iMessage Whiteboard

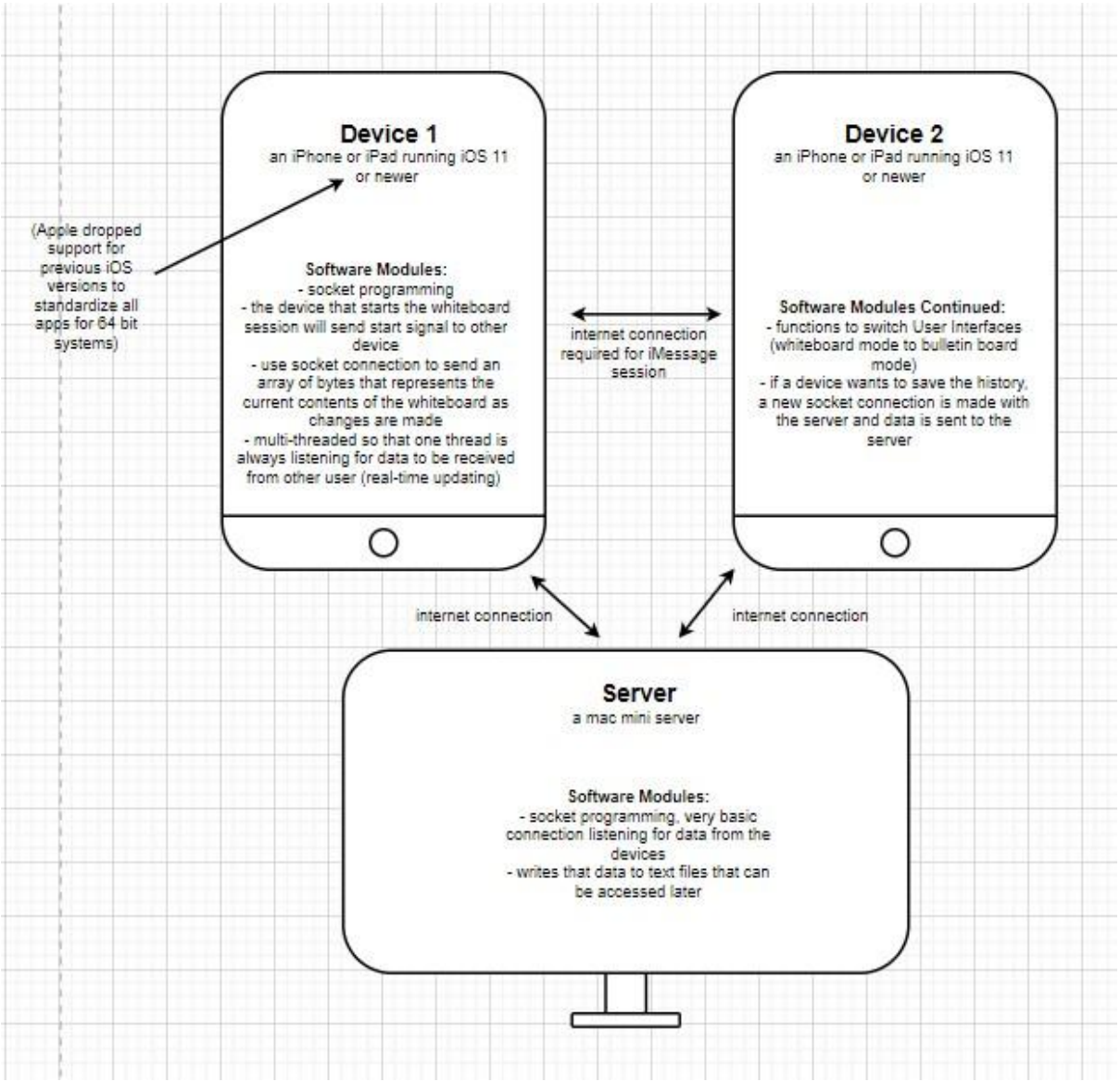
Cassidy Johnson
c_johnson49@u.pacific.edu
Github URL
Last revised 02/07/2021

