DrawBuddy

B2: Lisa Mishra, Ronald Gonzalez, Denise Yang
18-500 Capstone Design, Spring 2022
Electrical and Computer Engineering Department
Carnegie Mellon University

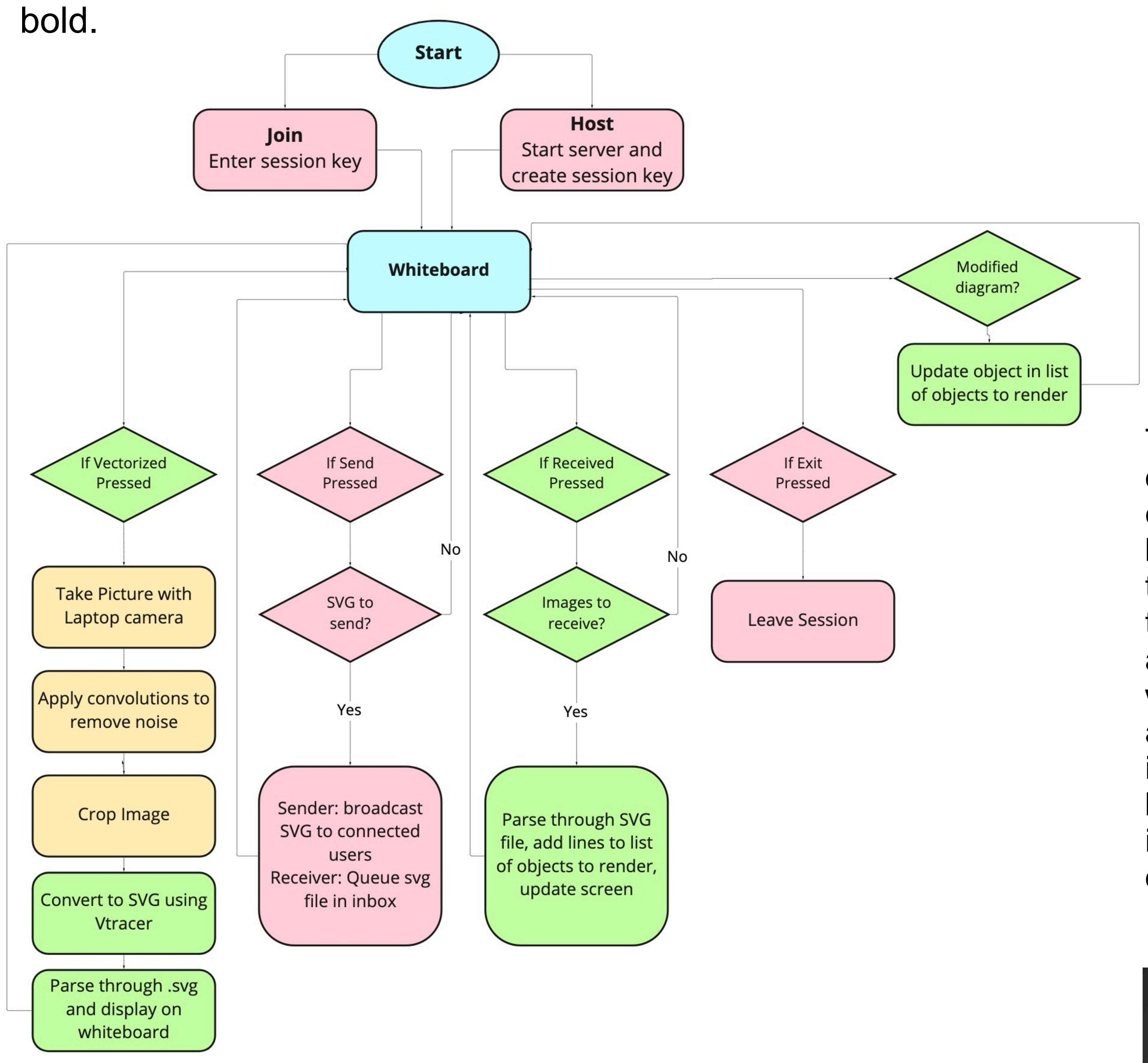
Product Pitch

While virtual platforms like Zoom are useful tools for remote lectures they are not as adept for group work that involve drawing diagrams especially with only a mouse/mousepad. Cue DrawBuddy, a more accessible and natural web app that can convert black and white hand drawn images to SVG files that cam be displayed, modified and sent to peers in a shared session.

