Final Submission

White-Out

Jaren Mills, Hunter Bastian, Joshua Patterson, Davis King

Capstone Project [Spring, 2023]

The University of West Florida

04/23/2023

CIS4595 Capstone Project

Dr. Owsnicki-Klewe

Table of Contents

Project Description

White-Out is a collaborative annotation application that allows for collaborative annotation of screens by multiple users. It is meant to be used alongside existing screen sharing applications, such as Discord, Cisco Webex, etc. The project currently has the capability to handle multiple rooms as well as multiple users simultaneously through a client-server socket connection to our AWS EC2 instance, and is able to handle each user drawing at the same time. When a room is created, the user that created the room is labeled as the 'host' and is able to grant/revoke annotation permissions and kick users that join the room. There are multiple tools that the users can use while annotating, such as a color picker, undo/redo buttons, a clear screen button, and the ability to save the current drawings as a PNG file.

Final Requirements and Comparison with Initial Requirements

Initial Requirements	Final Requirements
Screen Sharing 1) As a user, I can access a shared screen. 2) As a host, I can share my own screen. 3) As a host I can disable my own screen share. 4) As a host, I can disable another user's screen share.	**No Screen Sharing**
Draw on Screen 1) As a host, I can draw on the screen. 2) As a host, I can grant drawing permissions. 3) As a host, I can revoke drawing permissions. 4) As a user with permission, I can draw on the host's shared screen.	**Draw on Screen** 1) As a host, I can draw on the screen. 2) As a host, I can grant drawing permissions. 3) As a host, I can revoke drawing permissions. 4) As a user with permission, I can draw on the host's shared whitebaord.

Host a Session	**Host a Session**
1) As a host, I can grant permissions to guest users. 2) As a host, I can revoke permissions from guest users. 3) As a host, I can end a session. 4) As a user, I can host a shared session.	1) As a host, I can grant permissions to guest users. 2) As a host, I can revoke permissions from guest users. 3) As a host, I can end a session. 4) As a user, I can host a shared session.
Join a Session	**Join a Session**
1) As a user, I can join a session. 2) As a user, I can leave a session.	1) As a user, I can join a session. 2) As a user, I can leave a session.
Add Text Notes	**Whiteboard Controls**
1) As a user, I can put text notes on my screen. 2) As a user, I can remove my text notes.	1) As a user, I can choose the color of my drawing. 2) As a user, I can undo a drawing. 3) As a user, I can redo a previously undone drawing. 4) As a user, I can clear all of my drawings.

Final Timeline and Comparison with Initial Timeline

Here is our Initial Timeline:

Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6
1/29 - 2/12	2/12 - 2/26	2/26 - 3/12	3/12 - 3/26	3/26 - 4/9	4/9 - 4/23
Base App, Screen Capture, Screen Drawing Presentation 1	Session Hosting, Session Joining	Host control bar, Text Notes, Host perms Presentation 2	User text chat, Client nicknames	User accounts, Database	Safety Sprint :) Final Presentation

Here is our Final Timeline:

Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6
1/29 - 2/12	2/12 - 2/26	2/26 - 3/12	3/12 - 3/26	3/26 - 4/9	4/9 - 4/23
Base App, Presentation 1	Base App, Technical Documentation	Session Hosting, Session Joining, Presentation 2 Group Report 1	Group Report 2, AWS instance creation	Group Report 3, Group Report 4, Troubleshooting AWS, drawing tools finished	Multi-room functionality, GUI updates, Offline drawing, SSH sockets

