COMP90015 Distributed Systems Assignment 2: Distributed Shared White Board

HoKwong HO

1020934

The University of Melbourne

May 5th 2023

1. System Architecture

The system architecture of the white board system is similar to assignment 1. The project is separated into two components to fulfill the requirement of the assignment.

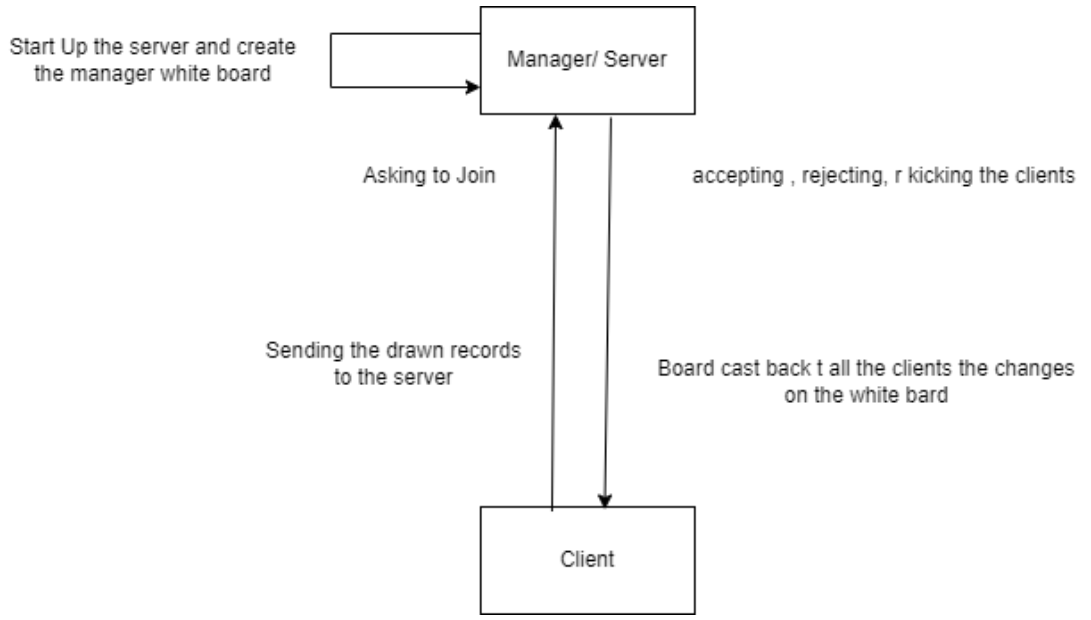

Figure 1: Architecture Diagram

1.1 Manager

The files in manager package can be separate into 3 key

functionalities, which is setting up connections and communicate with the connected clients, setting up UI for the system and handle all the button functionalities like save, chat, kick, etc., and handling mouse movement on the white board and drawing functionalities.

1.2 Client

The client package is similar to the manager package; however, each client will only allow to communicate with the manager (main server) and the communication between clients is not allowed. Als, client will not have the ability do save, open, and create to the current drawing.

The server is implemented with the thread-per-connection architecture, in which, a thread will be created for every client.

1.3 Advantages

Due to the reason that there will not have a lot of clients connecting to the server, using thread-per-connection will only need to create a small amount of thread, which saves the cot of creating the threads. Also compared to thread-per-request, as we know we will send many requests when doing a single drawing action. Therefore, the cost of terminating thread will be huge if we use connection-per-request.

1.4 Disadvantages

As mentioned previously, there will be a lot of requests send for every drawing action, which if multiple users are drawing at the same time, the server will take a lot of time to respond to all the requests.

A synchronization problem may occur between each client and each request pushed by a single client as multiple requests are pushed at the same time; thus, more synchronization handling is required.

2. Communication Protocols and Message Formats

2.1 Communication Protocols

Information transfer between the server and client will be done with reliable communication using the TCP – Socket connection protocol. I have picked TCP Socket protocol for two reasons. Firstly, TCP – Socket connection protocol fits thi project the best. This project only required the communication between a main server, which is the manager, and the client. Which, this protocol allows the client and the manager to connect directly. The middleware layer use in RMI

will not exist, which will decrease the overhead in data transfer and improve performance. Secondly, it is easier for me to implement TCP -Socket protocol since I have already done it in assignment 1.

