

DISTRIBUTED SYSTEMS (COMP90015)

NAME: ABDUL HANNAN GHAFOR

STUDENT ID: 1136001

Semester 1, 2022

ASSIGNMENT 2

SHARED WHITEBOARD APPLICATION

Context

This assignment requires us to develop a Distributed shared whiteboard Java based desktop application. The Application, being shared, enables multiple users to collaborate on a single drawing project. The application is focused on real time sharing of canvas so any action performed by a collaborator on the whiteboard is replicated to the screens of every other person in that project. This whiteboard also features different shapes to choose from, mainly triangle, circle, rectangle and line. Apart from that users can also scribble and write some texts on the canvas. This application also has a chat based system where different users, working on the same project can send texts to the other collaborators. A user management system is also designed which enables the application to track multiple projects at the same time with each project containing multiple collaborators. The multithreading and communication between the collaborators is done through Remote Method Invocation (RMI) which is a thread-safe implementation. Because of RMI, it can support multiple concurrent users and the server is able to handle multiple requests at the same time.

Components

My project has multiple components

1. Whiteboard Server Interface, an RMI based implementation to handle multithreading and communication between the users. It is basically a message router.
2. Whiteboard Server, which focuses on different whiteboard projects and user management.
3. Whiteboard Client Interface, a RMI based callback interface which enables the server to invoke the methods on client side to perform replication actions.
4. Whiteboard Client, a JavaFX based client application. Spanning over multiple files and multiple screens, this application uses FXML documents and controllers to create beautiful GUI which displays the paint canvas along with the chat window.
5. Shapes Library, which contains the implementation of different shapes required for this assignment.

Application Walk-through

First, Whiteboard server application is run. That binds the interface to the RMI Registry. The default IP for the RMI Registry is localhost. After that the whiteboard client main application is run. That will first does the lookup for the RMI based whiteboard server. After finding that server, it starts a login page. This login page asks for different information to connect to a particular whiteboard instance. At first it will ask for your username with which the user wants to register. Once the user put in the username and click register, the client will invoke the server's register function to register the user to the server. In return, server will send client ID to the user

Along with clientID, a list of the whiteboard instances currently running is also returned to the user. This list is in the form of the whiteboard IDs. User now has the option to either choose from the running instances or create a whiteboard project instance himself. In order to create the new whiteboard project instance, the user will have to click the create button. Once that button is clicked, the server's createwhiteboard function is invoked where the server creates the whiteboard instance object and set the user its manager, After the creation of the whiteboard instance project, the server will return the ID of the project and update the list. Now the user can either choose the project instance that he has created or connect with any other whiteboard instance that is already available there.

In order to connect with the whiteboard project instance, the user will have to click one from the list view and then click connect. This will take the user to another window which is the window of whiteboard application. The user are given multiple operations to choose from which are mentioned below.

1. Choose the shape that he wants to draw. These shapes are
 - a. Rectangle
 - b. Circle
 - c. Triangle
 - d. Line
2. Choose to scribble on the canvas.
3. Choose to input a text on the canvas. This will open another window where user is asked to input a text. Once user has written the text on the window, he will have click on canvas to paste the text.
4. Unlimited color options to select any color the user wants for either shapes, scribbling and texts.
5. A chat area, to text the other users within the project.
6. Useronline area, where user can see username of different users along with their ID.
7. Information about the username, the whiteboard instance, and the role of the user within the whiteboard project is also displayed.

There are two roles that the user can have,

1. Manager
2. Collaborator.

If the user creates the whiteboard project instance, then he becomes the manager of the project, all the other users who join afterwards are given the role of collaborator. A manager is given some privileges over the collaborators. These privileges are

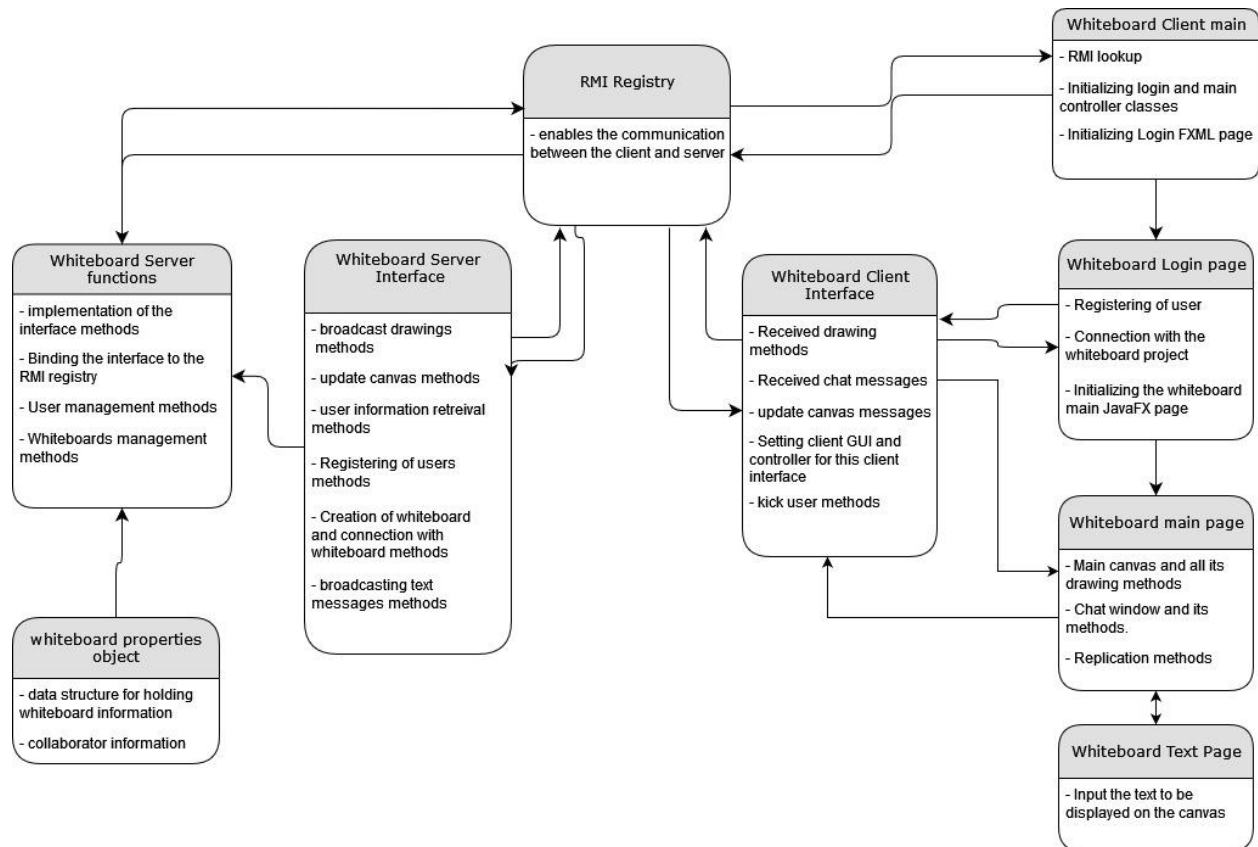
1. Ability to clear the dashboard by clicking the option File -> New.
2. Ability to open an existing dashboard.
3. Ability to save the state of dashboard.
4. Ability to kick the users/peer out.

When the File-> save option is selected, the user is given an option choose the directory where he can save the files. The dashboard is saved in a PNG format. Similarly, once the File-> open option is selected, the user is given an option to choose the directory and the file which he wants to open. All the multithreading and concurrency is dealt with RMI because of which the server becomes the message passing router apart from user management and maintaining the state of multiple dashboards.

System Architecture.