DrawBuddy should be able to render diagrams drawn with black ink with line thicknesses of 0.4mm-1mm on white letter sized paper held anywhere from 1-3 feet from the camera **within 120s**. Any modifications and broadcasts should be reflected within **100ms and 500ms respectively**. We aimed for users to rate the rendered image **9/10** with 10 being the highest. Our system is able to **convert diagrams in 100 ms**, **reflect updates in 15ms** and has varied results with .svg broadcasts. **Users rated our system 5.3/10.**

System Architecture

Figure 1. **DrawBuddy Flowchart:** Our architecture can be broken down into image processing (yellow), converting to and from .svg files (green), and socket communication (red). There are 4 main states: Start, Join, Host, and Whiteboard indicated in bold.


Start
Join
Enter session key
Host
Start server and create session key
Whiteboard
Modified diagram?
Update object in list of objects to render
If Vectorized Pressed
If Send Pressed
If Received Pressed
If Exit Pressed
No
No
Take Picture with Laptop camera
SVG to send?
Images to receive?
Leave Session
Yes
Yes
Apply convolutions to remove noise
Crop Image
Sender: broadcast SVG to connected users
Receiver: Queue svg file in inbox
Parse through SVG file, add lines to list of objects to render, update screen
Convert to SVG using Vtracer
Parse through .svg and display on whiteboard

System Description

Drawbuddy relies on the laptop camera to capture the image and then performs image processing (reducing noise, detecting lines, masking/ cropping image) via OpenCV. DrawBuddy will convert the initial .jpeg into a .svg via Vtracer. Then our system will extract the objects in the .svg file with a parser that we wrote in Python and displayed on a web app made with PySimpleGui. Diagrams can be modified and then later exported as .svg files using our parser.