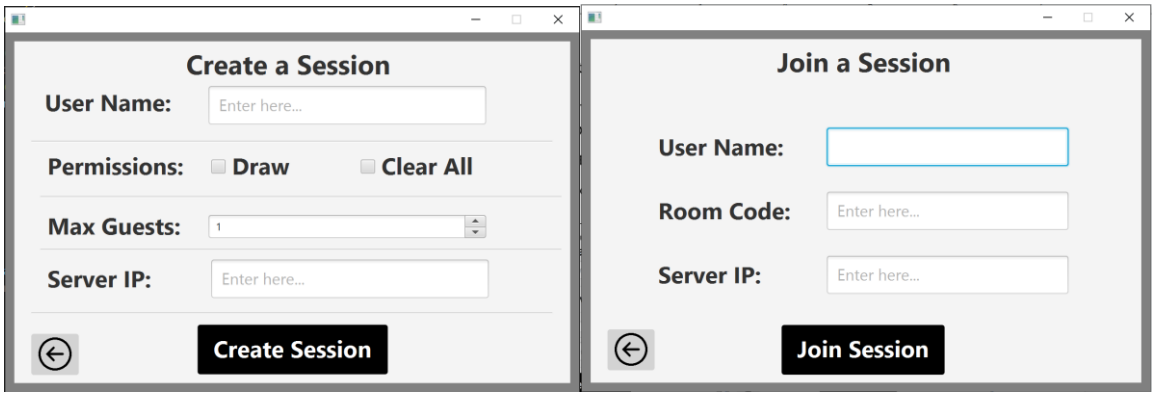
Project Results Compared with Expectations

a) Initial Use Cases that are Functional:

i) Draw on Screen

ii) Host a Session

iii) Join a Session



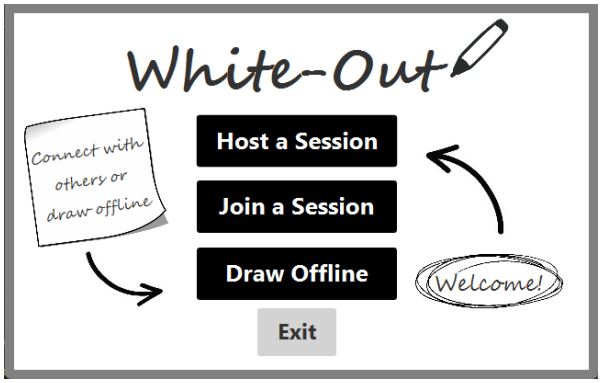
b) Removed/Added/Modified Use Cases:

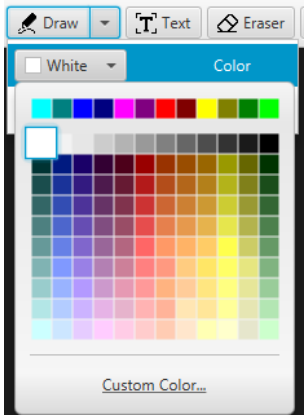
i) Removed:

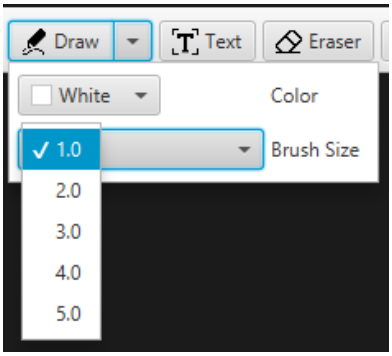
1) Screen Sharing: We realized that the implementation of screen sharing would be very difficult with the time frame that we had, so we decided to remove this use case.

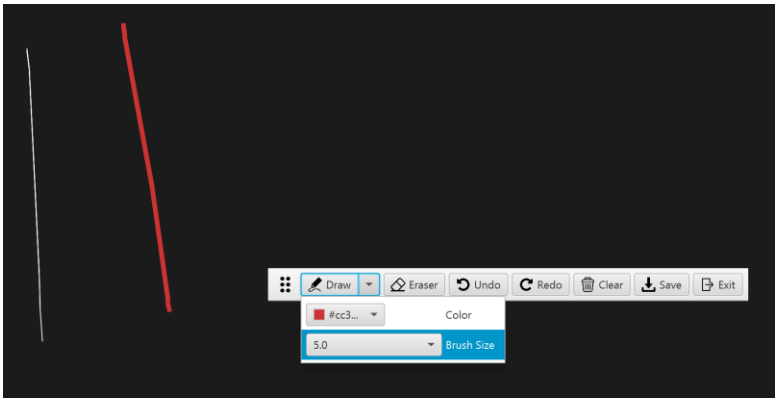
ii) Added:

1) Offline Drawing: We wanted to create an option for users to begin an offline whiteboard if they did not want to create a room.
2) Whiteboard Controls: We wanted the whiteboards to have editable controls, such as color choices, undo/redo capabilities, clear, etc.









iii) Modified:

The initial use cases have not been changed, so there are no modified use cases.

