THE UNIVERSITY OF

MELBOURNE

COMP90015 Distributed System

Assignment 2

Share Whiteboard Report

Joseph Liu
1101805

25th May 2023

A. *Problem Contexts*

The problem context for this assignment is to design and implement a shared whiteboard server using one of the distributed technologies. The shared whiteboard is a multiuser system with a supported GUI. Every user could draw and see the real-time actions of other users. The basic features include drawing lines, circles, ovals, rectangles, text input and others. The whiteboard must be created by a manager so that other clients can join the shared whiteboard.

B. *System Architecture*

In this project, Remote Method Invocation (RMI) has been selected as the communication mechanism. There are two remote objects involved: the first one is IRemoteWhiteBoard, which is bound by the Registry created by the server, and the second one is called IRemoteWhiteBoardClient, which is created when a client establishes a connection with the server.

1. RMI and communication

For the IRemoteWhiteBoard class, it mainly includes methods for broadcasting actions and holding lists that store necessary data. The broadcasting method, for example, triggers the related broadcast function in RemoteWhiteBoardServant when a client draws on the canvas, broadcasting the drawing action to other clients so that they can update their canvas in real-time.

For the IRemoteWhiteBoardClient class, its main job is to store each client's information, as well as their whiteboard, and to take actions based on actions from other clients. For example, when a client draws on their canvas, the RemoteWhiteBoardClientServant class could trigger a function to update their canvas after broadcasting by the RemoteWhiteBoardServant.

2. Message Storage

There are primarily four types of messages that need to be stored: the clients, their usernames, the shapes they have drawn, and the text messages in the chatbox. As mentioned in 1., all this data is held within the IRemoteWhiteBoard Server, organized and stored in ArrayList object.

C. *Class Design*

This project consists of a total of twelve classes, which include one server class, two client classes, two RMI classes, two RMI servant classes, one Enumerate class for shape types, one class for storing shape attributes, and three other GUI classes. Refer to *Appendix A Figure 1* for the UML diagram.

1. Server Class and RemoteWhiteBoardServant

The server class primarily initializes and registers an IRemoteWhiteBoard object on

the RMI registry. The RemoteWhiteBoardServant is responsible for communicating with RemoteWhiteBoardClientServant, enabling each client to perform actions on their respective whiteboards.

2. ***CreateWhiteboardClient and JoinWhiteboardClient***

   There are two types of clients: the first client connecting to the server should be the manager, who uses the CreateWhiteboardClient to establish a connection. Other clients should join the whiteboard using the JoinWhiteboardClient. Both classes have similar content. They both look up the IRemoteWhiteBoard object through the RMI registry and initialize an IRemoteWhiteBoardClient to store their data, such as the username and whiteboard. Additionally, they create a whiteboard object to implement the whiteboard GUI. However, for the JoinWhiteboardClient, they need to gain approval from the CreateWhiteboardClient (acting as the manager) before they can join the whiteboard created by the manager. The manager client also has certain privileges on the whiteboard.

3. ***RemoteWhiteBoardClientServant***

   This class stores the client's information, including the username and the whiteboard object. Its main purpose is to facilitate communication with the RemoteWhiteBoardServant and the client's whiteboard. When receiving a message from the server, this class triggers the relevant function to initiate the corresponding action on the client's whiteboard. This flow ensures synchronization between the clients' whiteboards.

4. ***ShapeType Enumerate***

   The Enumerate class holds all the different types of shape tools available on the whiteboard GUI, including LINE, RECTANGLE, OVAL, CIRCLE, PENCIL, and TEXT.

5. ***WhiteboardShape***

   This class stores the details of each shape drawn by users. The details include the shape type, color, stroke (or font in the case of TEXT), text input (for TEXT), and the location of each shape.

6. ***GUI related Class***

   The Whiteboard class serves as the main GUI representing the users' whiteboard. The whiteboard frame consists of five main parts. On the left-hand side, there are buttons for different actions that can be chosen by clients. These actions include painting tools, defined in the ShapeType Enumerate, and adjustable attributes such as color and stroke for the drawn shapes. In the middle part, there is a JPanel representing the whiteboard area where users can draw. The upper right corner displays the user list of joined users, while the lower right corner contains the chat box for user communication. Both the user list and chat box are placed within a JScrollPane to allow for scalability. Additionally, the manager window features a file menu bar at the top, providing privileges for the manager to perform actions such as saving the canvas

or creating a new canvas.

The object of ColorPalette class becomes visible when the user presses the 'Color' button on the whiteboard tool. This class provides a GUI for the user to select a color based on the RGBA value. The user can input each value and preview the color by pressing the 'OK' button, and if they decide to make the change, they can press 'Apply & Quit'.

The KickUser class represents the GUI that appears when the manager attempts to kick users by double-clicking on their name in the user list scroll pane. After performing the action, a dialog will be displayed, asking the manager to confirm if they really want to kick out the user.

D. Implementation Details

This section will discuss the implementation of both basic and advanced features, with the related screenshots provided in the Appendix.