The architecture of the system follows a server client model, where server is responsible for message routing and user management, and client is responsible for the GUI. RMI is used for the communication of messages. With RMI, we can invoke method on servers or clients. RMI is multithreaded which means that the multiple clients can invoke server's same method simultaneously given the data structure is thread-safe. Thread-safe data structure means that the data structures being used within the method, which is invoked through RMI, is able to withstand simultaneous read and write operations. RMI uses stubs, marshalling and unmarshalling to communicate with the remote objects. Basic communication protocol is defined by the RMI.

In order to communicate through the RMI, there are two interface created, one at the server side so that the client can invoke server's method for operations like registering user, creating the whiteboard, adding the user as a collaborator to the existing project etc. A callback client interface is also created with the help of which the server can use the method defined within the client like drawing on the canvas as well as writing texts on the chat window.



The diagram above shows the whole system architecture. Following libraries have been used for the implementation of the above architecture.

1. JavaFX
2. RMI
3. Java Swing for saving the documents.

There are three main parts of the server.

1. Whiteboard Server interface
2. Whiteboard main server application
3. Whiteboard properties data structure.

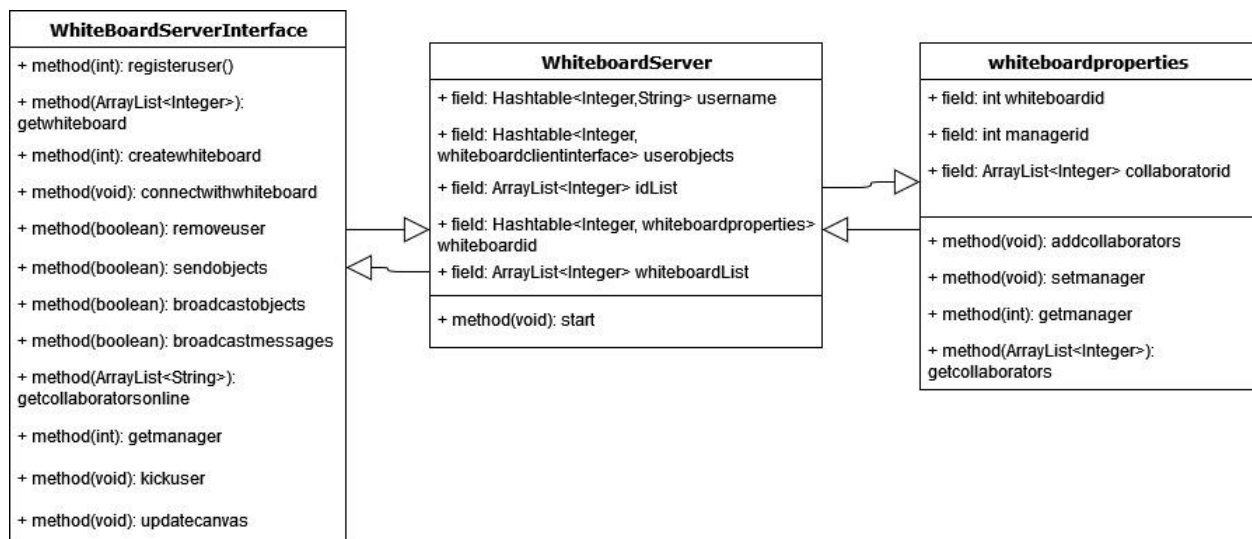
As mentioned in the system architecture diagram, the main server application focuses on registering the server interface to the RMI registry. It also does the user management by registering the users, assigning the user to different whiteboards, maintaining the information about the whiteboards, maintaining the collaborators on different whiteboards among other things. In order to maintain the information about different whiteboards, it uses the data structure “whiteboard properties”. It also maintains the information of different RMI client objects which is used by the interface to communicate with the client. Whiteboard Server Interface is responsible to replicate all the canvas state on all collaborators. It is also responsible to route text messages to all the collaborators.

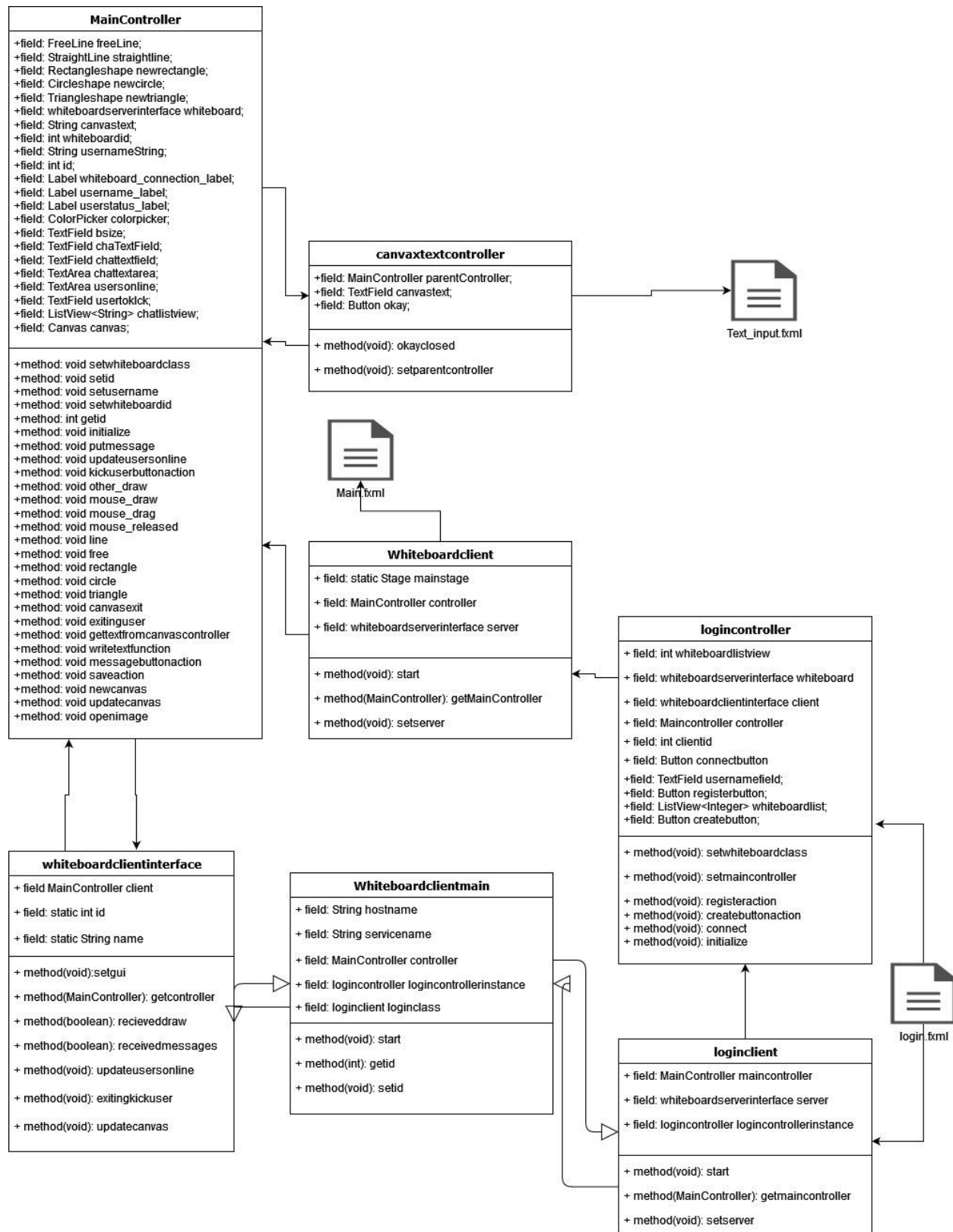
Client side of the application has FXML based GUI application. JavaFX library is used for this purpose. There are multiple components of the client application.