2.2 Message Formats

In this system, I decide to transfer Strings using Data Transfer Streams. I choose to use String I because firstly, due to the reason I am using thread-per-connection, I have to minimize the size of data that is transmitting s that even if lots of users are using the white board at the same time, the server can still handle the data flow. Hence transferring String is definitely a good choice than transferring images. Secondly, String transmission is more reliable than image transferring. In a up-to-time system, losing information is a common, while serious problem. Using String as the message transferring format will hugely decrease the possibility of data lost. Hence makes the system more reliable.

3. Design Diagrams

3.1 Class Diagram

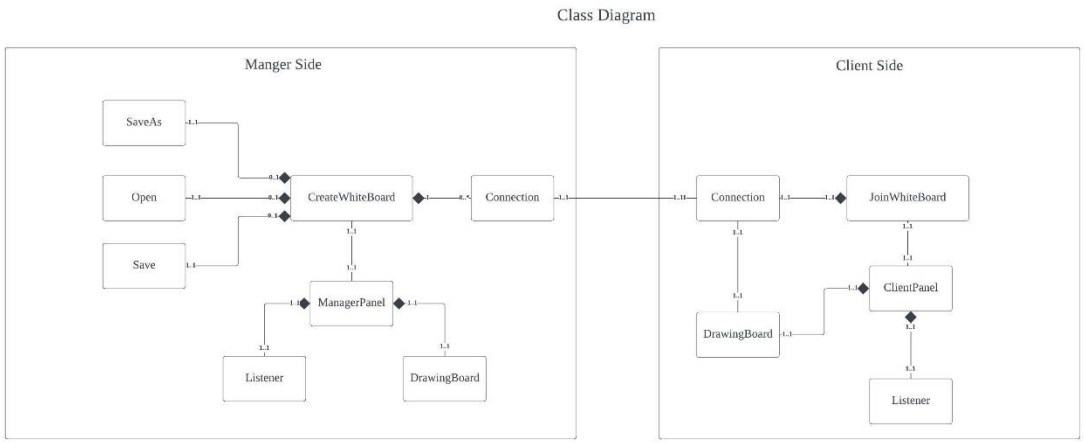

Figure 2: Class Diagram

3.2 Interaction Diagram

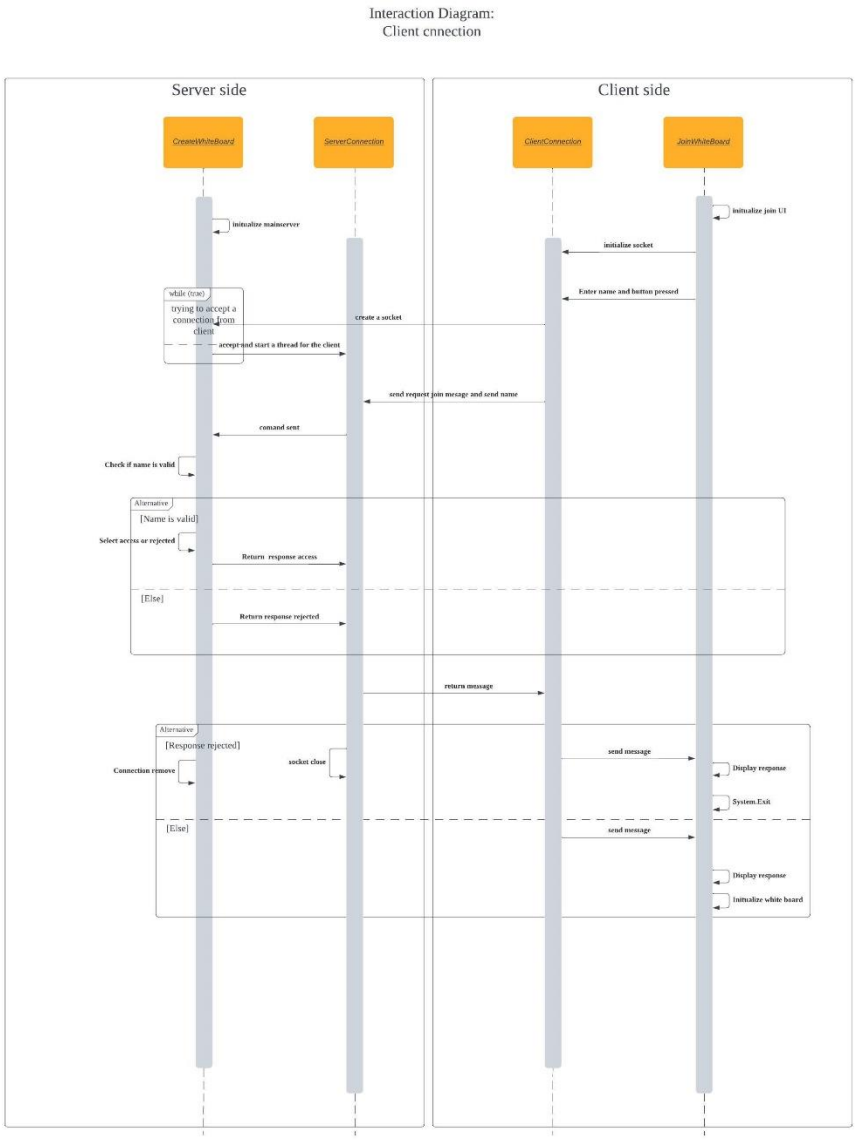

Figure 3: Interaction Diagram: client drawing

This interaction Diagram shows what happened when a client is created. In the client side, a GUI will first be generated, then a connection will be set up between the server and the client. Once the Join button is pressed on the client side, a “request join” message with the client’s name will be send to the server side. The name of the client’s name will be first checked to see if there is a duplicate name. If not, a window will pop up in the server side to let the manager choose can the client join. If the manager chooses yes, a “access” message will be sent back, or a “duplicated name” or a “rejected” message will be sent. Once the client gets the message, it will do action, create white

board UI or close base on the feedback given.

