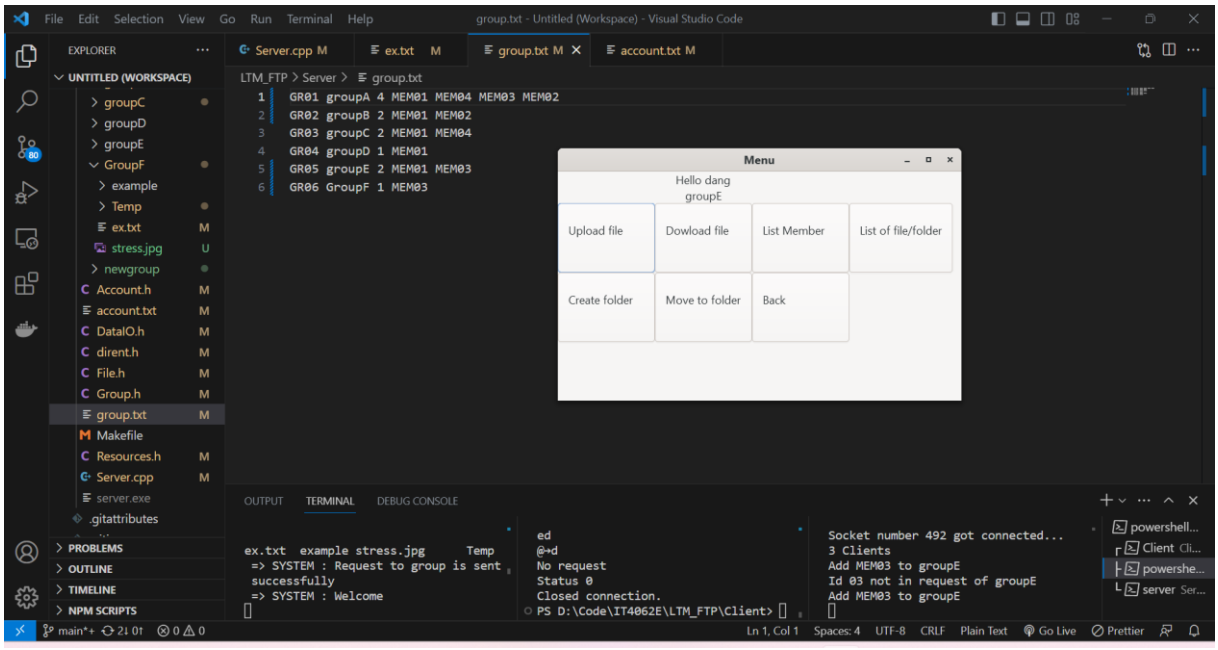
As in demonstation, I login as MEM03 and I want to join groupE



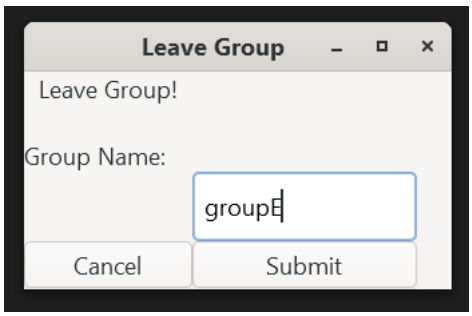
Now I login as groupE leader (MEM01), enter groupE and clicks on ‘show request’ button. The below screen will be displayed.