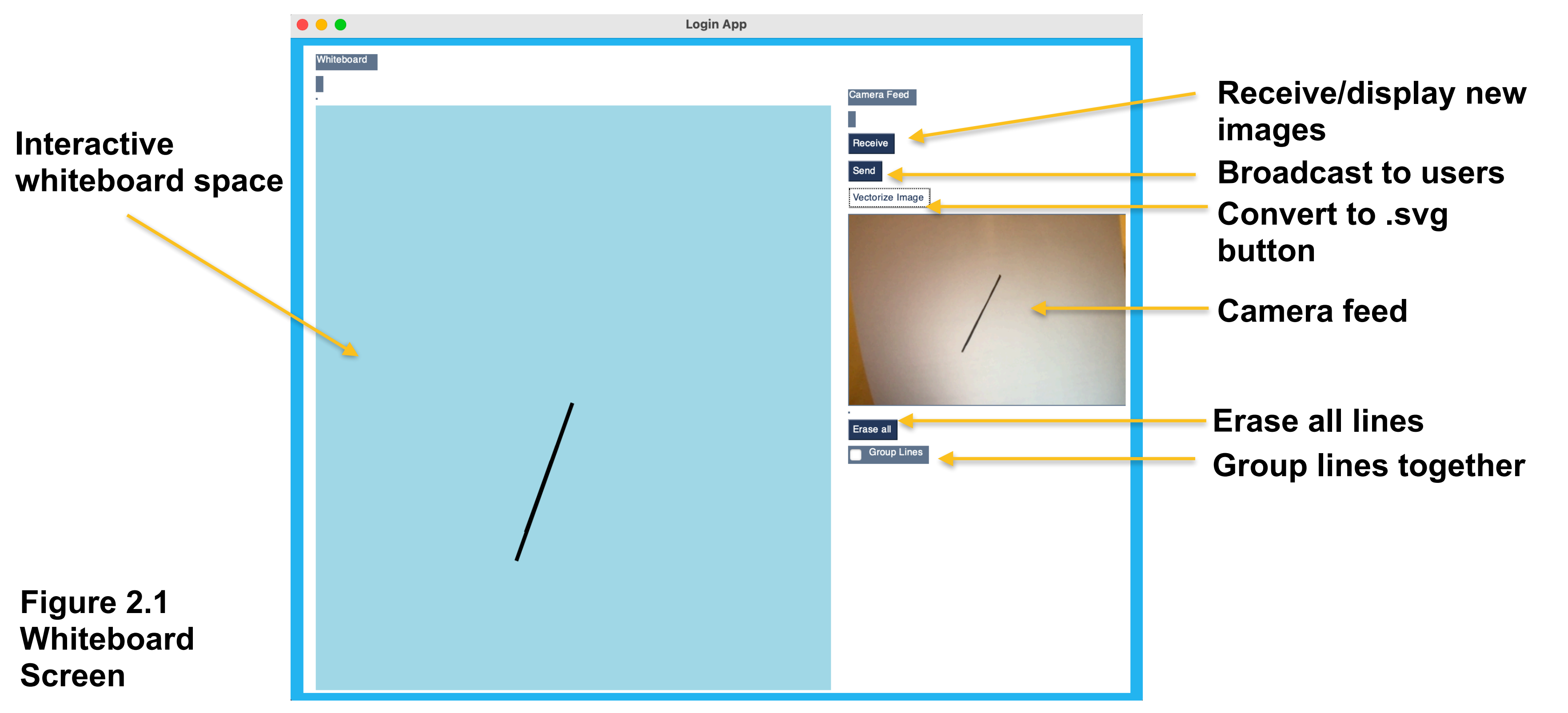


Figure 2.1 Whiteboard Screen

To create and join sessions DrawBuddy uses PySocket to start servers and register users as clients. .svg files can be sent over these sockets.