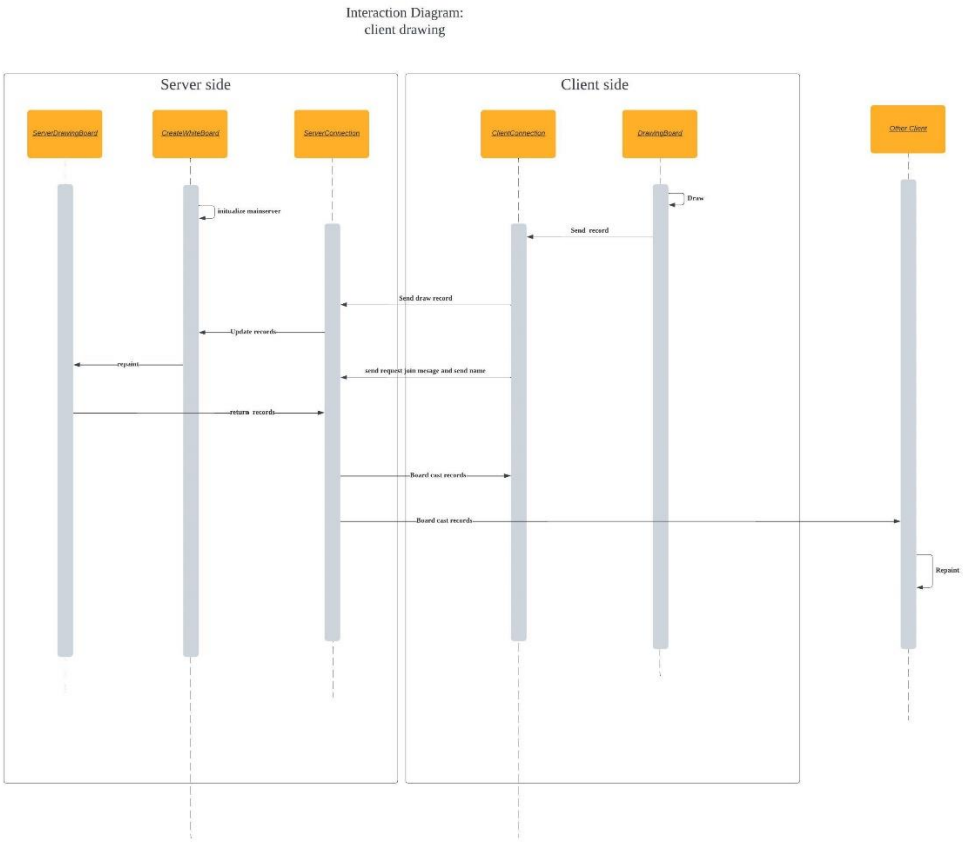

Figure 4: Interaction Diagram: client drawing

This interaction Diagram shows what will happen if a client draws anything on the board. First the drawing record will be sent to the server with a “draw” request. Once the server gets the request, the received record will be added to the record list and the server will paint additional picture base on the added record. Then the server will board cast the record to all the connected client. Once clients get the new request, they will all update their white board.