1. Basic Feature

Basic feature includes Shared interactive canvas, drawing tools: line, circle, oval, rectangle, text input and choose colors.

a. Shared interactive canvas

When the whiteboard is created by the manager, all other clients can join and use the same whiteboard. To achieve this, every action made by each client, including the manager, is sent to the IRemoteWhiteBoard and stored in an ArrayList<WhiteboardShape>. The action is also triggered using the IRemoteWhiteBoard's broadcast function called broadcastShape(WhiteboardShape shape), which communicates with each IRemoteWhiteBoardClient and refreshes all the canvases. When a canvas is refreshed, it retrieves the ArrayList<WhiteboardShape> from the IRemoteWhiteBoard and loops over each WhiteboardShape element. For each element, a function called draw(Graphics2D g2d) stored in the WhiteboardShape is executed to support the drawing mechanism. In summary, whenever a client performs an action, all clients' canvases are repainted with all the shapes again. The share whiteboard GUI refer to *Appendix B figure 2*.

b. Line, circle oval and rectangle

The drawing of the four basic feature shapes is supported by the java.awt.Graphics2D package. The size of the drawing is determined by the mouse listener, which listens to functions such as mousePressed, mouseReleased, and mouseDragged. These mouse-related functions record the points of each action and store all the related points into an ArrayList<Point> called pencilPoints. These points are later used to initialize a new WhiteboardShape object, along with other features such as colors, strokes, and shape types. The shapes example refers to *Appendix B figure 3*.

During the repainting process, the draw function in WhiteboardShape selects the appropriate drawing method based on the ShapeType. For example, it may

call the drawLine(Graphics2D g2d) method to draw a line shape on the user's canvas when the shapeType == ShapeType.LINE. Each corresponding drawing shape function draws based on the stored attributes, such as color, stroke, and points.

c. Choose Color

When the user presses the 'Color' button on the GUI, a color palette will appear. Unlike the 16 colors available in the basic features, the color palette provides almost unlimited options with 16,777,216 possible color combinations. The user can simply input the values for R (red), G (green), and B (blue). Upon pressing the 'OK' button, the background of the color palette frame will change to the user's chosen color, allowing them to preview how it might appear. If the user is satisfied with the color, they can apply it by pressing the 'Apply & Quit' button, and they will return to the whiteboard with the new color. The color palette refers to *Appendix B figure 4*.

d. User List

In the upper right corner, there is a JScrollPane that displays all the users currently in the whiteboard, excluding the manager. Only the manager has the authority to implement the kick function, allowing them to remove selected users from the whiteboard. Refers to *Appendix B figure 5*.

e. Approval to join

When a client joins the whiteboard, they need to be approved by the manager. A JOptionPane will show up, and the manager needs to decide whether to allow the client to join the whiteboard. If the manager chooses "Yes," the client will be able to join. Refers to *Appendix B figure 6*.

2. Advanced Feature

The project supports several advanced features, including a chat window that allows users to communicate with each other. Additionally, the manager has the ability to kick out specific users. Another advanced feature is the file menu, which is only displayed in the manager's whiteboard. It provides options such as new, open, save, save As, and close.

a. Chat Window

On the lower right corner of the canvas, there is a chat box that displays the communication between all the peers, along with a textArea where users can input their messages. Next to the text field, there is a 'send' button for users to send their messages to the chatbox. Each message is shown in the chatbox along with the username, allowing clients to differentiate the source of each message. Refers to *Appendix B figure 7*.

To ensure synchronization of the chat window among all the peers, it follows the same broadcasting flow as painting. First, the message is sent and stored in the ArrayList<String> called textList in the IRemoteWhiteBoard. Then, the broadcastMessage(String message) function is triggered to broadcast the message to all clients, prompting them to refresh their textList and repaint the

chatbox. This ensures that every client updates their chatbox in real time.

b. Kick out certain user

The manager has the ability to kick out users from the user list displayed in the upper right corner of the canvas. Although there is no explicit button for kicking users, the manager can double-click on each user's name. This action triggers a dialog asking the manager to confirm if they really want to kick out the user (refers to *Appendix B figure 8)*. If the manager clicks 'Cancel', the dialog is closed and they return to the whiteboard. If the manager clicks 'Sure', the selected user is kicked out and their name disappears from the user list. Refers to *Appendix B figure 9*.

To ensure that this action is broadcasted to all users, it triggers the kickUser(int userIndex) method in the IRemoteWhiteBoard. This method first disposes the window of the selected user, removes them from the List<IRemoteWhiteBoardClient> clientList, and their username from the List<String> users. Then, it loops over all the remaining clients to update their user lists shown in the whiteboard, reflecting the removal of the kicked user.

c. File menu for manager

In the manager's whiteboard window, an additional file menu appears at the top of the window, allowing the manager to perform certain actions on the canvas. The file menu refers to *Appendix B figure 10*.

The "New" button indicates that the manager wants to clear all the drawings on the canvas, creating a new empty canvas.

Clicking the "Open" button brings up a JFileChooser, which allows the manager to load previously saved data. This enables the saved drawing shapes to appear on the canvas. Refers to *Appendix B figure 11.*

The "Save" and "Save As" buttons also bring up a JFileChooser, allowing the manager to choose where to save the file and specify the filename. However, when using the "Save" button, the JFileChooser is only displayed for the initial save. After that, if the manager presses the "Save" button, the file will be saved directly with the same name and path as before. On the other hand, the "Save As" option allows the manager to save the drawing data wherever they want and specify a new name each time if desired. Refers to *Appendix B figure 12.*

Finally, the "Close" button allows the manager to close the whiteboard, terminating the connection for all users.