System Architecture

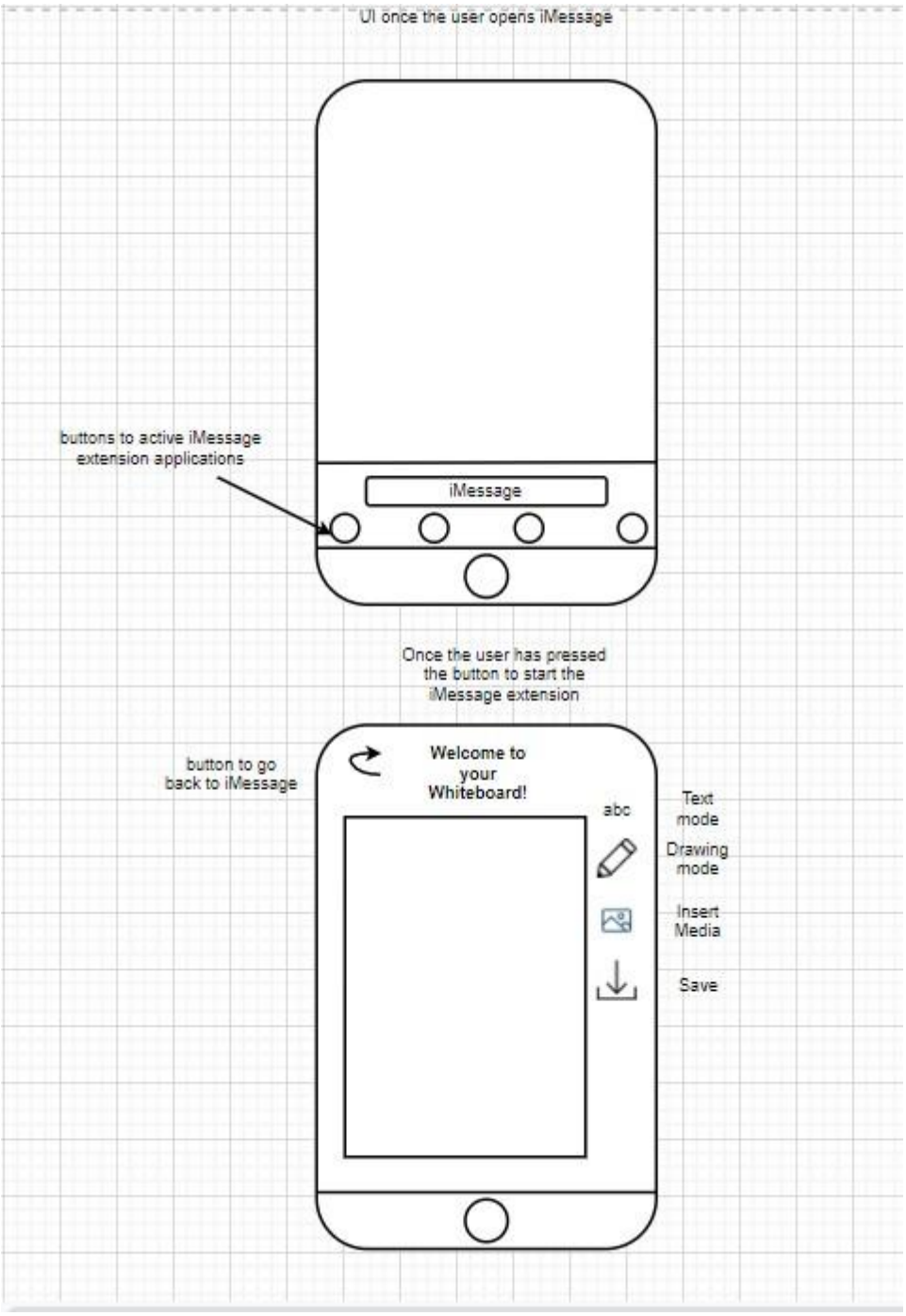
As you can see in the attached system architecture image, there are two users in an iMessage chat and one server that I have set up on a Mac Mini Server running MacOS Big Sur. This server has the XCode IDE set up to run the Swift code that will run on the server side.

This was originally designed to be largely a peer to peer application, but I've instead made it a client server architecture. This is because the iMessage chat session restricts and protects user's information such as their IPs. They only give us a UUID. So, I store a hash of UUIDs to IPs in the server side and use the server to give messages from one client to another.