Software Evaluation

a) Functionality:

i) We have done our best to follow the Red, Green, Refactor testing methodology as we developed our project.

ii) Manual testing of the drawing and server-client socket connection.

iii) Connection testing of our AWS instance through the AWS console.

iv) A small number of identified functionality issues that were identified throughout development are still open.

1) On occasion, clients will have another client's drawing appended to their stroke if another client submits a stroke while the user is actively drawing.
2) On occasion, dotting the screen rather than creating a stroke can submit null characters that will break the connection.
3) On occasion, a client will be unable to see their own drawings on their screen. These drawings are still handled properly by the server and are seen by other connected clients. This issue is remediated by restarting the client.
4) Currently, canvas resolution is not adaptive. It is possible for a client to draw along the border of a higher resolution screen and

have their drawings not populate on another client's screen if that client's resolution is smaller.

b) Security:

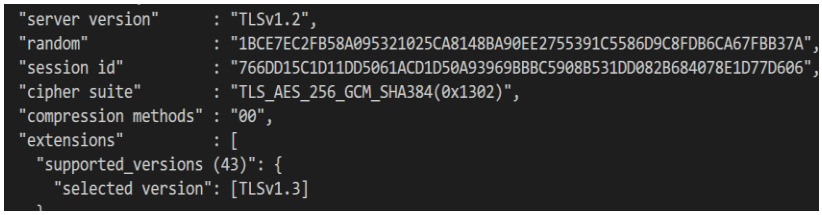
i) TLS v1.3

1) Initial handshake request is made at TLS1.2, which is unsecure, however, the clients are enabled for TLS 1.3 and this protocol is activated upon client-server handshake.

2) Dual AES and SHA encryption is utilized for the socket connection

3) Public Certificates are designed using SHA 384 with RSA.

4) Asymmetric key based encryption proves to be secure and quick on a Socket layer connection.



ii) The current SSL system makes deployment complex due to the utilization of a self signed certificate. Obtaining a cert signed by a certificate authority would solve this, however most certificate authorities require the utilization of a web server and http based protocol.

iii) Regex based input constraints are implemented client side. The server side needs similar sanitization to ensure a modified client does not bypass security measures.

1) This provides the greatest security risk.

2) Code Injection via a modified client could potentially create a security risk for other clients of a relevant room.

3) Server and clients both utilize a ‘default clause’ based escape that terminates the socket connection in the case of an unexpected or recognized command. This mitigates the threat of code injection.

iv) A version of the client and server are available without TLS protection. The data transmitted from these versions is readable as cleartext.

1) No databases or user accounts are implemented in this project. The lack of requested PII mitigates this risk. Cleartext leaks are limited to:
    (a) Session nickname
    (b) Room size
    (c) current permissions
    (d) room permissions
    (e) room code
    (f) brush color
    (g) brush size
    (h) stroke coordinates
    (i) command category (i.e. undo, redo, clear, clear all, draw, change permissions, kick, join room, host room)

Work to be Done

- A system needs to be built to identify a bad connection during the initial handshake, right now the client just connects to a faulty whiteboard if the connection fails.
- Implement the ability for the server to escape dangerous messages in message strings.

- Improve SSL connection system for final deployment. Currently using self-signed certificates.
- Client side bug involving multiple thread usage of the Graphics Context controls needs to be resolved. A resource prioritization protocol similar to collision avoidance in wireless communications could be implemented to correct this.
- Previously intended text feature was not able to be implemented in this scope, GUI needs to reflect that it is no longer included in the intended release period.
- Implement a deployment system for easily downloading and running the application on a new system.
- Eraser functionality is not yet implemented server side. This requires an addition to the protocol and was considered out of scope considering the time frame of the project.