1. Whiteboard Client main application.
 - a. Responsible for initializing the login page.
 - b. It also connects with the server.
2. Login Page
 - a. It initializes the main controller and login controller.
 - b. It uses the FXML document to spawn the GUI application.
 - c. Responsible for registering the user to the server as well as connecting the user to the whiteboard.
3. Main whiteboard page.
 - a. This is invoked by Login page once the user is assigned a specific whiteboard.
 - b. It also uses the FXML document to spawn the GUI application.
 - c. It uses the main controller initialized by the login page.
 - d. Main controller has all the definitions of canvas drawings and chat application. It uses the custom shapes library available as a package for different shapes.
4. Whiteboard client Interface.
 - a. This contains all the methods which are used by the server to maintain the state of the whiteboard across all its collaborators.
 - b. It also contains methods that are used by server to deliver the text messages.
 - c. It also contains the user management methods that are used by server to manage different users.

Class Design

Following is the detail of different classes within the server side of the application.





Above is the UML class diagram of the client side of the application

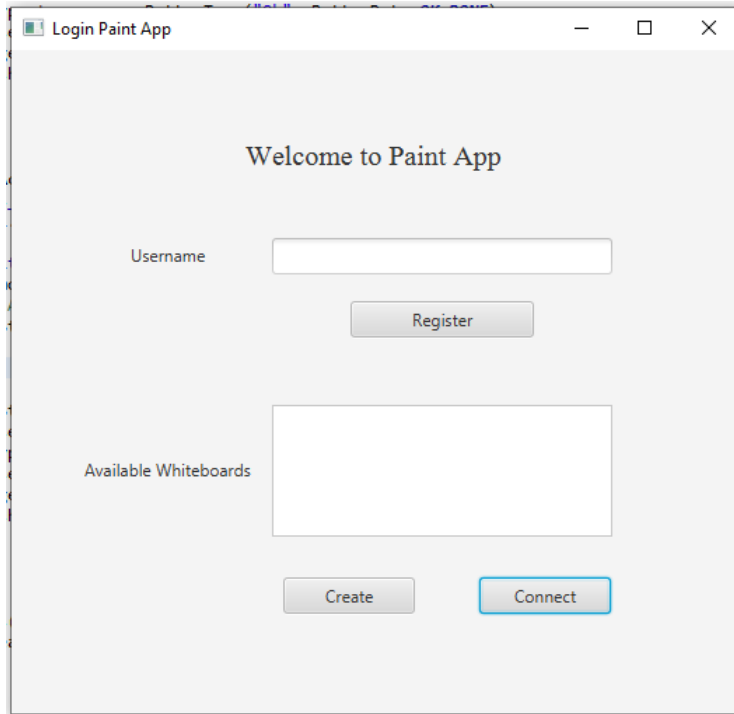
Communication Protocol

While the basic network communication is done through RMI and uses TCP sockets. Application level communication protocols are still needed to be set for the sending of the drawings and messages. Since the JavaFx is not serializable, the communication protocol that I have come with is described below.

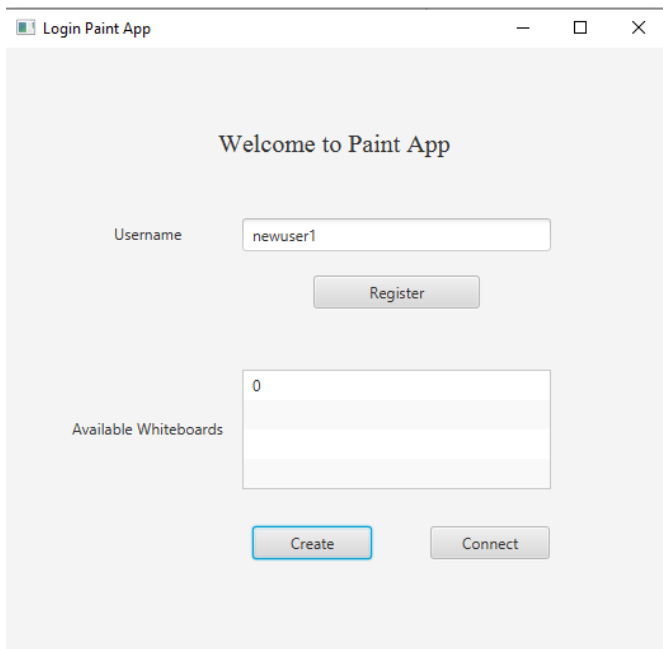
1. Different method are invoked on the basis of mode of drawing selection and mouse actions on the canvas.
2. There are three mouse actions,
 - a. Mouse clicked.
 - b. Mouse dragged
 - c. Mouse released.
3. The modes are
 - a. Text
 - b. Line
 - c. Free
 - d. Rectangle
 - e. Circle
 - f. Triangle
4. The information sent to the other users are
 - a. Mode
 - b. Mouse actions (start, drag, end)
 - c. Coordinates (X,Y) of the canvas
 - d. Color
5. Based on this information the users replicate the drawings of other users.

GUI

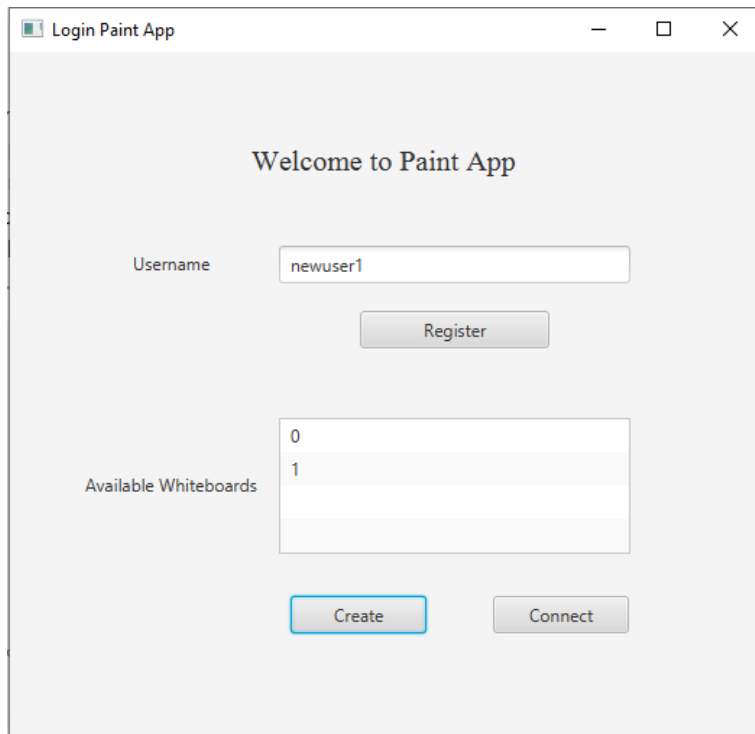
The GUI is designed in the following way.



After starting the server, whenever the client application is started, the above login screen will appear to the user. As displayed in the above screenshot, the user is asked a username in the first textfield. After putting the username in the textfield and pressing the register button, the user will be registered with the server. Below the register button, there is a listview that will display the numeric IDs of all the available whiteboard project instances present.



The user can either create a new whiteboard project instance or join the already present project instance. Server will automatically create the project instance for the user if the user clicks the create button. The new dashboard will join the list of the available whiteboards instances.