The User interface is shown in the second photo. There are no interfaces to external systems other than the client-server architecture shown below.


Device 1
an iPhone or iPad running iOS 11 or newer
(Apple dropped support for previous iOS versions to standardize all apps for 64 bit systems)
Software Modules:
- socket programming
- the device that starts the whiteboard session will send start signal to other device
- use socket connection to send an array of bytes that represents the current contents of the whiteboard as changes are made
- multi-threaded so that one thread is always listening for data to be received from other user (real-time updating)
internet connection required for iMessage session
Device 2
an iPhone or iPad running iOS 11 or newer
Software Modules Continued:
- functions to switch User Interfaces (whiteboard mode to bulletin board mode)
- if a device wants to save the history, a new socket connection is made with the server and data is sent to the server
internet connection
internet connection
Server
a mac mini server
Software Modules:
- socket programming, very basic connection listening for data from the devices
- writes that data to text files that can be accessed later

User Interface



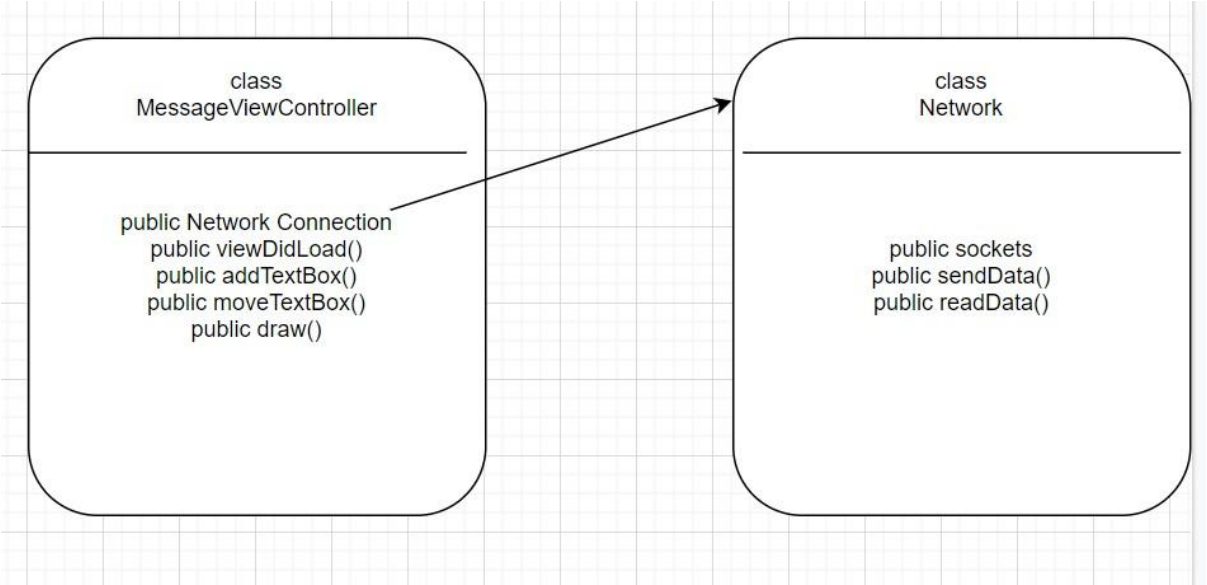
Hardware and System Requirements

Server:
The server must be running MacOS and should have a reasonable hardware configuration. “Reasonable” is defined as at least 256GB harddrive, at least 4 core CPU, and no GPU is needed. The Server requires at least one open TCP port that is not taken by another application.