Figure 2.2 Welcome Screen

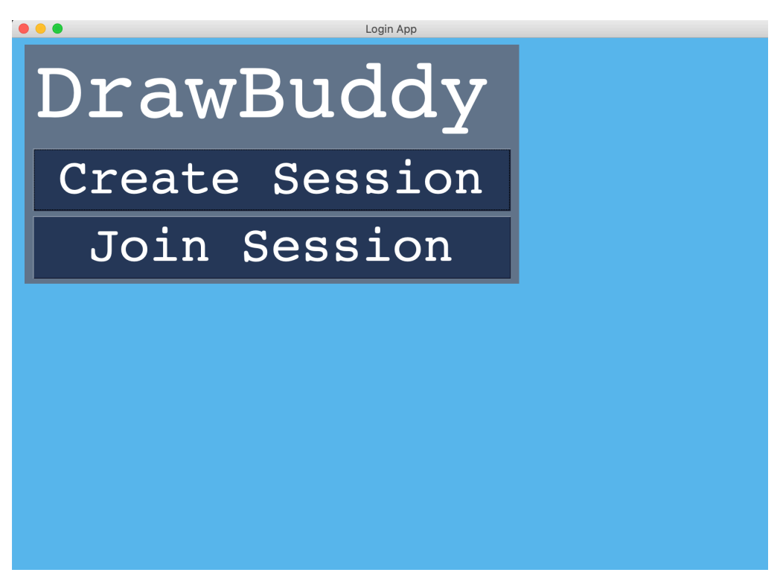


Figure 2.3 Start Server Screen

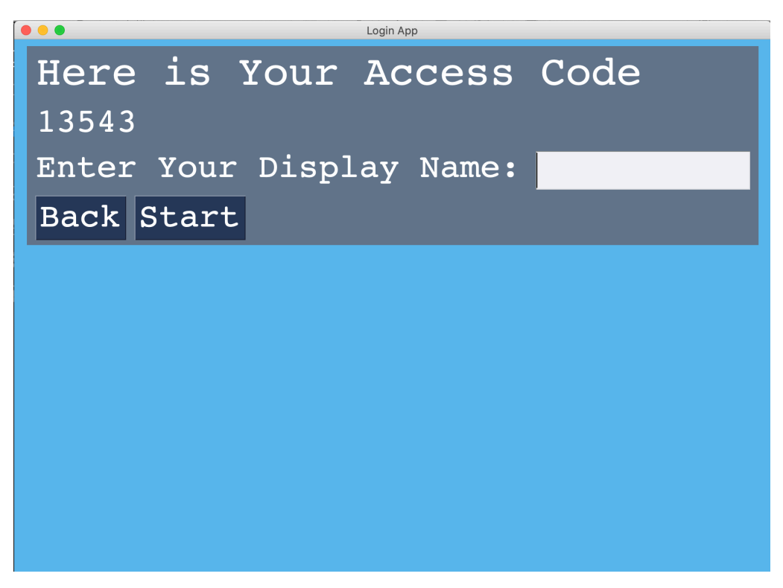
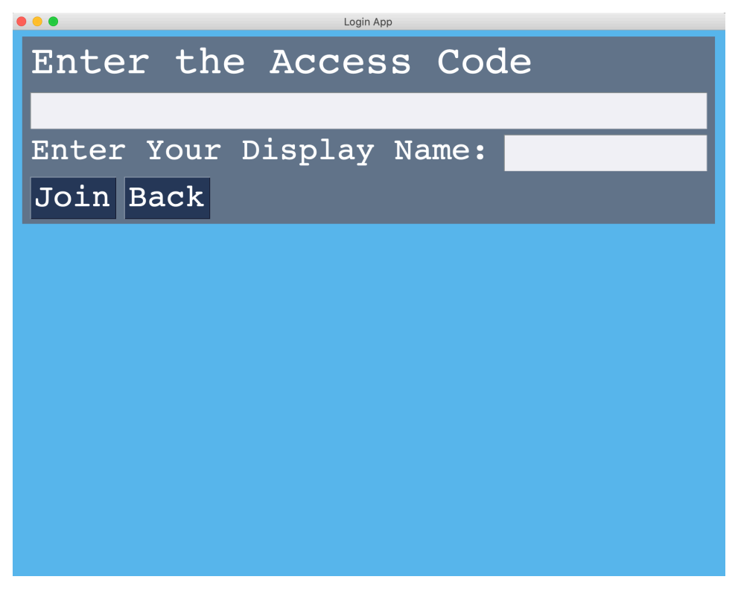


Figure 2.4Join Screen



System Evaluation

To quantify the quality of our diagrams we compared the number of lines rendered with the true number of lines. We tested with 1, 5, 10, 20, and 50 lines. Ideally we would see a 1:1 ratio, but as the number of lines increase, we render less lines. As a result of these inaccuracies we fell short of our user rating goal.