E. New Innovation

This part introduces the creative aspects of this project, including the color palette, pencil tool, and stroke/font size options.

a. Countless color, Opacity, brighter and darker

As mentioned above, the color is selected using a color palette GUI, allowing the

user to choose a color based on its RGB value. This provides a wide range of options, with a total combination of up to 16,777,216 different colors. Additionally, the color palette offers an additional dimension called Alpha, which determines the opacity level of the color. By setting the value to zero, the color becomes transparent. Refer to *Appendix B figure 13.*

Furthermore, the color palette provides two handy buttons: "Brighter" and "Darker". These buttons allow the user to adjust the color's brightness, helping them move closer to their desired color shade. Refer to *Appendix B figure 14*.

b. Pencil tool

Besides basic drawing tools such as circles, ovals, rectangles, and lines, this project also offers a pencil tool that allows users to draw any shape they like, just as they would with a regular pencil. Refer to *Appendix B figure 15.*

c. Stroke adjustable

There are two additional buttons called "Thicker" and "Thinner" that allow users to adjust the stroke width of each shape. In the case of text, these buttons would instead adjust the font size. Refer to *Appendix B figure 16.*

Appendix A

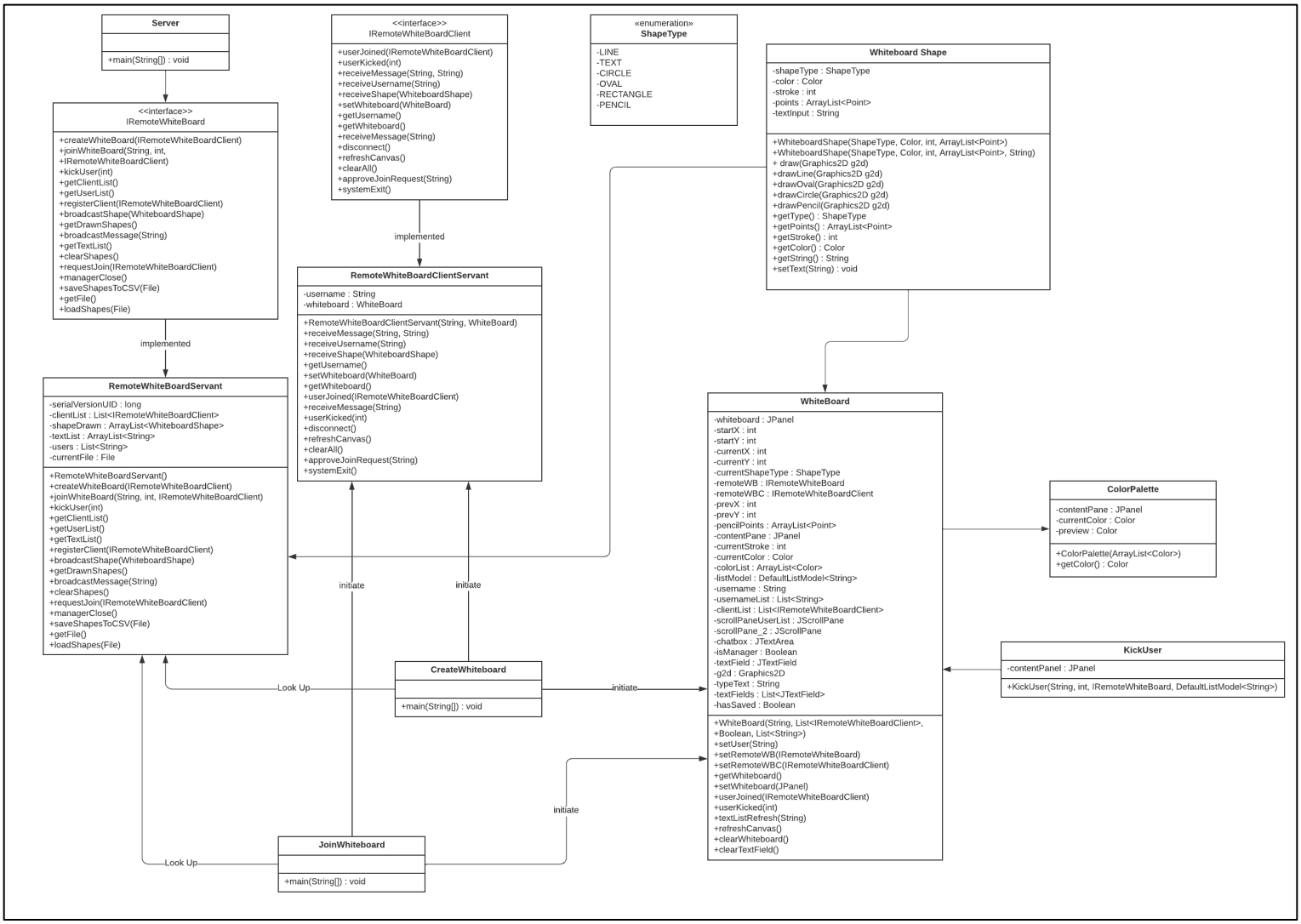

Figure 1. UML Diagram

Appendix B

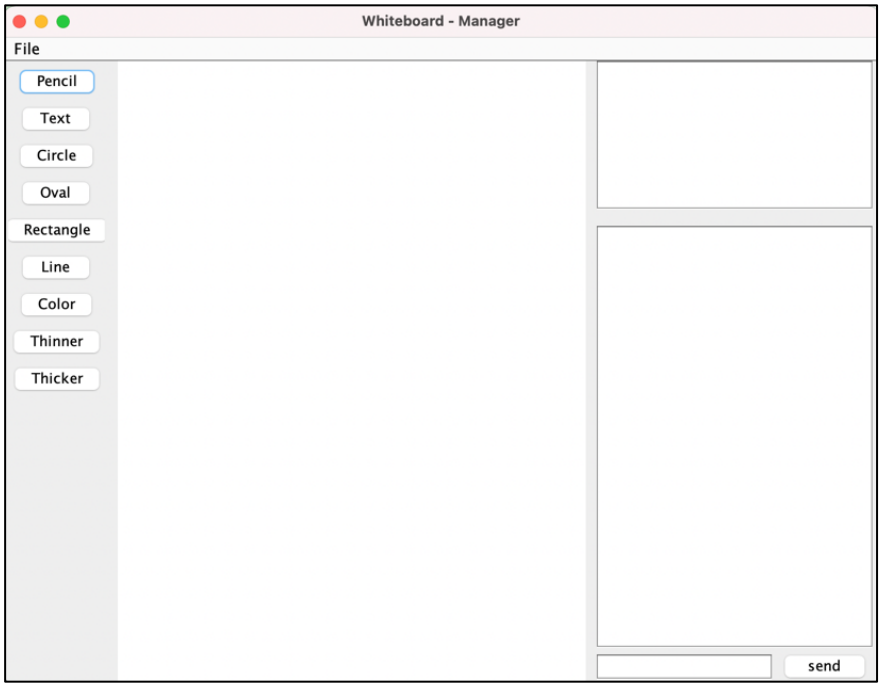

Figure 2. Whiteboard GUI

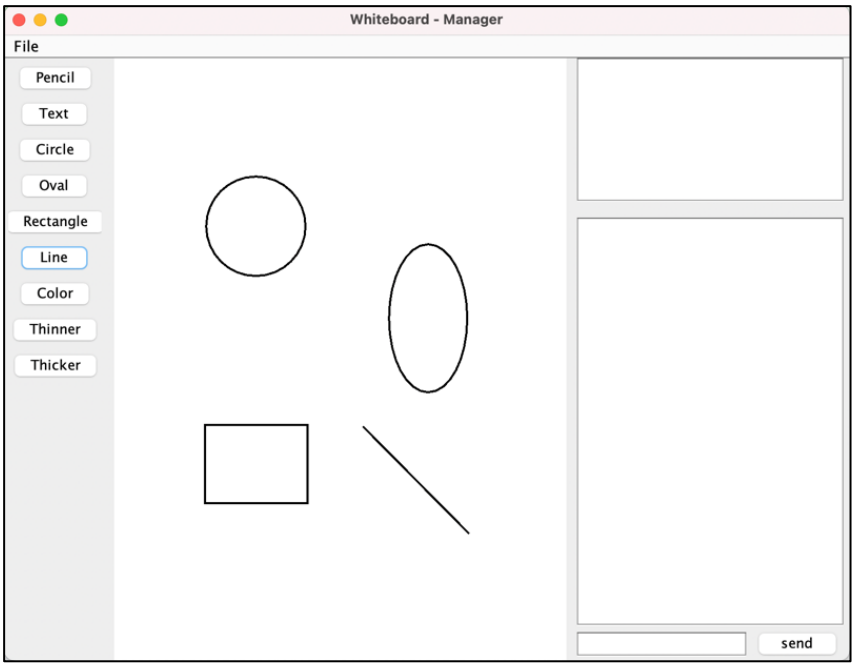

Figure 3. Shapes Example

Figure 4. Colour Palette

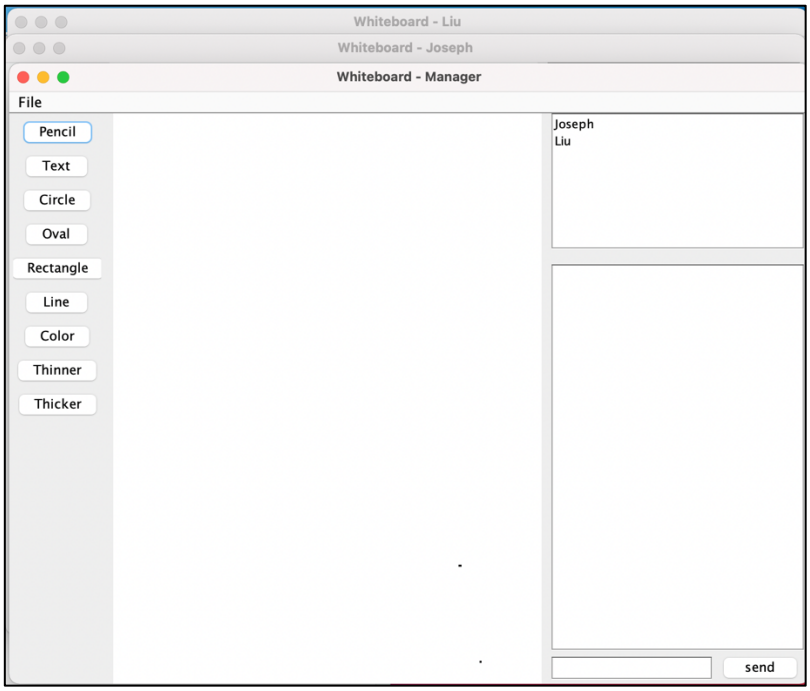

Figure 5. JScrollPane to show active users (excluding Manager)

