COMP90015 Distributed Systems
Assignment 2: Distributed Shared White Board

ZHAOFENG QIU 1101584
ZHIHAN GUO 1100249
AOQI ZUO 1028089
SHIRU XIE 1077303

October 24, 2019

1 Description

In this project, we design and implement a shared whiteboard system that can be edited simultaneously, using a P2P structure with a centralized index server. The system supports a range of features such as freehand drawing, drawing multiple shapes such as lines, circles, rectangles, and oval with specific colours and thicknesses. Undo and Redo method is also provided to the user. We also implement a "File" menu, which allows the manager to new, open, or save a file in different formats. Besides, the system also provides a chat window so that all the users in the same shared whiteboard can send messages to each other. Chat contents are encrypted when transmitted, which ensures users' privacy. Moreover, we provide a lobby system for our users to create, join, and search for specific whiteboard rooms, which we think is a very user-friendly improvement.

2 Methodology

2.1 System Structure

The whole system is implemented by a P2P with a Centralized Index Server Architecture. Although different users may have different permissions to operate on one whiteboard, they are essentially both servers and users, interacting cooperatively as peers to view and modify a canvas without distinction between clients and servers. Besides, a centralized server saves connection information for each user and each whiteboard room.

2.2 Communication Mechanism

The communication in this project takes place via TCP sockets and RMI. Specifically, the communications between the centralized server and all users are designed in TCP sockets with thread-per-request architecture. In the centralized server, a socket is listening for and accepting requests from clients. For each request, a thread is created to handle the request between a client and the server.

Among users, the connection and chat message exchange between host and guests are implemented by TCP sockets with thread-per-connection. In the host of the whiteboard room, a socket is listening for and accepting connections from visitors. For each connection, a thread is created to handle the connection between a host and a guest. Moreover, the interactive modification to the whiteboard between users takes place via Two-way RMI.

2.3 Data Storage

In the centralized index server, we only record the users' ID and their whiteboards' connection information. When users get the connection information from the centralized server, the connection between them is independent of the central server. The User's paintings can be saved to local files. We provide exports of three types of file formats, including "jpg", "png" and "wb". "wb" is an editable file of our program. Only files with the file extension "wb" can be opened.

2.4 Concurrency

The same user ID can not be registered by any two users. This is implemented by the "synchronized" creating room method. Besides, in each room, the guest can have access to the shared resources of the whiteboard. In other words, guests can draw simultaneously on a canvas, where concurrency can be guaranteed by synchronizing modification to the board by the "synchronized" method for any update operation to the canvas.

2.5 Message Exchange

The chat content exchanged from users is encrypted by a shared secret key to ensure confidentiality. Besides, we use the JSON format to transmit room and users' connection information between the central server and users, which is more practical than passing messages by the string. With RMI, there is no need to explicitly give the transport format of the whiteboard data because it is object-oriented.

2.6 Interface Utilities

In this system, we use the Java2D drawing package. Java 2D is an API for drawing two-dimensional graphics using the Java programming language. Java 2D is a powerful technology. It can be used to create rich user interfaces, games, animations, multimedia applications or various special effects.

3 Design Details

3.1 Centralized Index Server

With a centralized index server, users can establish a connection with others without knowing other users' connection information at the beginning. Once the user can connect to the central server, they can create or access others' Shared whiteboard. The GUI and the classes diagram of it are shown in Figure 1 and Figure 2, respectively. The Sequence diagram for creating rooms and joining rooms are shown in Figure 3 and 4, respectively.