Login Paint App

Welcome to Paint App

Username: newuser1

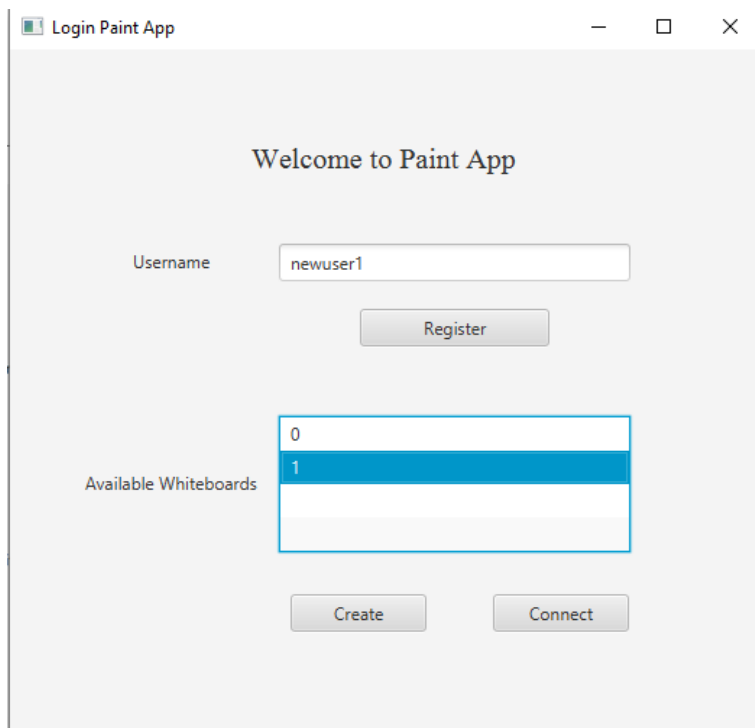
Register

Available Whiteboards:

- 0
- 1

Create Connect

Now the user can join the whiteboard that he created or any whiteboard that is already present. In order to join, the user will have to click the whiteboard ID in the list and click the button connect.



Login Paint App

Welcome to Paint App

Username: newuser1

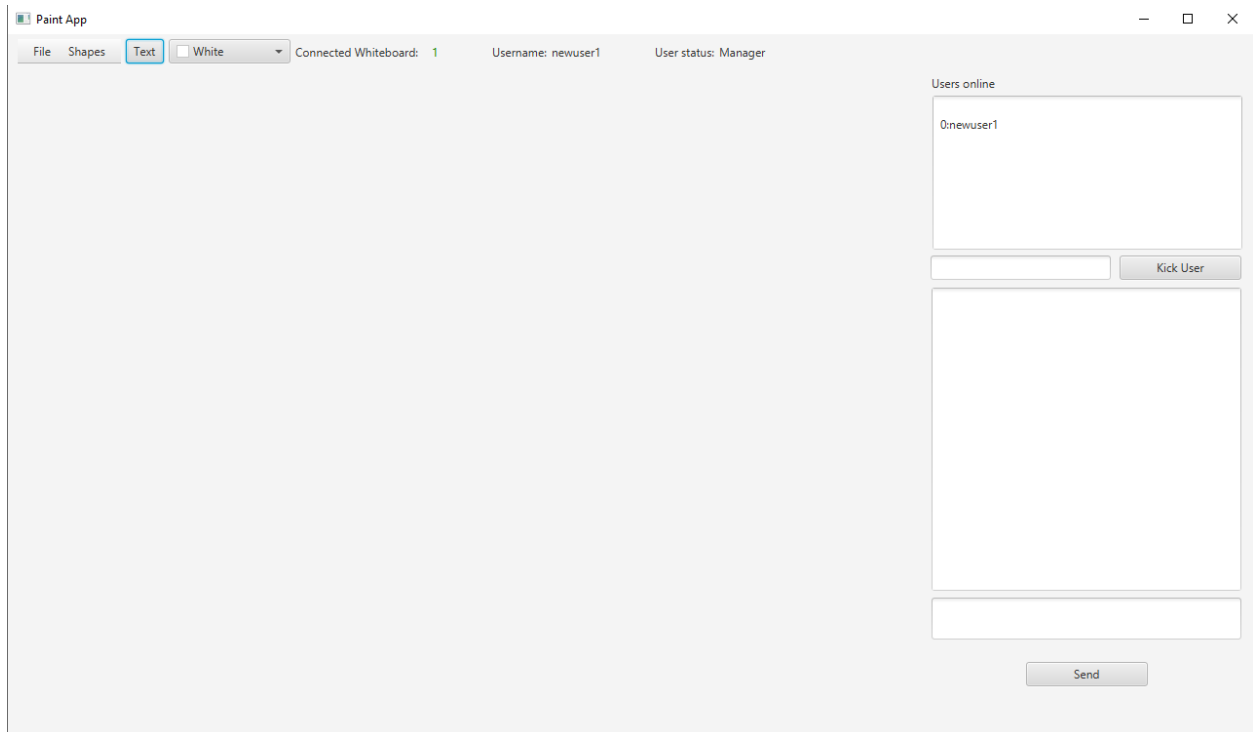
Register

Available Whiteboards:

- 0
- 1

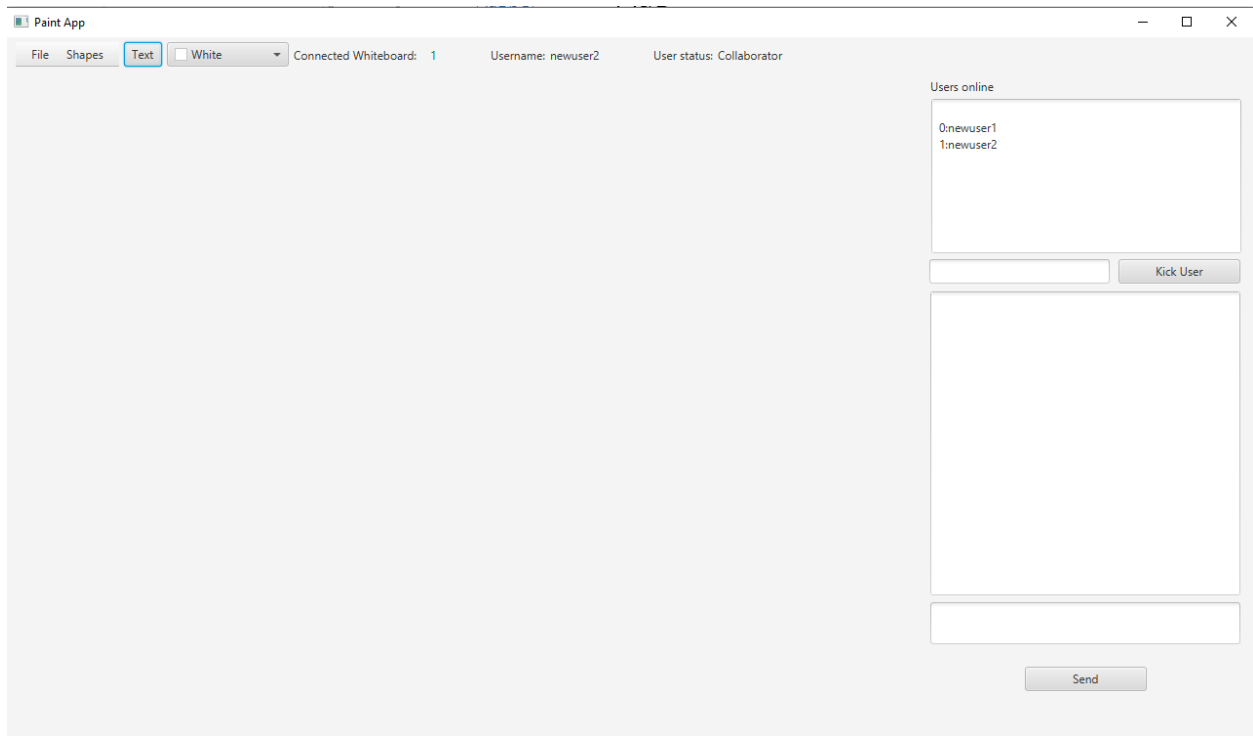
Create Connect

Clicking the connect button will take the user to the whiteboard dashboard. Since in this demonstration, we have clicked on the dashboard that the user has created. He will be given the status of the manager and he is the only one present in this whiteboard project as shown in the below picture.