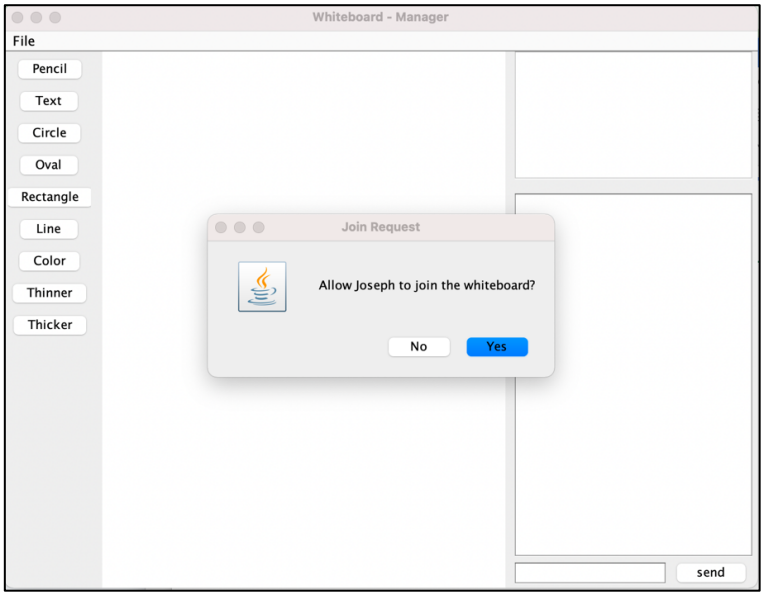

Figure 6. JOptionPane for Approval to Join

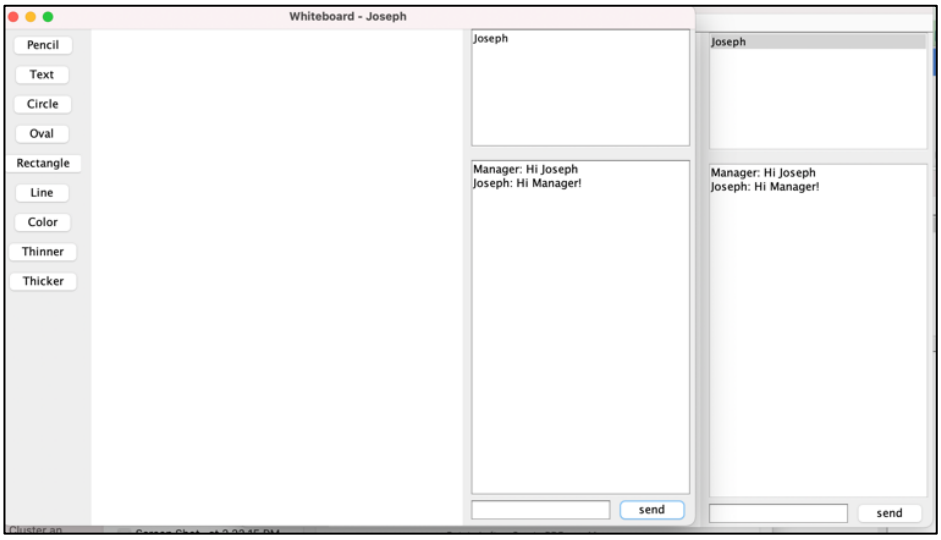

Figure 7. Chat Window

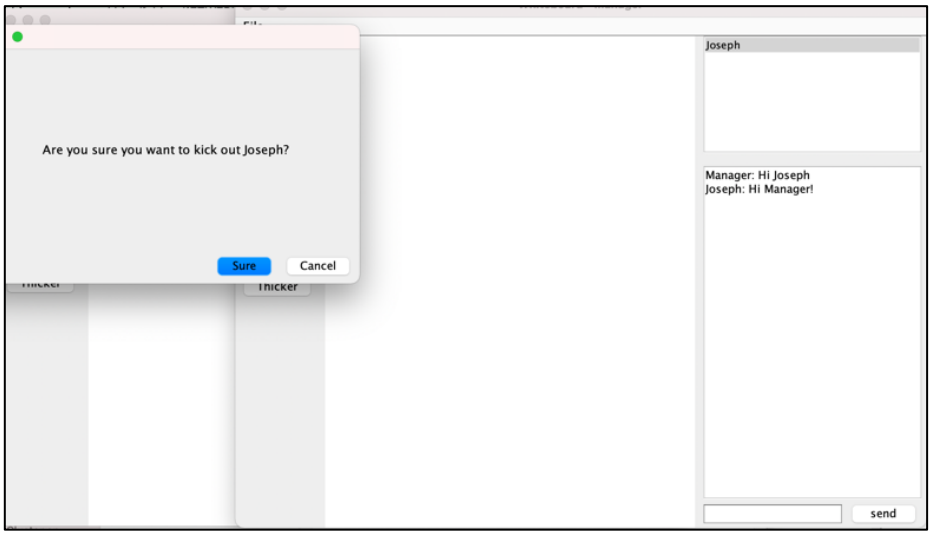

Figure 8. Confirmation to Kick User (Manager window)