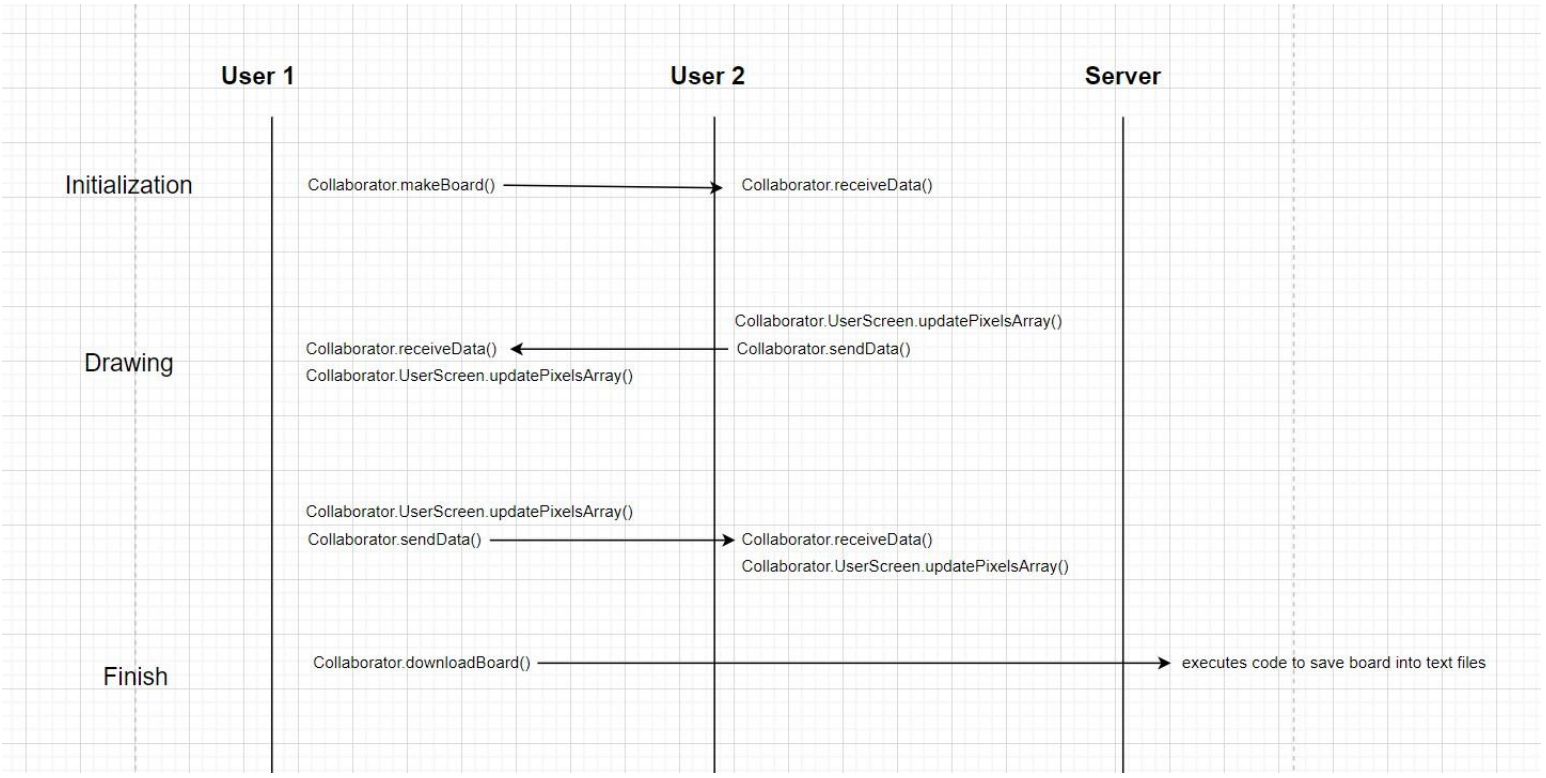
Client (the user’s device):
The user’s device should be an iPhone or iPad running at least iOS 11. Each device needs at least two (one for the peer to peer connection and one to connect to the server for saving) open TCP ports that are not taken by another application.

Software Design

Class Diagram and Class Specifications:



Interaction Diagram:



Design Considerations:

1. Accessibility: I don’t want users to have to open another application to use this. I want it to be native to the iMessage app.
2. Speed: I made the system peer to peer because I’m hopeful it will reduce lag when the whiteboard is updating. I’d like it to be as real-time as possible.

Glossary

- Server: the code running on a separate machine that performs some functionality that should not be done on the user's machine
  - in our case, the server runs code to store the whiteboard data more permanently
  - also: the physical machine that runs that code
- Client: the device of a user
- User: a person who will download the iMessage whiteboard extension and create whiteboards on it
- Socket: a basic TCP connection between two devices using an IPv4 address and a port number