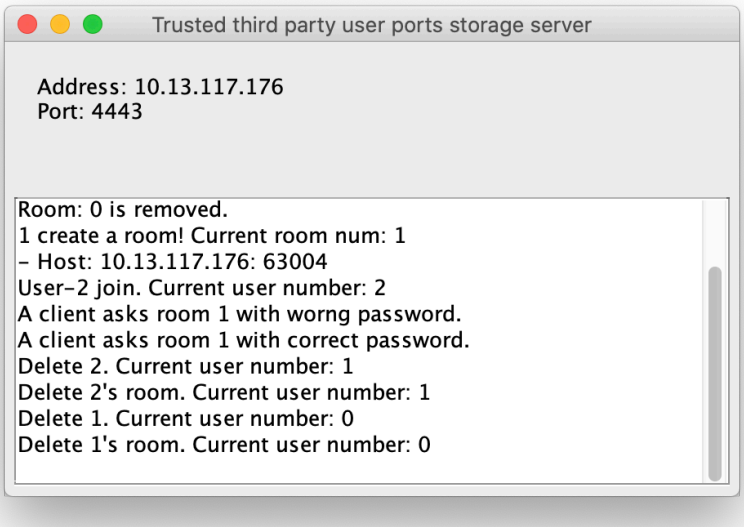

Figure 1: Centralized Server

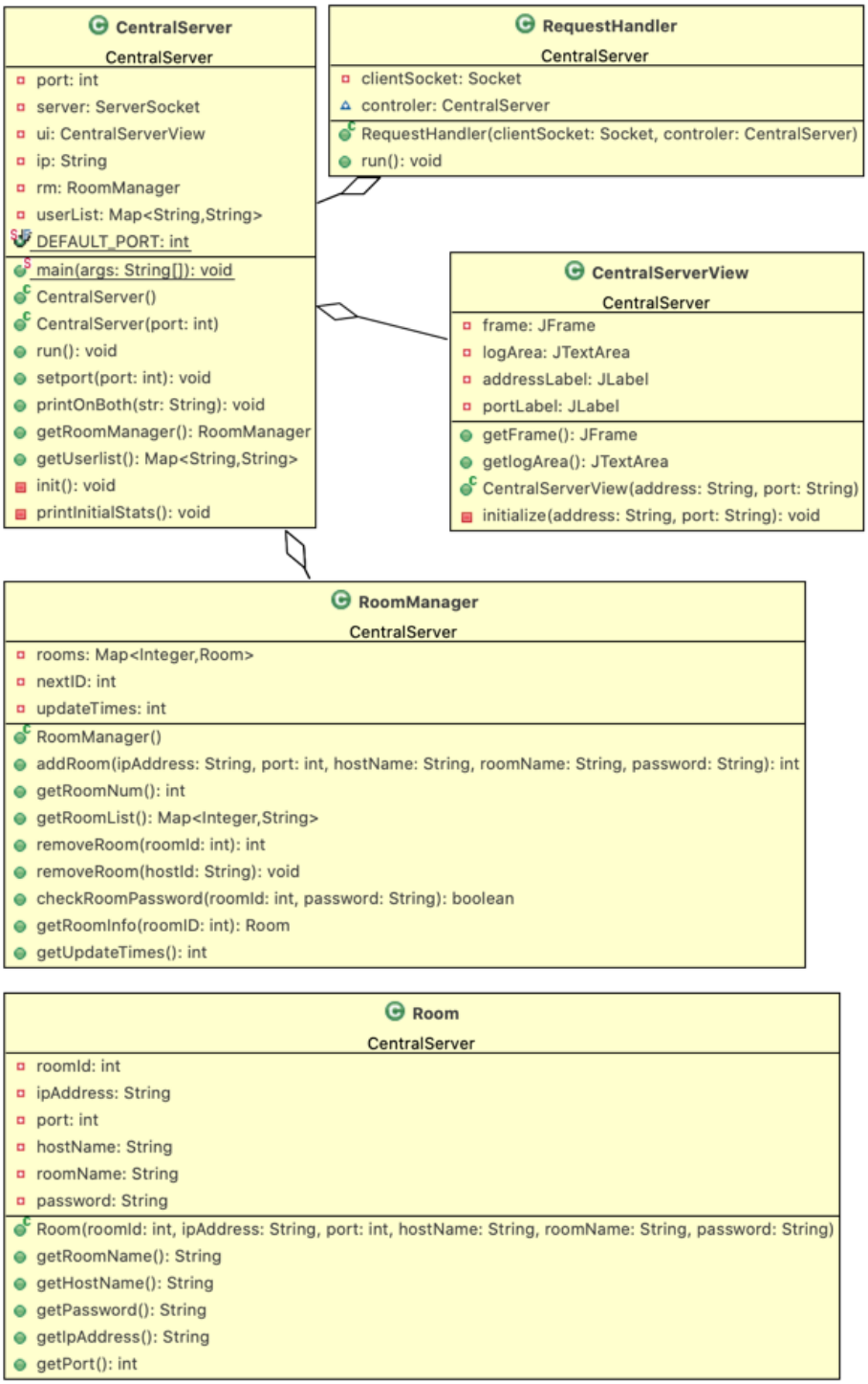

Figure 2: Centralized Server Classes Diagram

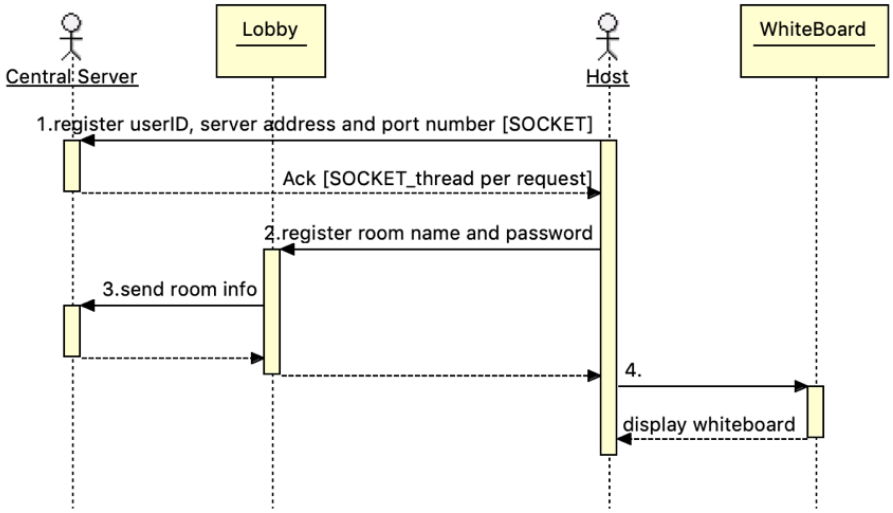

Figure 3: Create Rooms Sequence Diagram

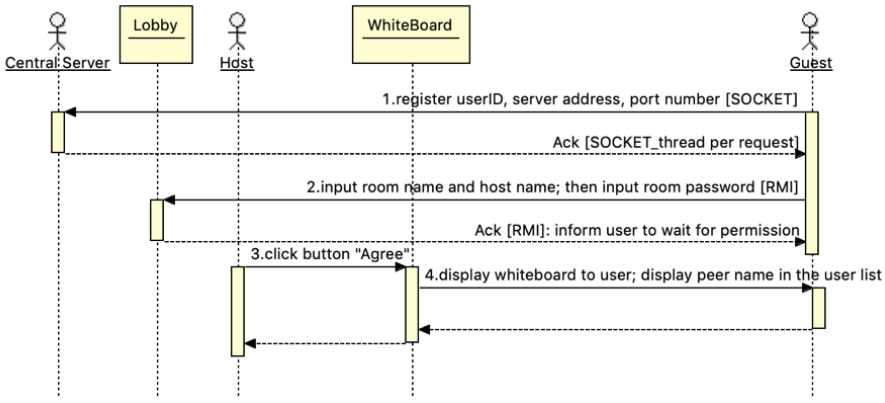

Figure 4: Join Rooms Sequence Diagram

3.2 App

The entire application is divided into three windows, including Sign In Window, Lobby Window and the Shared Whiteboard Window. The app class is mainly used to store global information of the user and switch between the three windows. The classes diagram of it are shown in Figure 5

Figure 5: App Class Diagram

3.2.1 Sign In Window

In this frame, the visitor should input the User ID to enter the shared whiteboard lobby. It is noted that if the user id exists, the visitor can not sign in the window successfully. The GUI and the classes diagram of it are shown in Figure 6 and Figure 7, respectively.