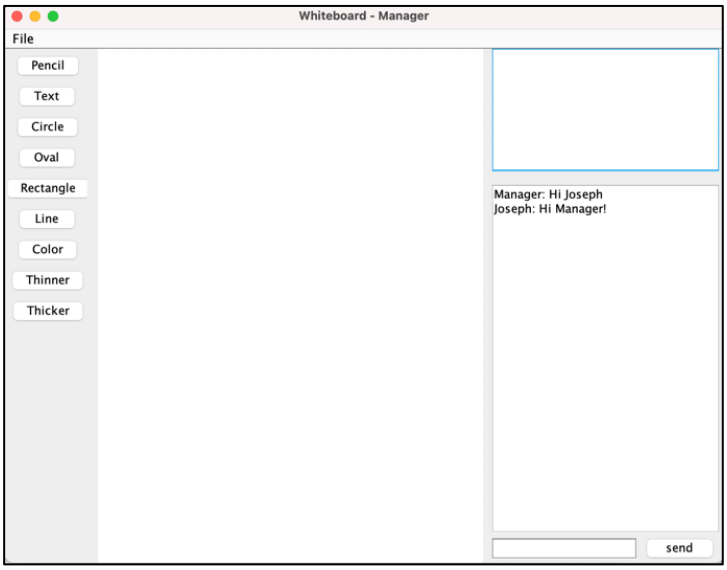

Figure 9. User kicked after confirmation

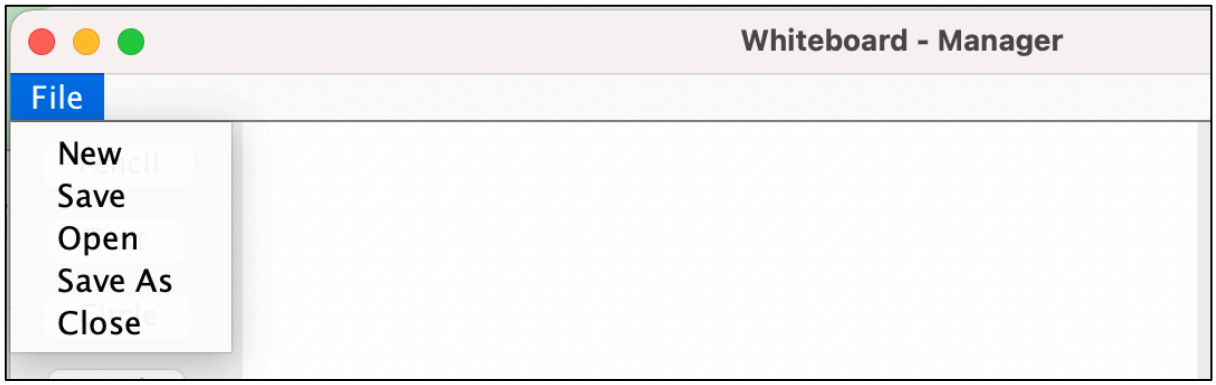

Figure 10. File Menu on Manager Window

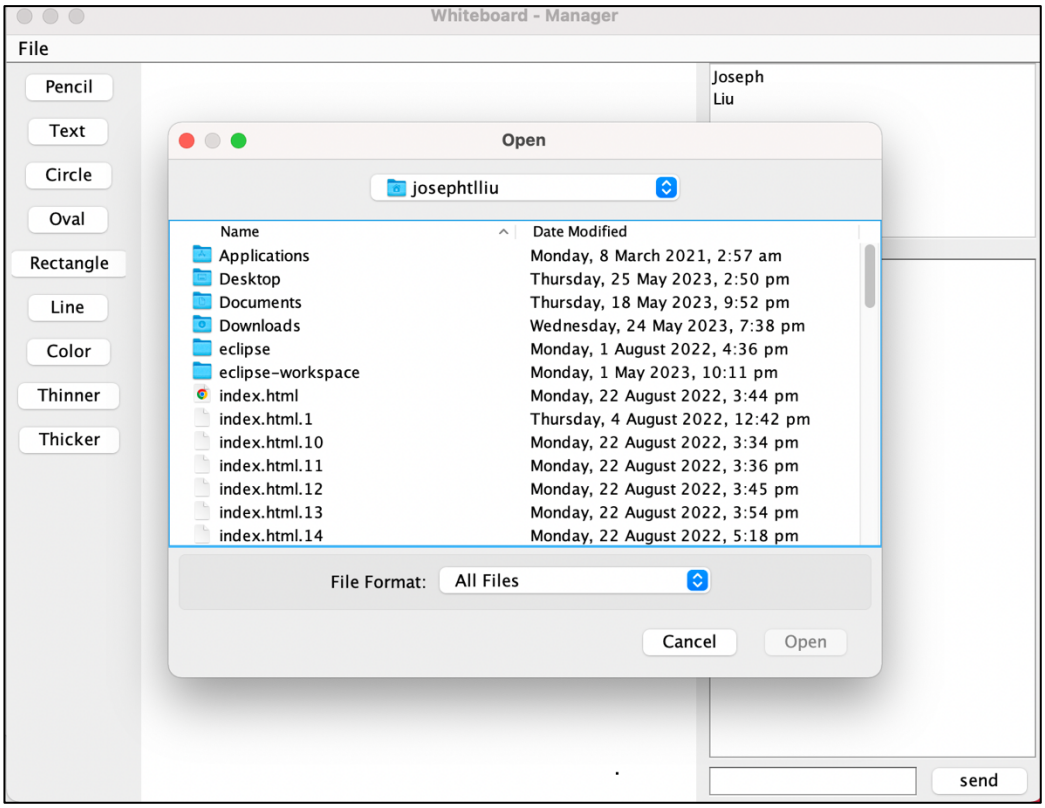

Figure 11. 'Open' button in File Menu bringing out JFileChooser

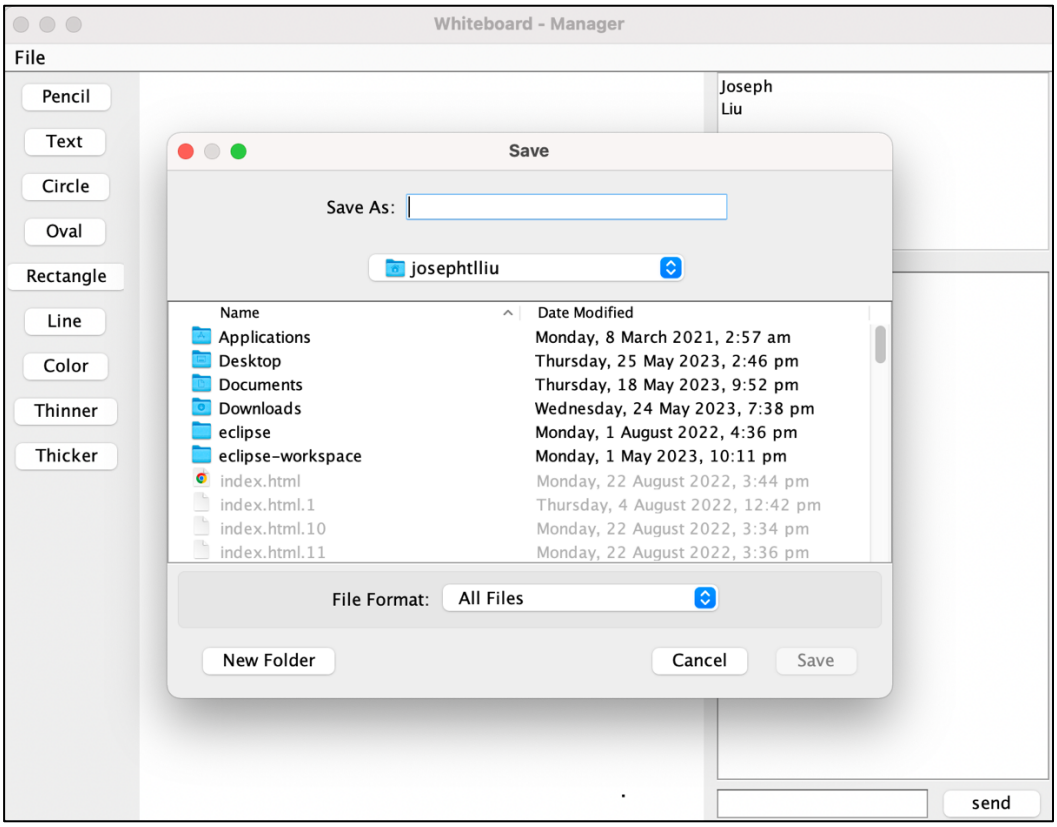

Figure 12. 'Save As' button in File Menu bringing up JFileChooser to save file with new name