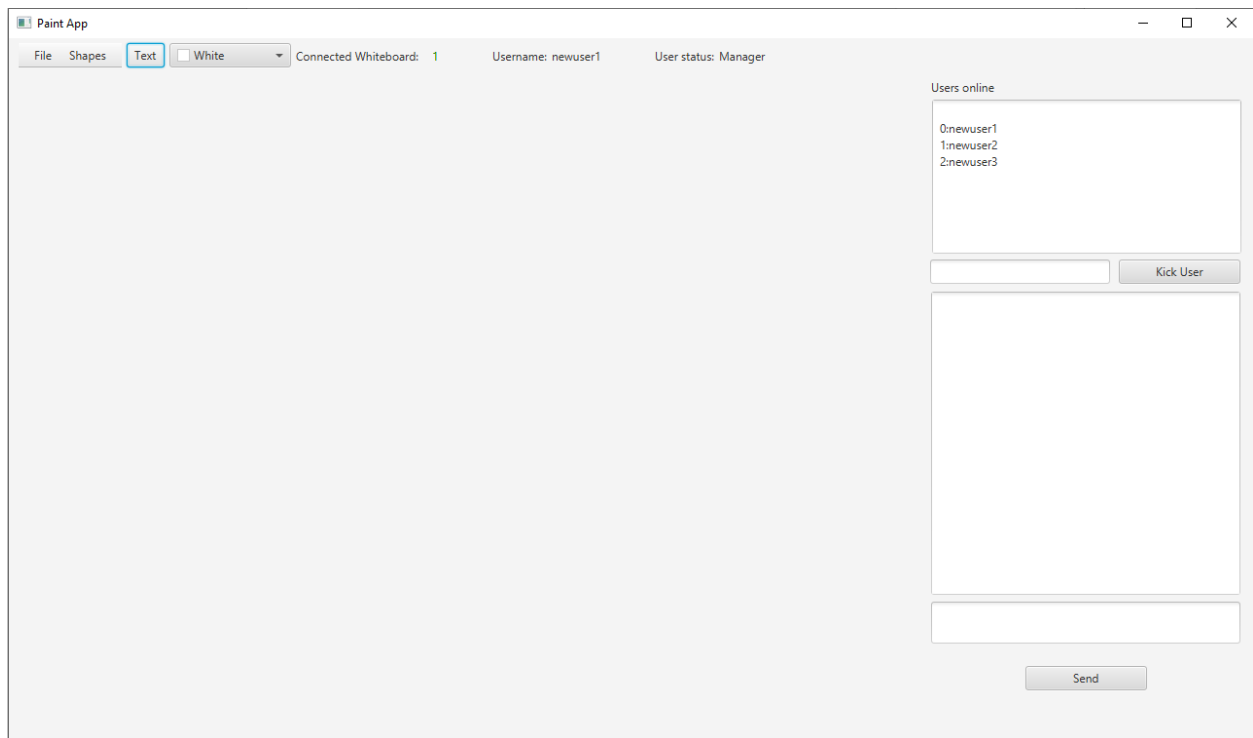
In the above picture, we can see that the connected whiteboard was "1", which was the whiteboardID. Username and user status is also mentioned in the GUI which in this case, is the manager. Users online gives the information of the userID and the username of the connected user which are separated by ":". In this case, it is "0:newuser1". The system differentiates different users with their unique IDs, so multiple users can have same usernames as well.

The following screenshot shows the dashboard of a collaborator.

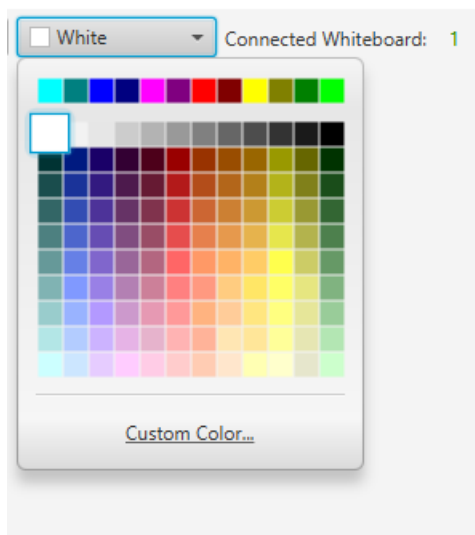


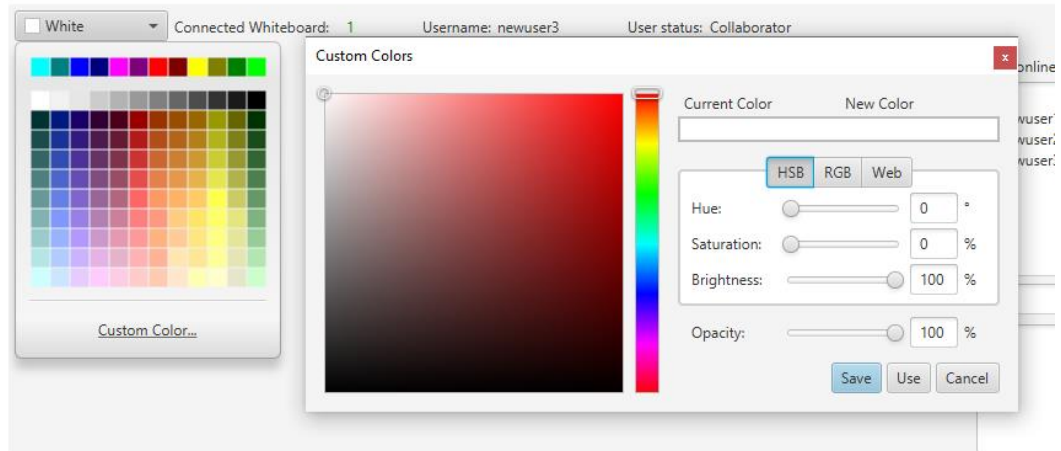
As you can see the status of this user is Collaborator. Upon joining the dashboard, we can see that the users online window now shows that there are two users. For the sake of this demonstration, we will also include another user to this whiteboard project instance.

As multiple users join the whiteboard project, users online window gets updated for each user as shown in the below pic where the manager's dashboard now shows that there are three users connected with this dashboard.