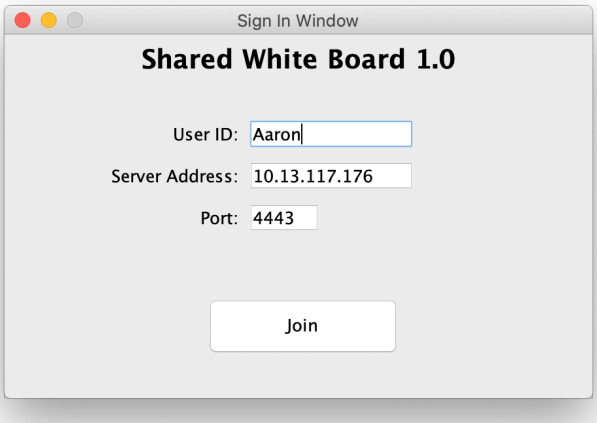

Figure 6: Sign In Window

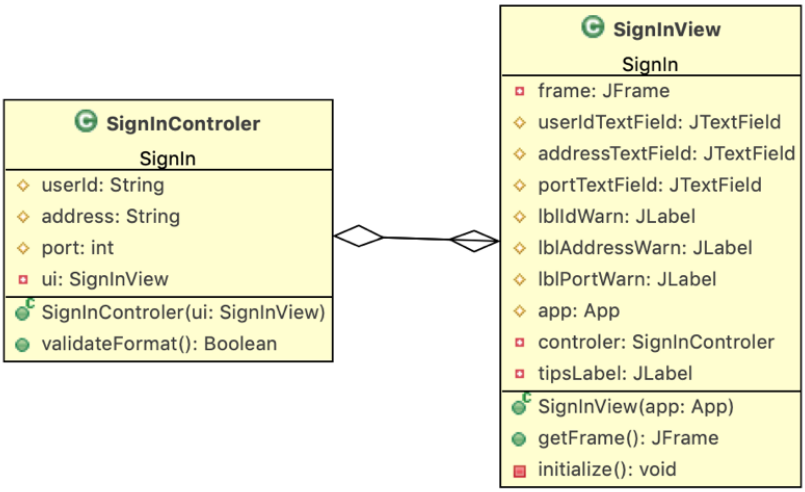

Figure 7: Sign In Class Diagram

3.2.2 Lobby

In the lobby, Users can create rooms themselves or join other people's whiteboard. Besides, to make it easier for users to find a specific room, we provide a room filtering mechanism based on the user's name or room name. The GUI and classes diagram of it are shown in Figure 8 and Figure 9, respectively.