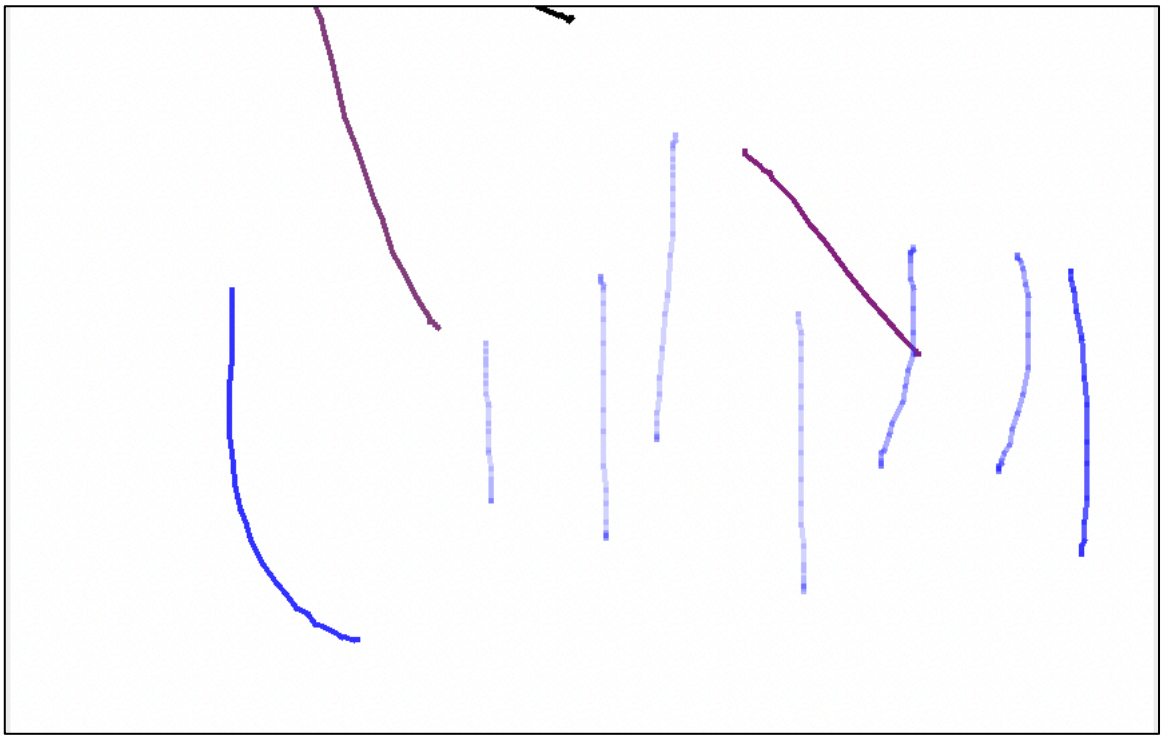

Figure 13. Opacity option for colours function on colour palette

Figure 14. Brighter and Darker options for colours function on colour palette