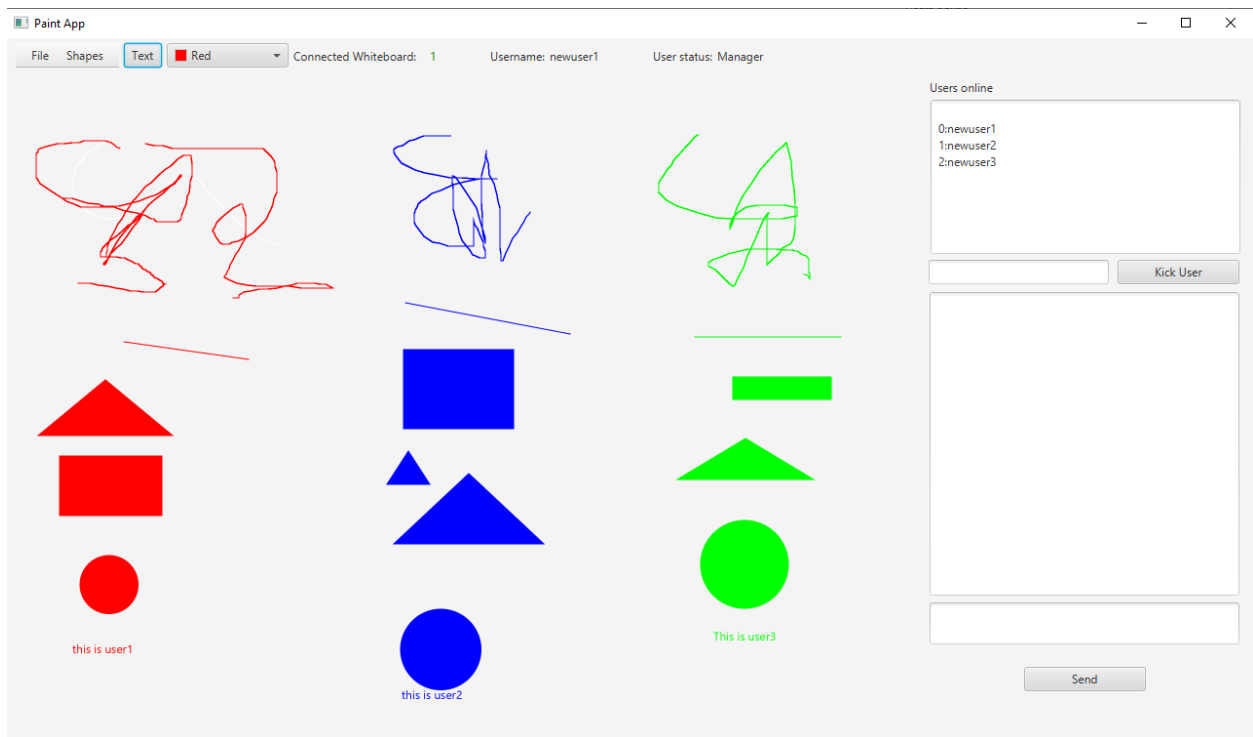
Any user who will draw or write on the dashboard will be shared with all the users in this whiteboard project. The role of the user does not matter there; it can be either manager or the collaborator. The color is also replicated along with the drawings. There are unlimited color options available for the users to choose from.

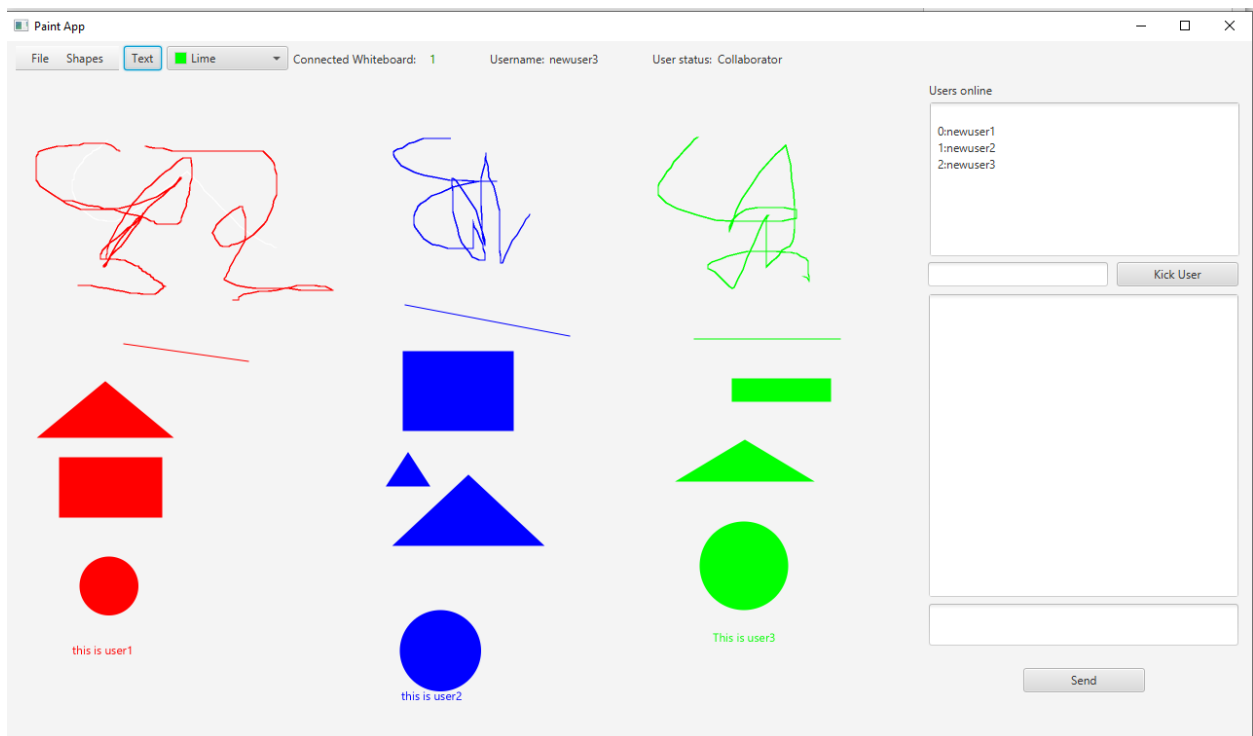
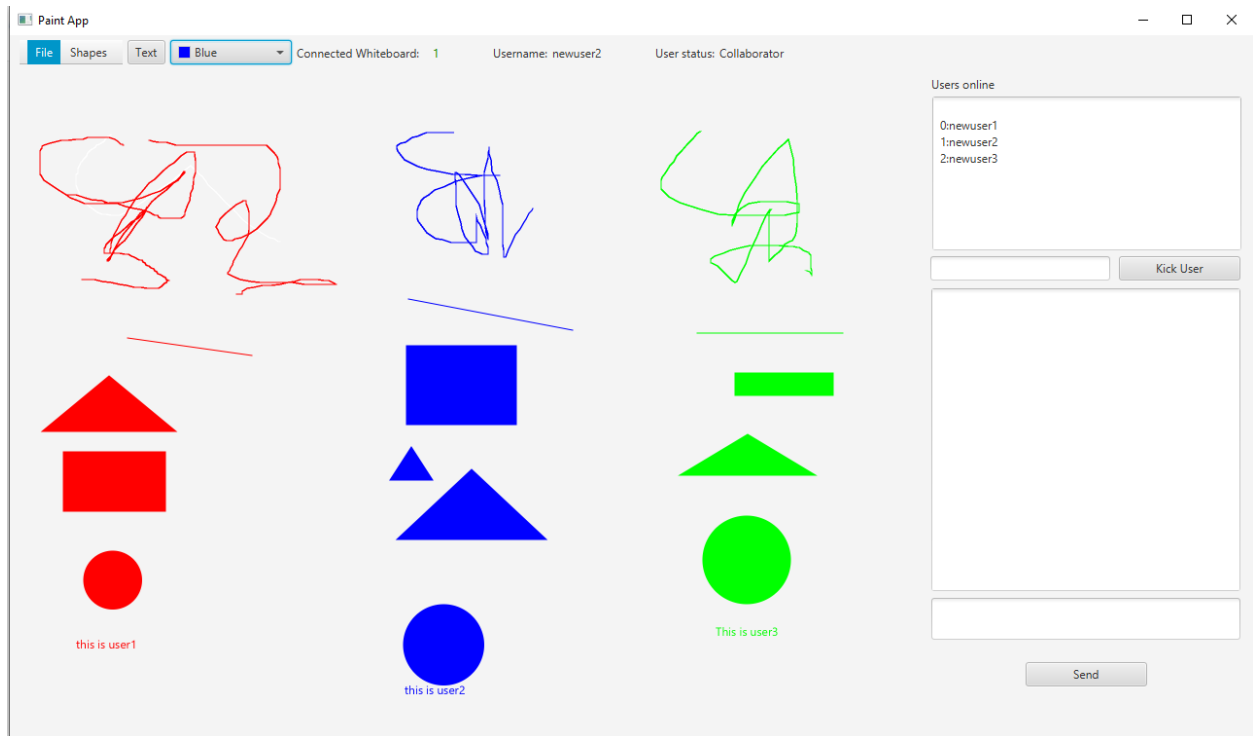