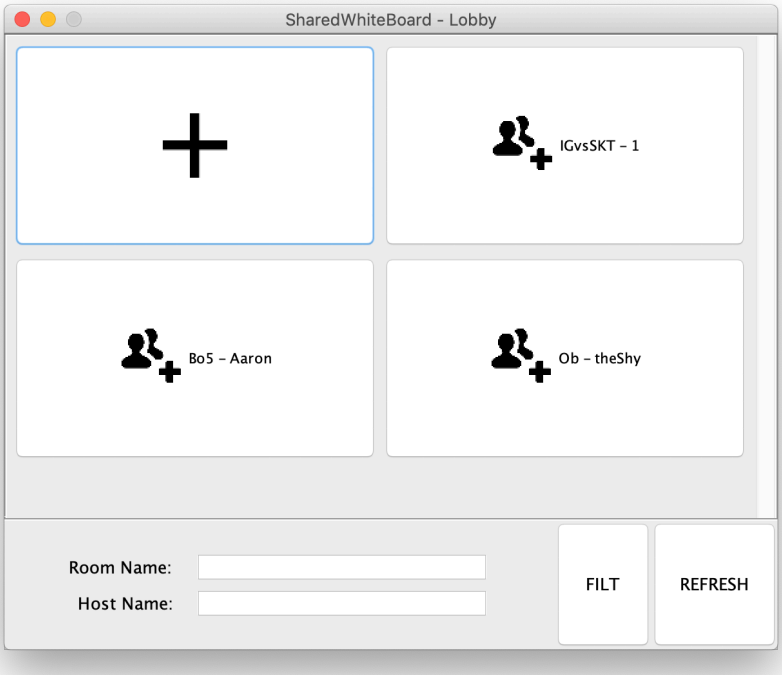

Figure 8: Lobby

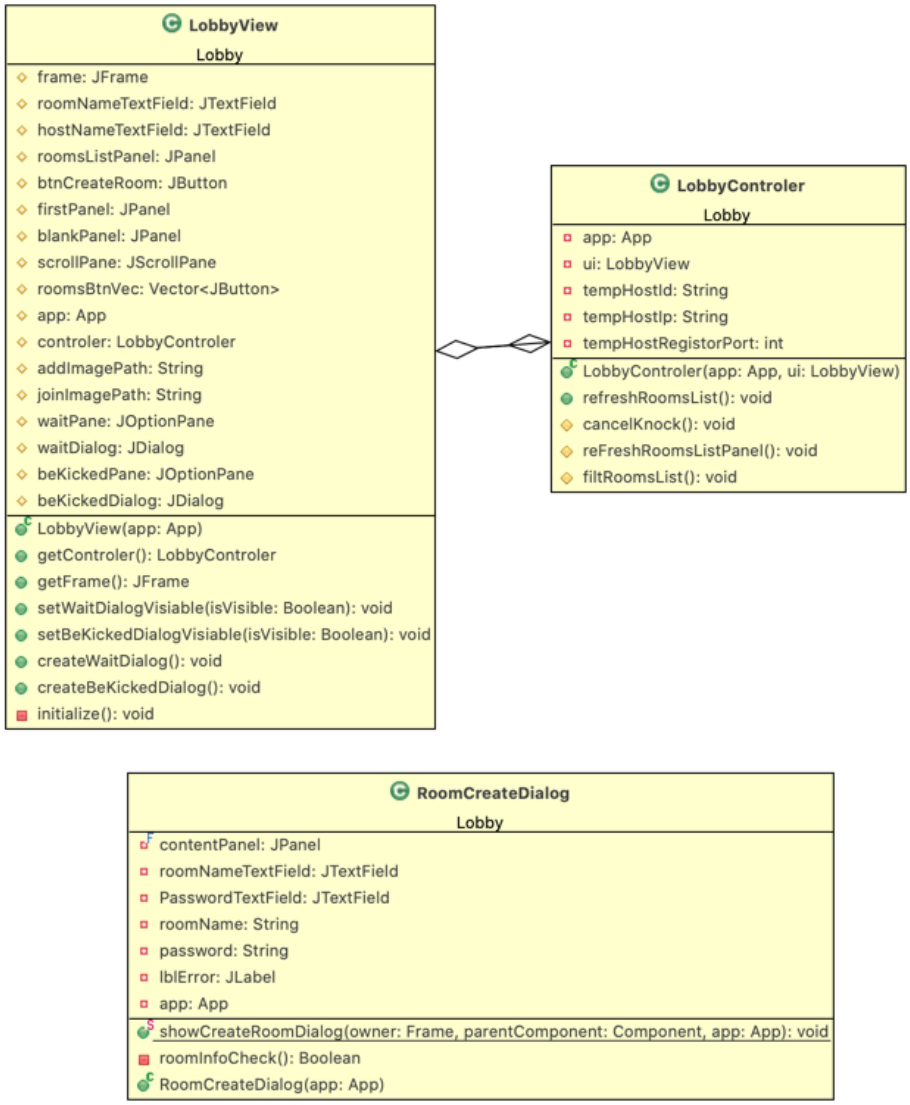

Figure 9: Lobby Class Diagram

3.2.3 Shared Whiteboard

In the whiteboard, users can do a free drawing or draw multiple shapes with specific colours and thicknesses. In addition to the default 16 colours provided by the system, users can choose any colour. The host of the whiteboard can also use the Redo and Undo method by shortcut. The GUI and classes diagram of it are shown in Figure 10 and Figure 11, respectively. We also implement a "File" menu, which allows the manager to new, open, or save a file in different formats. The logic of our file storage system is user-friendly and conforms to standard software storage specifications.