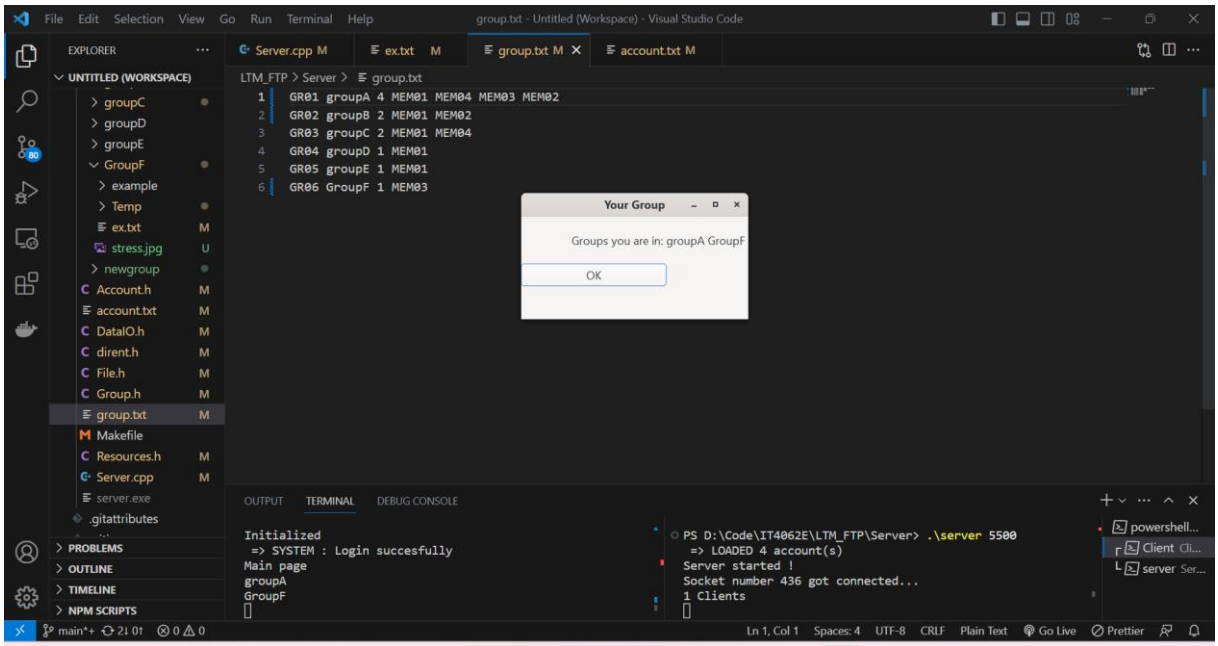
Only group leader can accept join request. Group leader enter the member ID and click on ‘OK’ button to add. The result as below.



Now back to account MEM03, we choose ‘leave group’ function. User enter the group name and submit