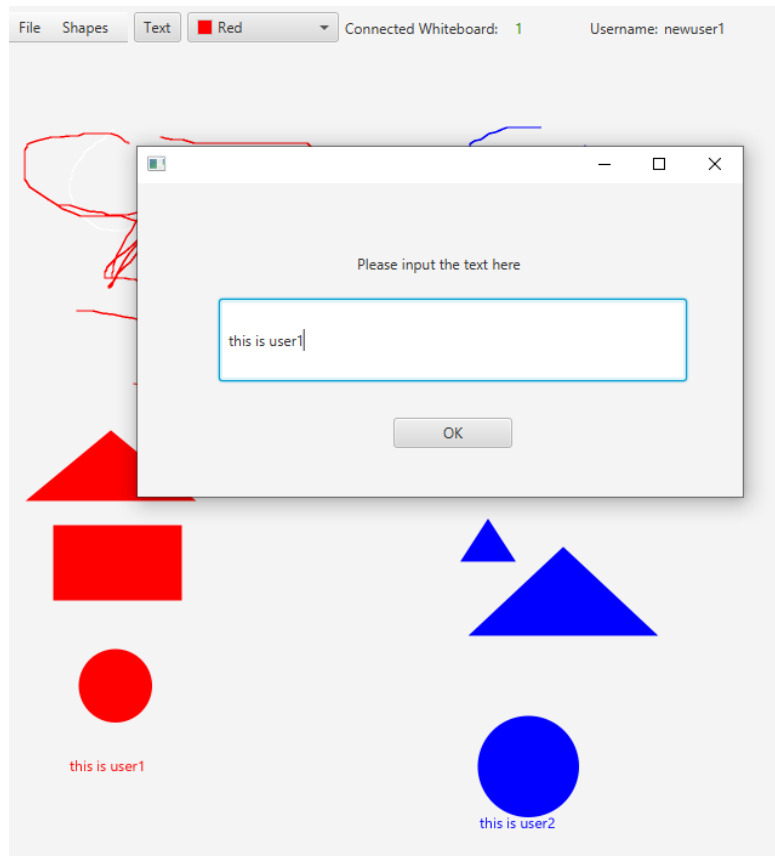
To show that the screens get replicated, we will choose the color red for user1, color blue for user 2 and color green for user3.

Below images just show the screens for all these users.





The text can put by clicking the press button “Text”, this will open another window, where the user the enter the text. After writing the text and pressing the okay button, the user can then click anywhere on the canvas to paste the text.



On the right side of the dashboard, there is a chat window where the user can chat with other collaborators on the project.

Users online

0:newuser1
1:newuser2
2:newuser3

Kick User

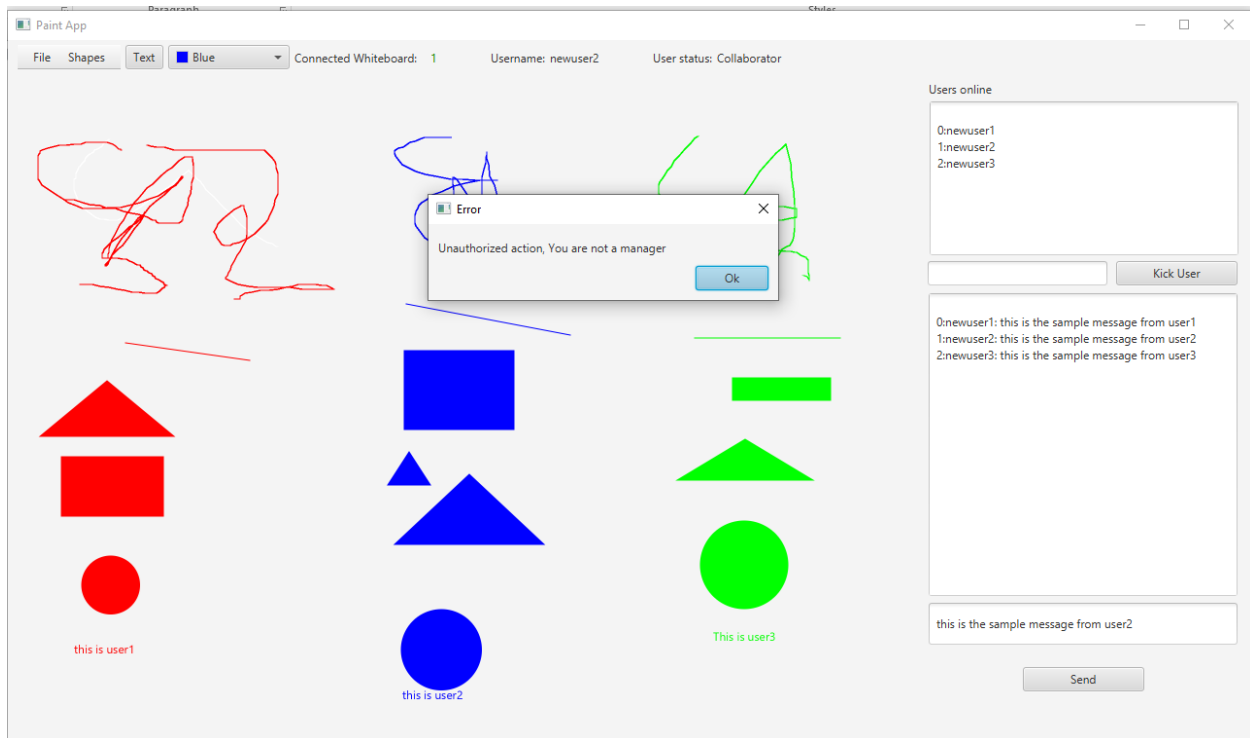
0:newuser1: this is the sample message from user1
1:newuser2: this is the sample message from user2
2:newuser3: this is the sample message from user3