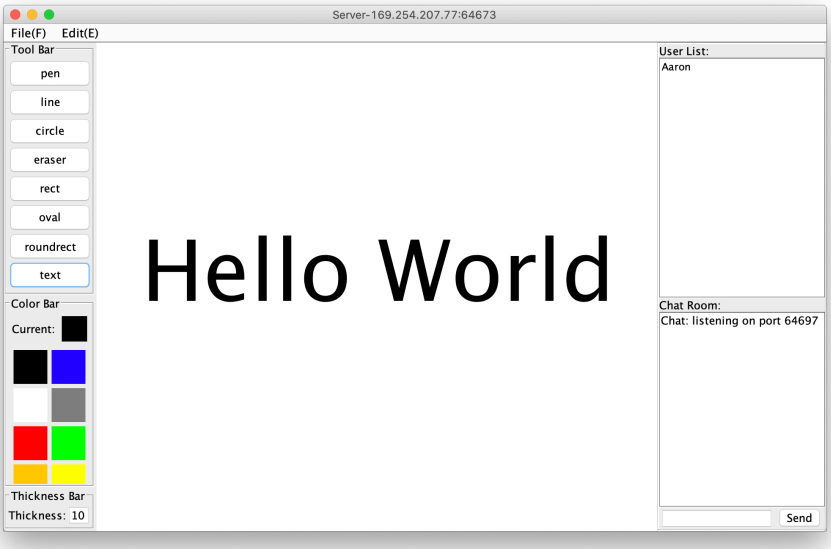

Figure 10: Shared Whiteboard

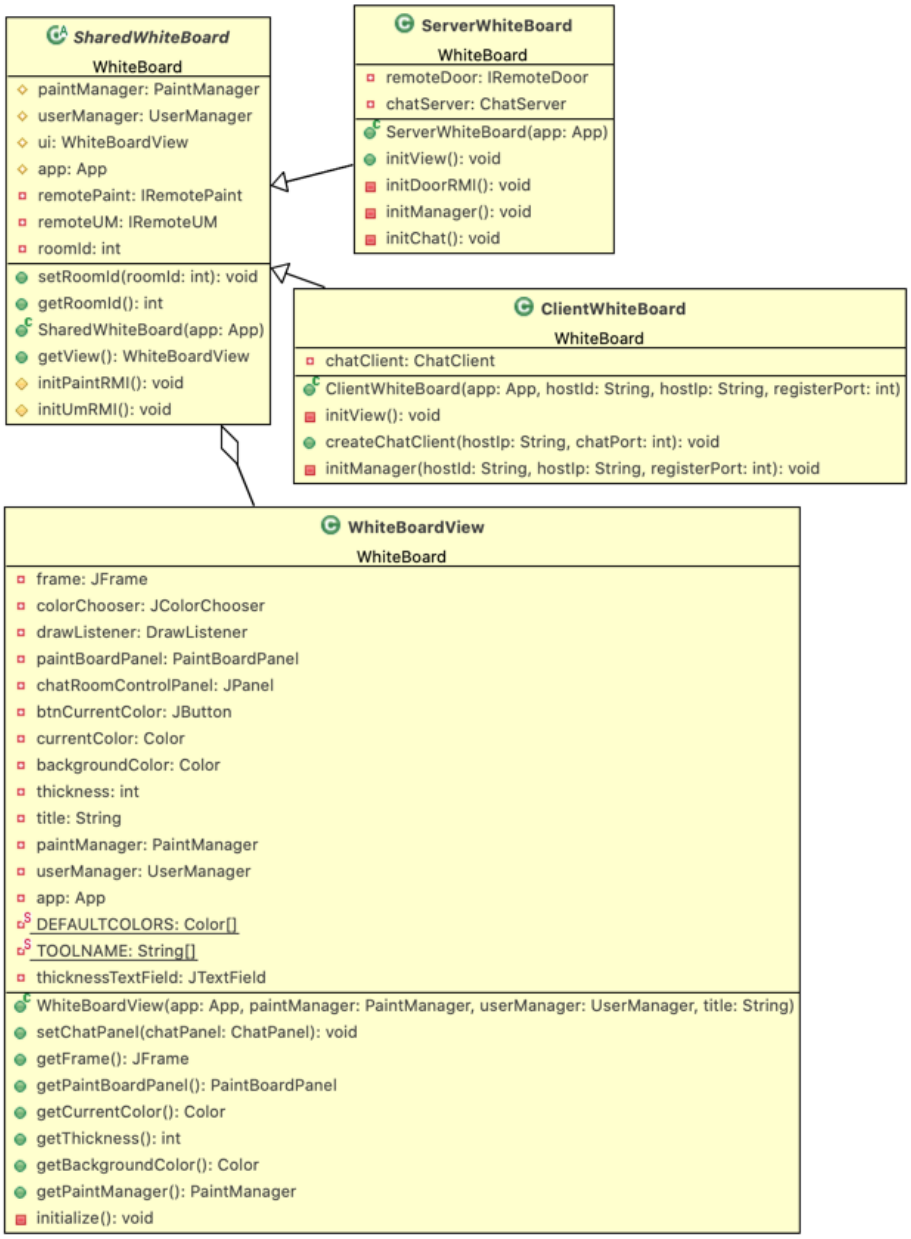

Figure 11: Shared Whiteboard Class Diagram

3.2.4 Room User Management

We offer a complete room user management solution. The classes diagram of the it is shown in Figure 12. The host of a room has absolute control over the whiteboard. When a visitor wants to join a room, first he has to use the correct password, get the connection information from the centralized index server. Then he has to get approval from the host before he can join the room. The host can kick anyone except himself out of the room. Of course, we also provide the user with the operation of returning to the lobby or closing the whiteboard. When the owner closes the drawing board, everyone else in the room would be kicked out. Users who are kicked out will be notified. The sequence diagram for leaving room and being kicked are shown in Figure 13 and Figure 14, respectively.