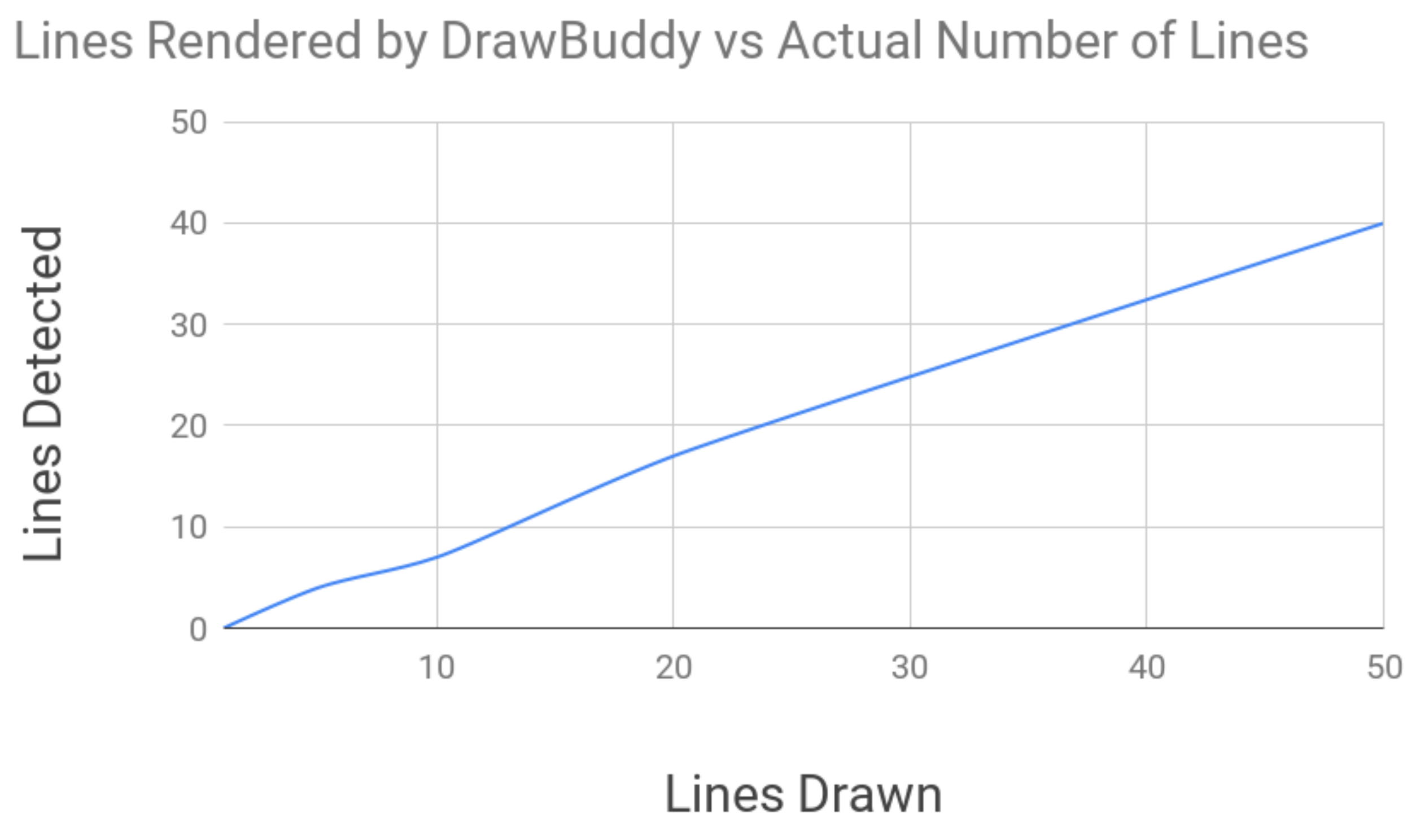


Figure 3.1 Comparison of number of lines rendered vs lines drawn. Our behavior is relatively linear however as we begin to lose quality as the amount of lines drawn increases