this is the sample message from user1

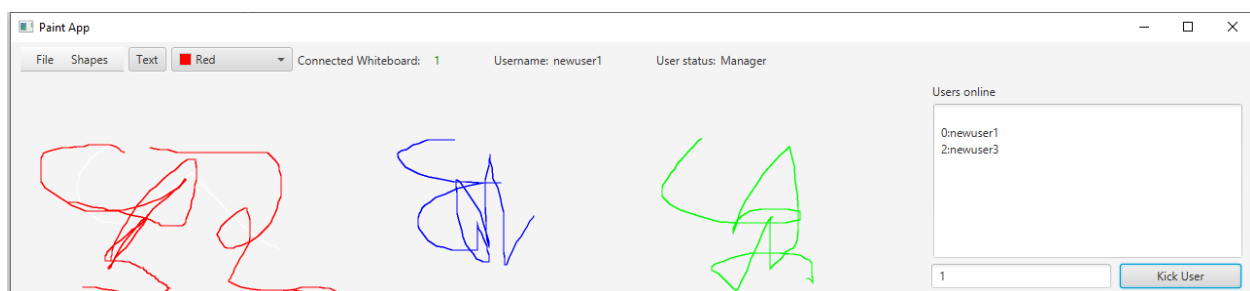
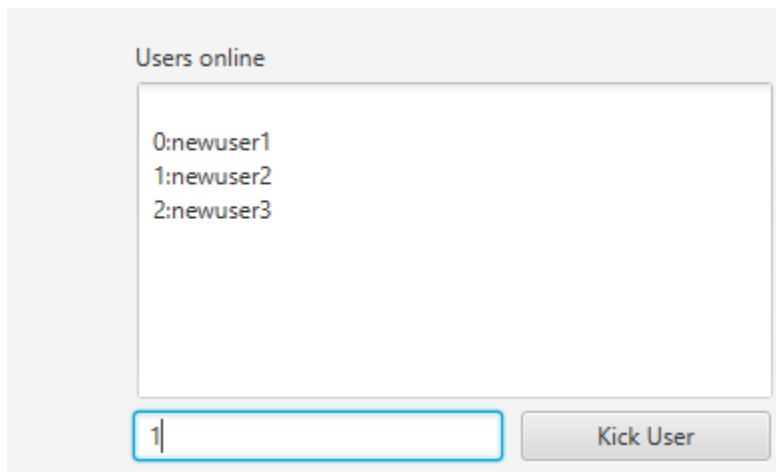
Send

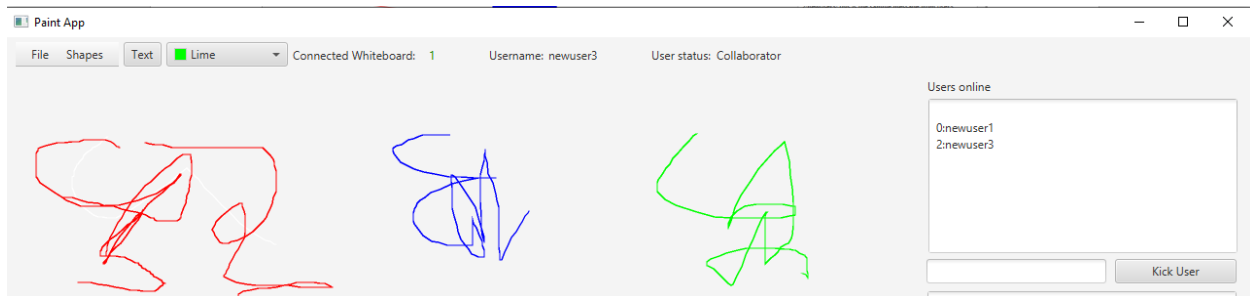
As you can see, the userID along with the username are mentioned with the chat to let other users know the user who sent a particular message.

The collaborators have limited privileges which mean that they cannot kick the peers out. Similarly they cannot reset, save or open a saved canvas on the current dashboard. When they try to do that, they get the warning message showing that they are not authorized to perform that action.



Manager can kick user out by mentioning the ID of the user and pressing the kick user button. After kicking a user, the useronline is updated across all the collaborators.

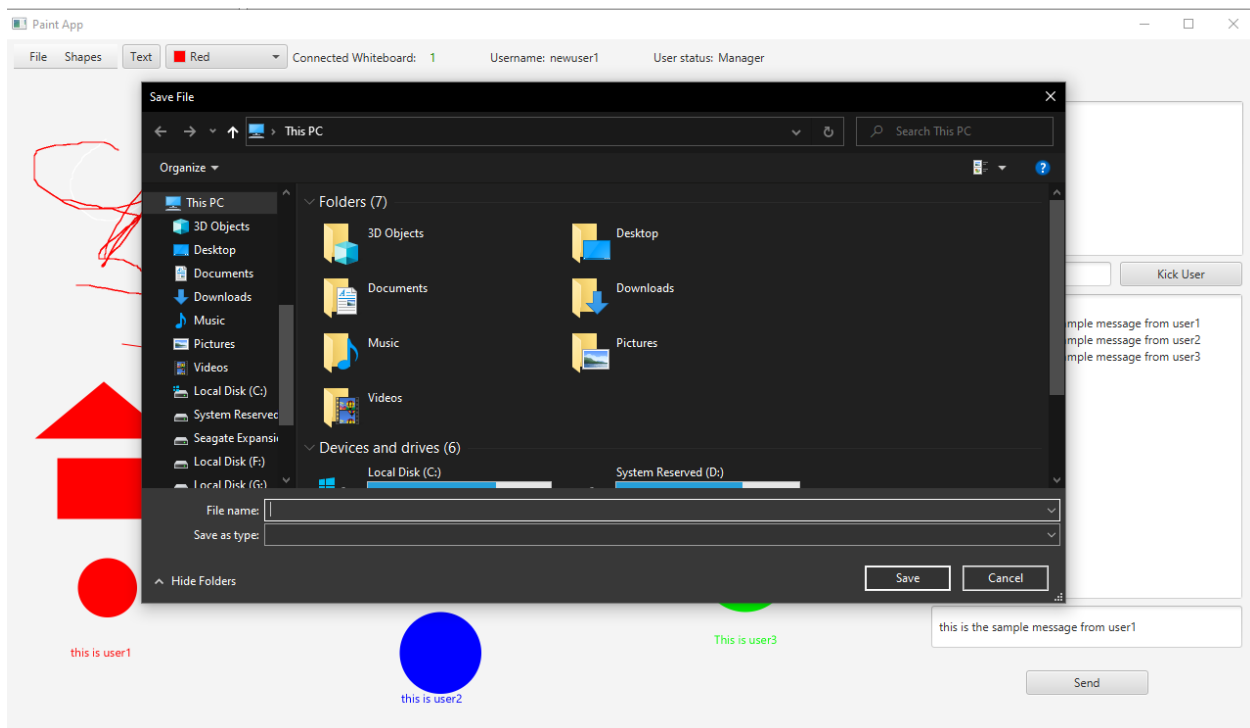




Within the file menu there are four options,

1. New
2. Save
3. Open
4. Close

Only the manager is allowed the new, save and open option. When the manager clicks the save button, it is asked to choose a file where he wants to save the drawing. The drawing is saved in PNG format and is writable.



Similarly, open will also give the option to choose the file which the manager wants to option.

Critical Analysis.

My implementation uses RMI to implement the shared and multithreaded whiteboard. This is a superior way to implement compared to socket programming where the programmer has to take care of socket connection as well as multithreading to make sure multiple users using the same data structure and methods does not corrupt the information somehow. RMI is thread-safe for most data structures.

However, we still need to take care of data structures that are not thread-safe and make sure they are synchronized. I have used JavaFX library over Swing library because of its superior graphics. While the graphics are superior, javafx objects are not serializable which means that they cannot be transmitted through the RMI interface. That is why information about the mode (free, line, rectangle, text, circle, triangle) and the method (start, drag, end) are sent through the RMI. Methods are based on the mouse actions. Coordinates and color information are also sent through RMI. Based on this information, the other clients replicate the drawing of one user.

Usage

1. Initiate the RMI registry
2. For server, use the following command.
 - a. **java -cp . --module-path "C:\Program Files\Java\javafx-sdk-18.0.1\lib" --add-modules javafx.controls --add-modules javafx.fxml -jar .\WhiteboardServer.jar**
3. For client, use the following command.
 - a. **java -cp . --module-path "C:\Program Files\Java\javafx-sdk-18.0.1\lib" --add-modules javafx.controls --add-modules javafx.fxml -jar WhiteboardClient1.jar**