4. Implementation Details

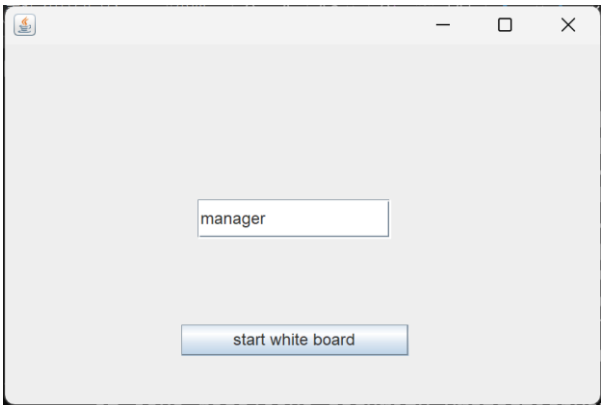

Figure 4: Manager starting window

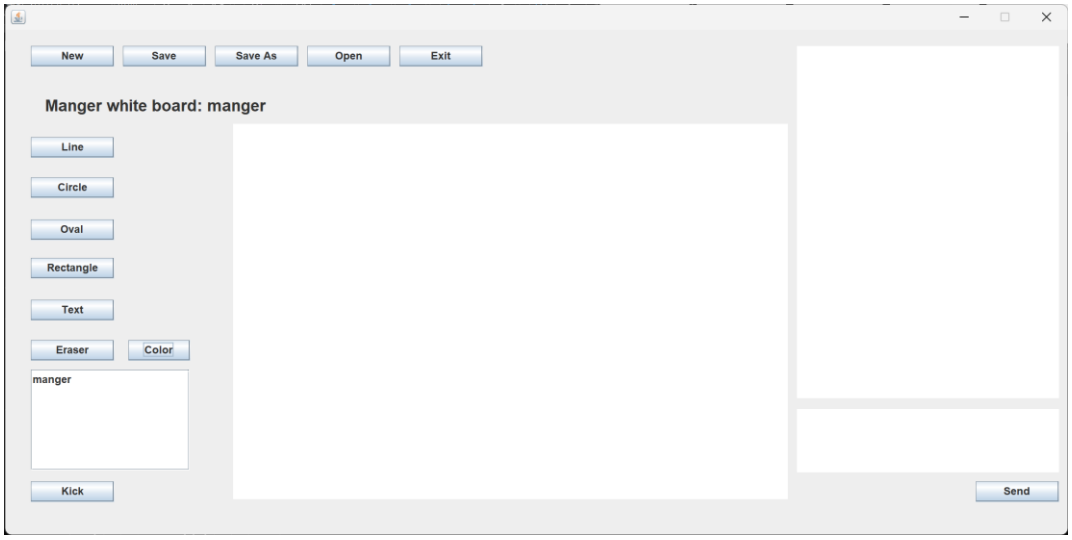

Figure 5: Manager white board

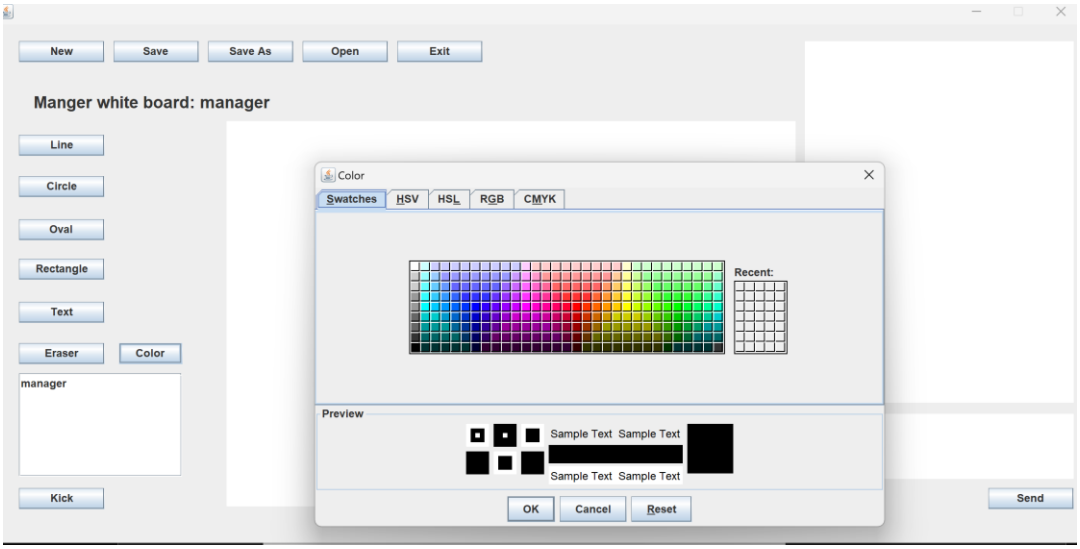

Figure 6: Color Panel

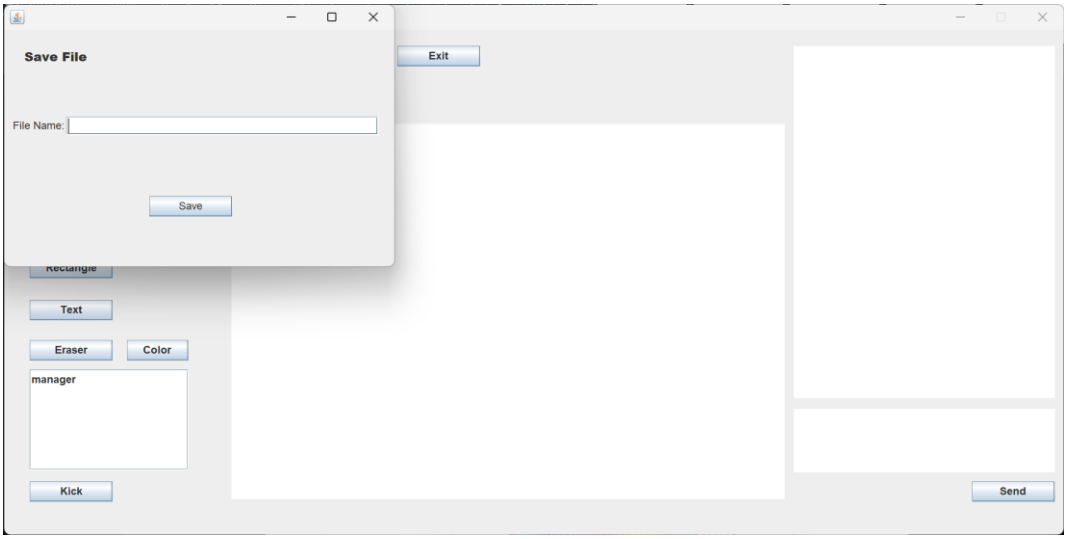

Figure 7: Manager save window