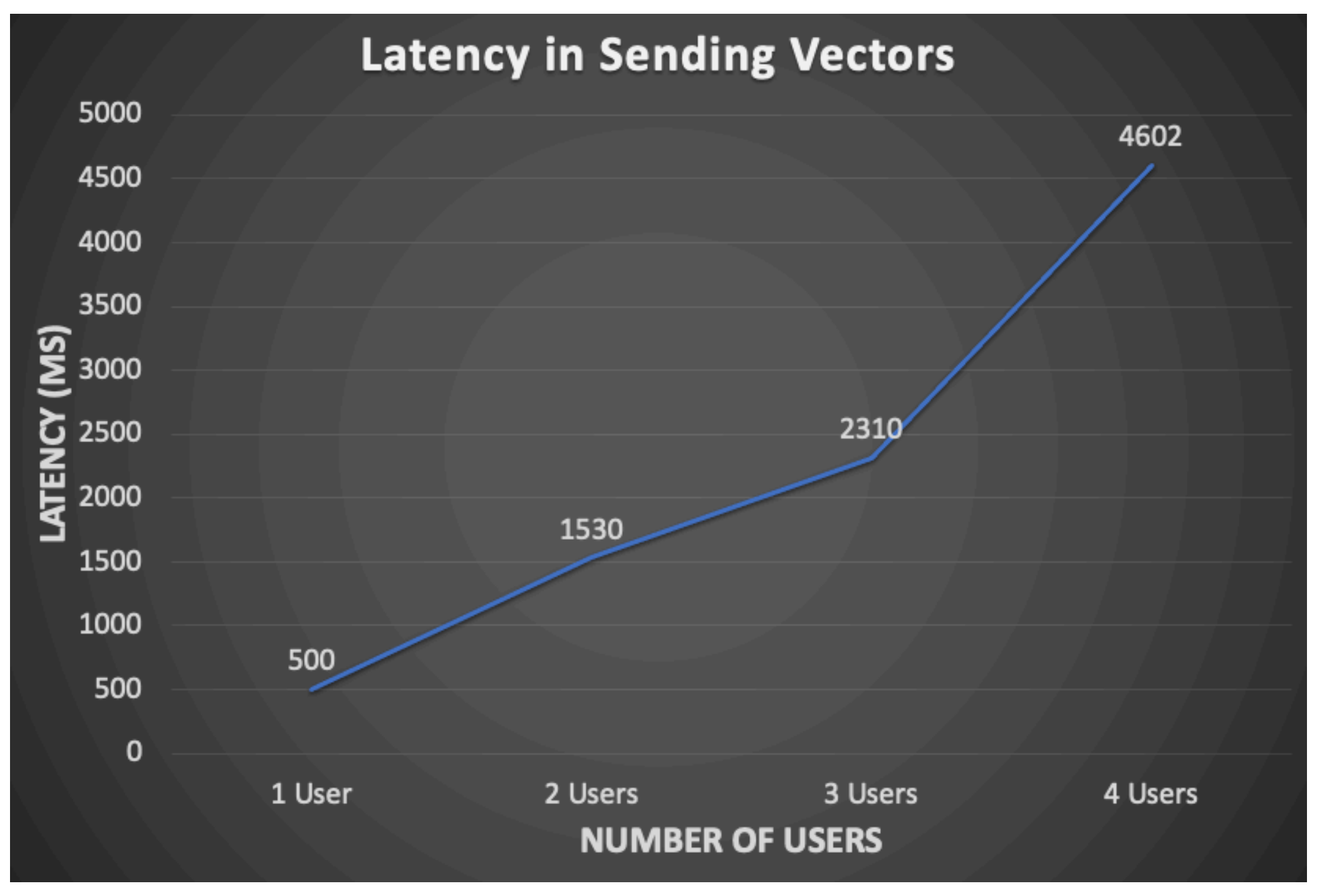


Figure 3.2 Comparison of sending 30 vectors latency to varied number of users.

To test our communication use case we measured the time it takes to send a compressed and encoded .svg file with 30 vectors from one user to a varied number of users. The latency increases somewhat exponentially as the number of users increase.

Conclusions & Additional Information

Find out more:

http://course.ece.cmu.edu/~ece500/projects/s22-teamb2/

We implemented our MVP of converting hand drawn lines into an .svg file that can be moved around the screen and resized/rotated before being sent to peers. Although the quality of the .svg rendering of the diagram can be improved with algorithms to remove similar lines that overlap, and line detection algorithms. Overall DrawBuddy taught us the importance of collaboration and communication not only as we considered how to quantify improvements to collaboration but also during the integration of our individual parts.

Electrical & Computer ENGINEERING
Carnegie Mellon