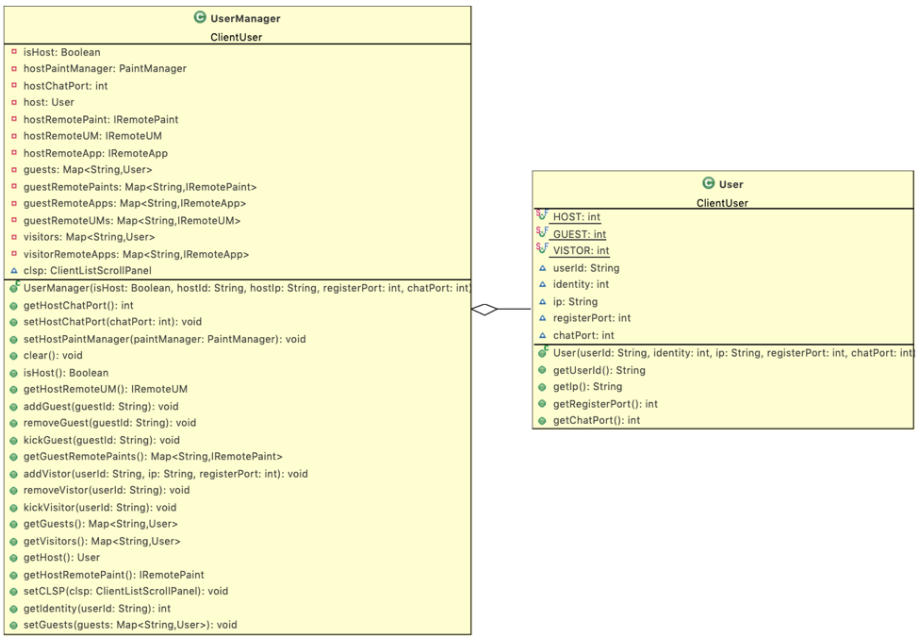

Figure 12: User Management Class Diagram

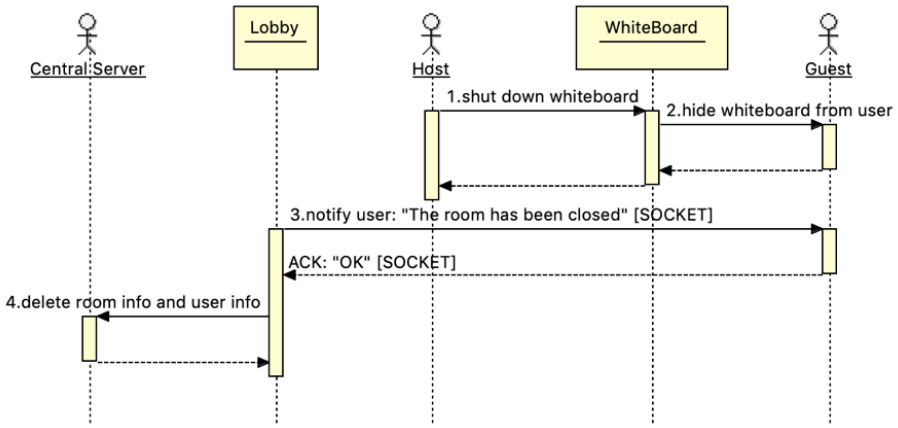

Figure 13: Leave Room Sequence Diagram

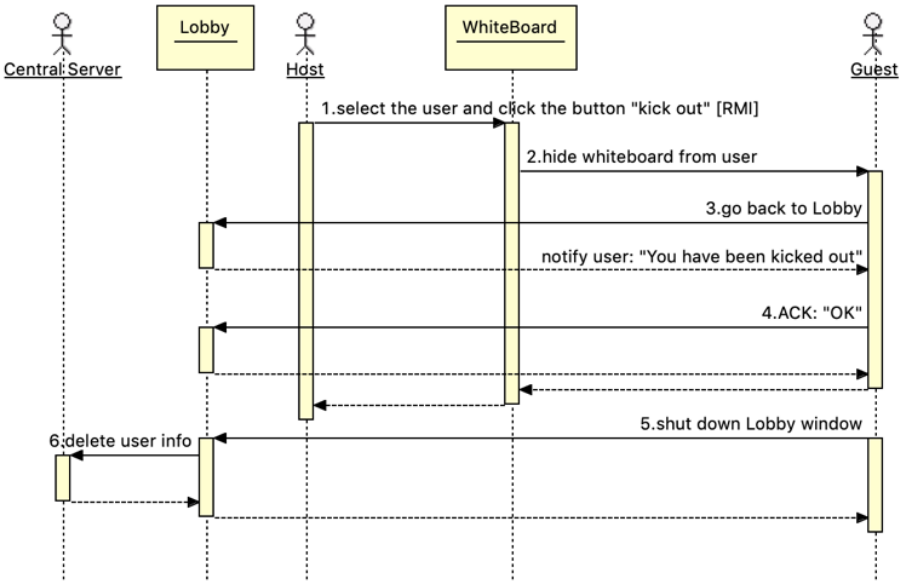

Figure 14: Be kicked Sequence Diagram

3.2.5 Chat System

We provide a chat system for users to chat in the same whiteboard. Users can chat while drawing because the chat system runs in a separate thread. The classes diagram of the it is shown in Figure 15.

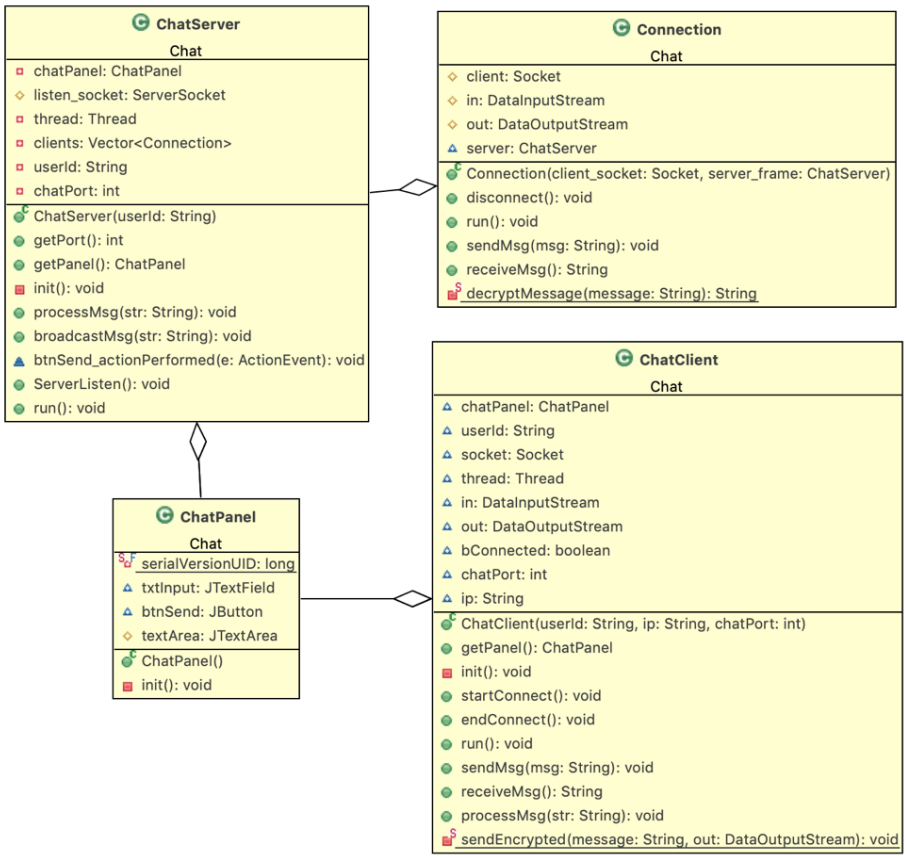

Figure 15: Chat System Class Diagram

3.2.6 Painting System

In our application, each drawing of the user would be considered as adding a shape to the whiteboard. Different graphics are inherited from the ”Myshape” parent class. Due to the variety of graphics, only the circle is used to represent the class diagram. The the classes diagram and sequence diagram of it are shown in Figure 16 and Figure 17