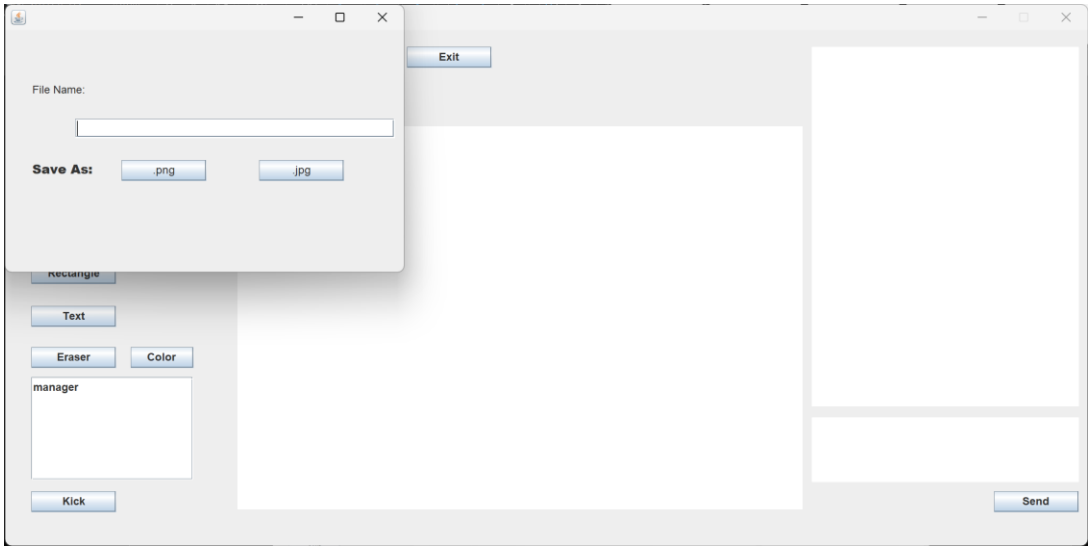

Figure 8: Manager save as window

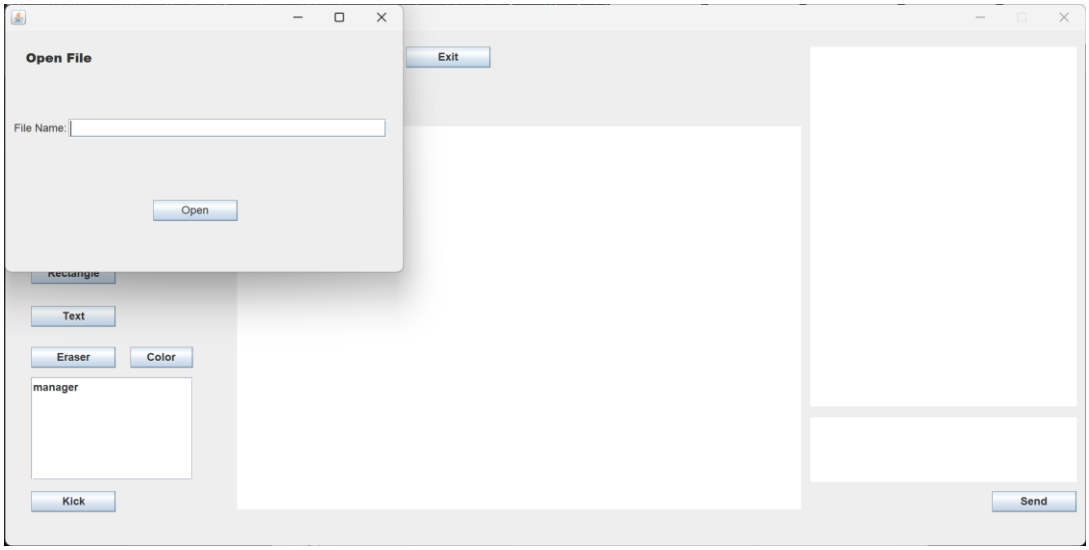

Figure 9: Manager open window

Figure 10: User starting window

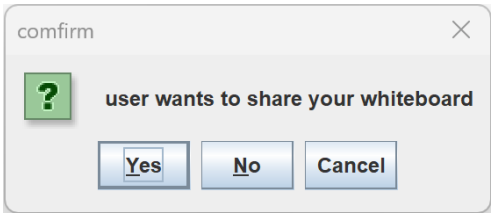

Figure 11: Manager “agree sharing” window

Figure 12: Client white board

5. New Innovations

- I added an eraser button so that people can erase what they have drawn on the white board.

- When a client leaves, all the connected users will be announced with a pop window.
- When a user gets kicked, he/she will get announced with a pop window.
- When a user is rejected to join, a pop window will announce the reason (duplicated name or rejected by the manager)