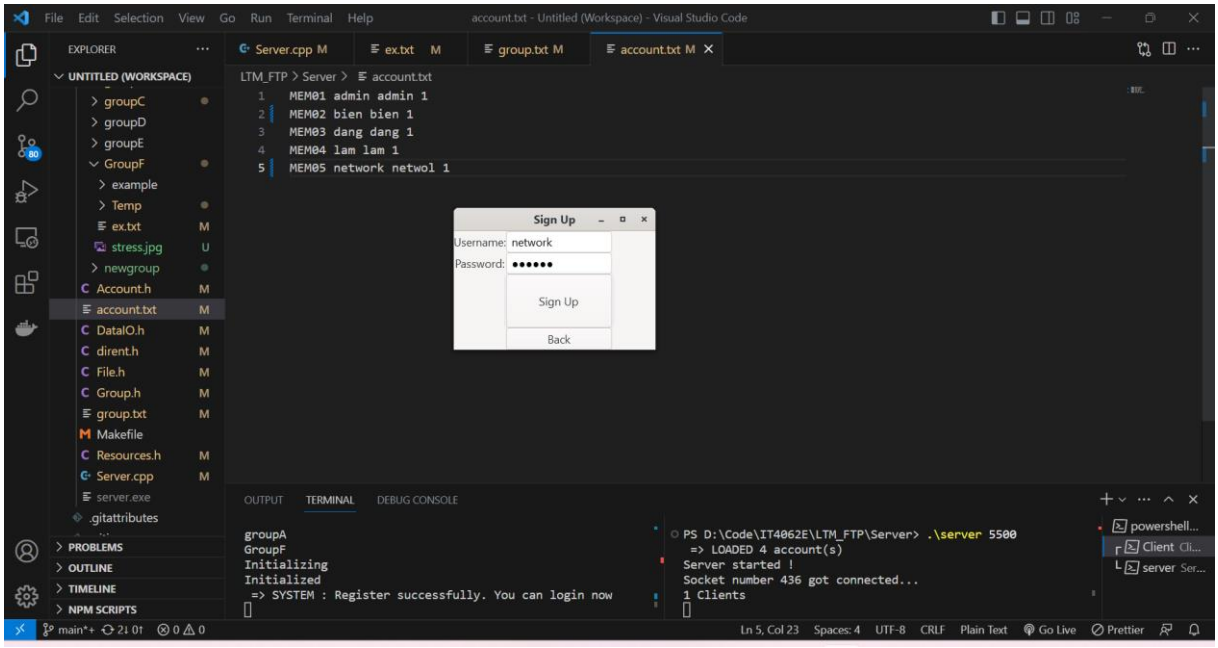


Now user leaves groupE and only int groupA and groupF



For who doesnot have account, we also support ‘sign up’ function. User choose ‘sign up’ and then enter username and password.

If information is valid, you will be added to user data storage



4.2. Testing

No	Test case	Description	Expected result	Actual result	Pass/Fail
1	Login	Login to system	Login successfully	As expected	Pass
2	Signin	Register new account	Signin successfully	As expected	Pass
3	Logout	Logout of system	Logout successfully	As expected	Pass
4	Create group	Create a new group	Create group successfully	As expected	Pass
5	Join group	Join other group	Join group successfullu	As expected	Pass
6	Leave group	Leave joined group	Leave group successfully	As expected	Pass

7	Create folder	Create new folder in a group	Create folder successfully	As expected	Pass
8	Delete folder	Delete folder in a group	Delete folder successfully	As expected	Pass
9	Upload file	Upload file to server storage	Upload file successfully	As expected	Pass
10	Download file	Download file to local storage	Download file successfully	As expected	Pass
11	Delete file	Delete file from server storage	Delete file successfully	As expected	Pass

Table 10: Testing usecase

4.3. Deployment

This application has been tested on local computer with 3 user concurrently login in same time.

Chapter 5. Conclusion

The network application for file sharing that has been developed demonstrates a robust and efficient solution for exchanging files across a network. Through the implementation of this application, several key conclusions can be drawn:

1. Seamless File Sharing: The application allows users to share files effortlessly over the network. The user-friendly interface and straightforward file sharing process make it accessible to a wide range of users, regardless of their technical expertise.

2. Asynchronous I/O for Improved Performance: The use of asynchronous I/O, such as I/O Completion Ports (IOCP), ensures that the application can handle multiple file transfers simultaneously without causing significant delays or resource bottlenecks. This leads to improved performance and scalability.

3. Enhanced Security Measures: The application incorporates robust security measures to protect sensitive data during file transfers. Encryption and authentication mechanisms ensure that only authorized users can access shared files.

4. Efficient Resource Management: By employing a thread pool and optimizing resource usage, the application minimizes overhead and efficiently manages system resources. This allows it to handle a large number of file transfer requests without consuming excessive memory or CPU power.

5. Real-time Progress Updates: Users are provided with real-time progress updates during file transfers, ensuring transparency and enabling them to monitor the status of their uploads and downloads.

6. Scalability and Performance: The application's design is scalable, allowing it to handle an increasing number of users and file transfers without compromising performance or response times.

7. Error Handling and Recovery: The application demonstrates robust error handling and recovery mechanisms. In the event of network disruptions or file transfer failures, users are informed of errors, and the system attempts to recover gracefully.

8. Simplicity and User Intuitiveness: The application's intuitive user interface and streamlined file-sharing process make it an accessible tool for both novice and experienced users.

In conclusion, the network application for file sharing has proven to be a successful implementation, providing a secure, efficient, and user-friendly solution for exchanging files over a network. Its adoption of modern networking techniques and adherence to best practices have resulted in a high-performing and reliable system that meets the needs of users seeking a seamless file-sharing experience.

At this point, the project is also completed. Group 2 would like to thank you for guiding throughout the process of making this project.

THE END