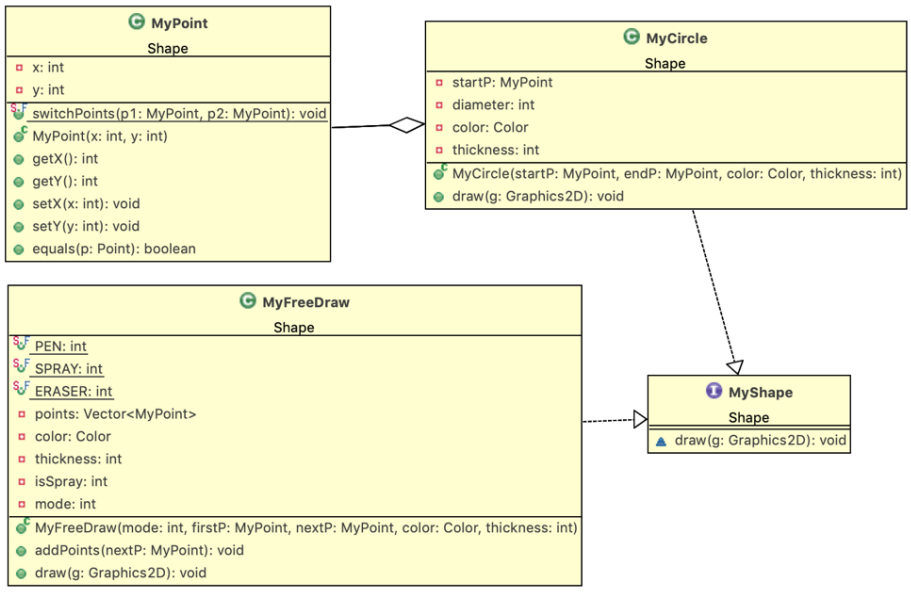

Figure 16: Shape Class Diagram

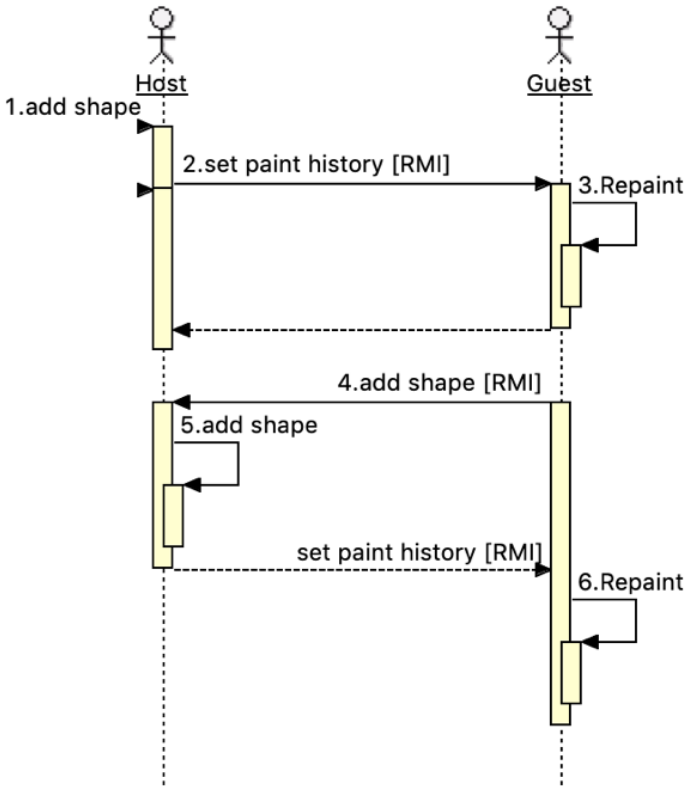

Figure 17: Add Shape Sequence Diagram

4 Failure Handling and Recovery

This system can handle various kinds of errors, including the invalid port number, lack of parameters, repeated user id, the incorrect password, etc. For each type of errors, the system has provided proper notification to help users understand how to deal with the exception. The problems and their solutions of the Centralized Index Server and the application are listed in Table 1 and Table 2, respectively.

Description	Solution
the invalid port number	invalid Port Number: Port number should be between 1024 and 49151
Delete not existed user	User id does not exist! can't delete.

Table 1: Centralized Indexed Server Errors and Solution

Description	Solution
the invalid port number	invalid Port Number: Port number should be between 1024 and 49151.
No centralized index server	Send feedback to client:can not connect to the server.
Incorrect password format	Password Format:\w1,8.
Repeated user id	User name exists ! change one!
Incorrect room name format	Room Name Format:\w1,8.
Id pattern	Please check the id format carefully.
Address pattern	Please check the address format carefully.
RMI Exception	Remove the RMI object and remove the guest.

Table 2: App Errors and Solution

5 Critical Analysis

5.1 Architecture Analysis

As mentioned in Section 1 and 2, our application is implemented using P2P architecture with Centralized Index Server. Also, some of the system in our application is based on TCP sockets and RMI for different communication needs.

TCP socket is greatly reliable, which guarantees the order of the arrivals of the request and the order of the response. We apply TCP sockets with thread-per-request architecture for communications between the centralized index server and all users, and thread-per-connection for chat contents exchange between users of a whiteboard room. The reason we use this different architecture is that the number of requests (or say communication) between users is much more than requests between centralized server and hosts.