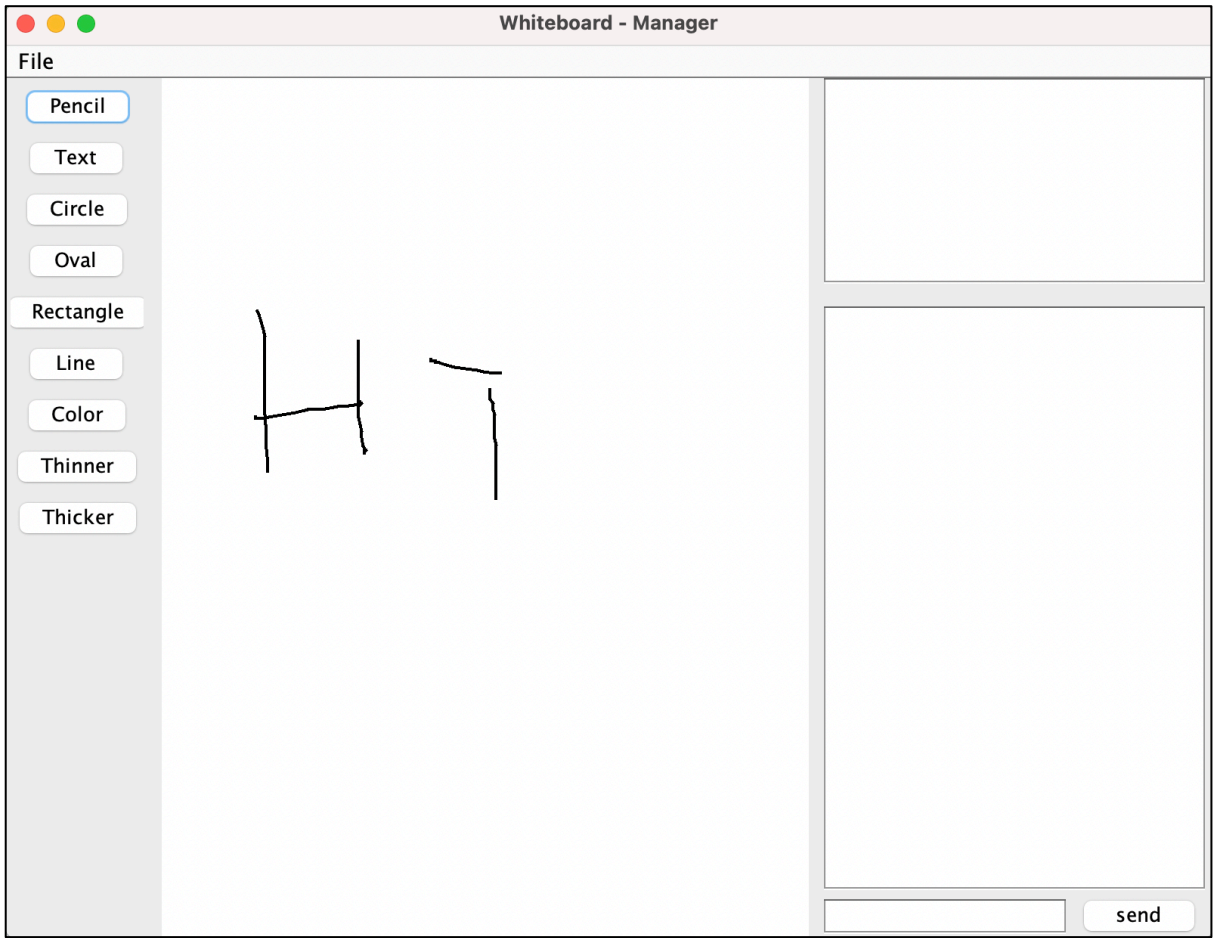

Figure 15. Pencil tool

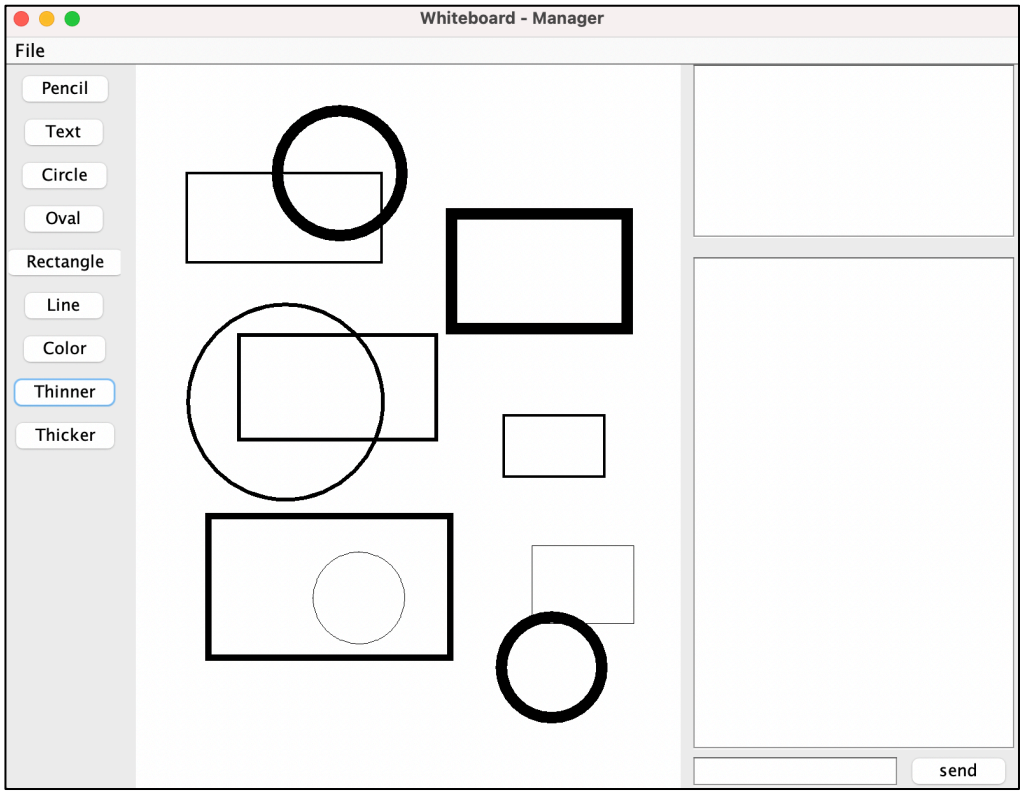

Figure 16. Adjustable stroke (Thicker and Thinner)