Due to the small amount of information exchange between the user program and the centralized index server, the centralized index server does not need to stay connected to the users for a long time. To lower such a thread-management overhead, we choose to use thread-per-request for it. For the exchange of chat messages between users, because of the frequent exchange of information, we think thread-per-connection is much effective and practical.

In order to allow multiple users to draw on a whiteboard at the same time, we use two-way RMI. With RMI, there is no need to explicitly give the transport format of the whiteboard data because it is object-oriented. The reason for using the two-way RMI is that the one-way RMI cannot meet the need for broadcasting. In the two-way RMI, the guests can modify the whiteboard through the RMI provided by the host, and the host can also call the RMI update canvas of the guests. In this way, we implement broadcasting.

5.2 Comprehensive Analysis

Apart from architecture, a well-designed fail module displayed in Section 4 makes our system more reliable. The encryption of chat contents also ensures users' privacy. A user-friendly interface also has been implemented. The creation of whiteboard room and modification to the whiteboard are synchronized. Therefore the same kind of operation by different users would not conflict. Besides, our P2P architecture also guarantees scalability. Since our centralized server doesn't have to deal with the user drawing operations, numerous whiteboard rooms can run at the same time.

However, there are some limitations with this system. Since the thread-per-connection model is implemented in this project, the scalability is limited. As the number of guests extending, the burden of the host of the room increases. There also exists the time when some clients may be delayed while a thread for a connection has several outstanding requests but another thread has no work to perform.

Besides, a problem with P2P is that the customer's experience depends largely on the performance of the host computer and the quality of the connection to the host's network. If there are many people operating in the same room, the burden on the host will be great. Moreover, compared to TCP socket, RMI needs to occupy more network overhead (protocol overhead).

Moreover, due to the current implementation still has some short part, after the canvas is updated, the host needs to call the clients' RMI one by one to update the client's canvas, and this operation is completed on the same thread, so if a user network is abnormal, all the other users in the same room would be affected.

6 Creativity

1. *Centralized index server and Lobby* The design of the Centralized index server and lobby makes our application more user-friendly, intuitive and

effective. Once users can connect to the central server, they can create or access others' Shared whiteboard without explicitly knowing the connection information of other users. The detailed introduction about it is in Section 3.2.2

2. *Encryption* The chatting contents are encrypted by a shared secret key, which can ensure chat messages' confidentiality.

3. *Strokes Preview* While a user is drawing, for example, a circle, the image of a circle will display immediately, and the preview image will change as the user moves the mouse. In this way, users can view the precise location and size of the circle and adjust it easily. Moreover, we redefine the class of free drawing and take the user's complete stroke as the object of transmission. This not only reduces the pressure of transmission, but also makes the painting experience smoother, and also allows us to implement the operation of Redo and Undo.

4. *Synchronizing a Stroke after Releasing the Mouse* Only when the user releases the mouse, the stroke or shape would be added to the whiteboard. Meanwhile, the corresponding update information will be transmitted to the servers. Especially when the users are drawing freely, this vastly eases the pressure to servers since not each pixel but each stroke is treated as an object to synchronize to the server end.

5. *Well Design of the File Storage System* The logic of our file storage system is user-friendly and conforms to standard software storage specifications. The User's paintings can be saved to local files. We provide exports of three types of file formats, including "jpg", "png" and "wb". "wb" is an editable file of our program. Only files with the file extension "wb" can be opened. The 'Save' function is also well designed. As the host chooses a local path to 'Save' a newly painted file or 'Open' an existing file, the path will be recorded for the next time to 'Save'. The newly saved image file will cover the previous one directly. It is especially convenient to back up the picture anytime.

6. *The Concept of Three Roles* There are three roles of users in this whiteboard system, including a host, guests and visitors. The role of required Manager is replaced by the Host, with the ability to manage other users named Guest. Besides, another kind of user who wants to enter the exact room to share one whiteboard is categorized as a visitor. When a visitor joins, he would be considered as a guest. This setting improves the efficiency of management.

7 Conclusion

This system successfully implements a multiple-user shared whiteboard in P2P with Centralized Index Server architecture. It allows users to create whiteboards to complete collaborative painting task with instant messaging function. It also allows the host of the whiteboard to open or save the painting, and manage other peer users. The mixture communication mechanism, such as TCP sockets with thread-per-connection, TCP sockets with thread-per-request and Two-way

RMI has been applied. A well-designed failure model makes this system more reliable. Besides, scalability and security have been guaranteed by different mechanisms. Apart from providing basic functions, this project implements all creativity demonstrated in Section 6. All of the effort above results in a robust and effective distributed whiteboard application.

8 Contribution

Team contribution are shown in Table 3.

Name & Std. No.	Contribution	Overall Contribution
Zhaofeng Qiu (1101584)	1.The implementation of connection between centralized server and users; 2.The implementation of painting interactively between users in RMI; 3.The implementation of UI for lobby, connection and failure module; 4.The process of detail and integration of all work;	25%
Aoqi Zuo (1028089)	1.The implementation of connection between hosts and guests in socket architecture; 2.Transmission and encryption of chat messages; 3.The detailed implementation of UI for chat board; 4.Report writing;	25%
Zhihan Guo (1100249)	1.The design and implementation of UI for drawing board; 2.The implementation of basic painting logic for single users; 3.Report writing;	25%
Shiru Xie (1077303)	1.The implementation of File manipulation for hosts; 2.The design of picture format; 3.The detailed implementation of UI for File Bar; 4.Report writing;	25%

Table 3: Contribution Table