# Whiteboard Development Guide

# Installation

It is required to have python 3.0 or higher installed to run this software

To use the software just decompress the winrar file and start the Server.py (or have somebody else start the Server.py) afterwards just run the client.py. When starting the client code one must input the IP and Port number, the IP must be set to the IP of the computer which is currently running the server.py and the port is normally set to 5000 (unless the who is running the Server changes it in the code).

# Main Functionality

This is a collaborative whiteboard that works based on having an active internet connection for all parties.

The whiteboard works like this. Once a user has selected a drawing widget (rectangle, line, oval , etc) it clicks the board and we register it’s initial point clicked, when the user releases its mouse we register the point the mouse was released and use that to construct a message. The message starts as a tupple of the format (A,B,C,D,E,F) where A is the type of the message, B, C, D and E are coordinates F is the color etc. Afterwards the class network transforms this tuple in a message of the type b”A B C D E F Ø “ and sends it though the network. The Ø character is an indication of end of message.

When the message reaches the server, the server dissects the parts of the message based on the spaces between the characters. The first part is the type, which indicate what type of message it is. This allows the server to handle the message properly, since we keep logs for only messages of the type that draw an object, but for delete messages we remove the object from the server log. After the message has been properly dealt with, the server echoes the message for all the members in the network.

Once a message has reached a client, it interprets its type and acts accordingly. For instance, if it’s message indicating to draw an object, it draws an object. If it is a permission message it receives a permission.

Here are some of the types of messages and their structure:

Examples:

* **['L', '505', '72', '675', '284', 'b', 'lol', 'm1'] =** Draw a line from (505,72) to (675,284), with blue color, draw by user named ‘lol’ and tag it as message 1
* **['DR', 'm0', '118', '206'] =** Drag object tagged as ‘m0’ by 118 and 206 pixels
* **['Z', 'm0'] =** Erase object tagged as ‘m0”

## 

## Message type translation

### Drawing messages

* “L” = Line
* “R” = Rectangle
* “O” = Oval
* “T” = Text
* “D” = Pencil Draw
* “C” = Circle
* “S” = Square

### Deleting messages

* “Z” = Delete item
* “E” = Erase all

### Drag message

* “DR” = Drag

### Talk Message

* “TA” = Talk

From these types of messages only the drawing messages generate logs in the server and in the clients. Deleting messages remove the logs from the server and drag messages move the location in the logs as well. Talk messages are not keeping logs as well, so if a user enters the chat it can only see what has been drawn before he enters, but not the things that have been said in the chat before his entrance.

# Classes

## Main functionalities

This part contains a summary of the classes and their main functionalities

**Client** = Responsible for tracking the user movements (mouse movement and mouse click) and assigning the different types of message to the different classes that handle it. Because of this, it is also responsible for constructing the messages of drawings that will be sent

**Drawing Tools** = Responsible for drawing the messages on the board, it contains all the drawing functions (draw rectangle, draw line etc) and it is responsible for identifying from the message which drawing function to use based on the first part of the message

**Whiteboard** = Responsible for placing all the buttons and implementing their functionalities, which means grating the user the ability to switch between drawing tools, colors, call the save function. Here we also have the implementation of the eraseAll function which is a function that erases everything that the user has a permission to erase. It also creates the permission related buttons.

**Permissions** = This class is responsible for managing the permissions. It interprets and sends permission type messages (relating to giving and receiving permission from users). It keeps track of the list of users which you gave permission to and the users which gave you permission to delete their stuff. The class whiteboard will use this data to construct the buttons

**MConnection** = This is the class responsible for managing the connection. Which means connecting to the server, receiving and sending all the messages. All the messages send pass through processing in this class where they will be turned from a tuple format into a format appropriate to send.

**Server** = The server file contains several classes. The server class in this program is only responsible for listening to new connections and establishing them once they arrived. After they arrived, we create an instance of the **Client** class written on the server. This instance is responsible for interpreting messages, adding them to the server logs and echoing the messages to all clients connected in the server. Finally, we have a **Pinger** class, this class sends periodic bytes to check if the connection between server and users, now the pinger fails we know a user has been disconnected and we can act upon it

## Extra functionalities

**Save and Load** = This class is responsible for saving the state of the board and restoring it using the load function. It was programmed to only redraw the things that were previously draw and not to restore it completely to the original state (if there is something draw on the board it will remain after the load has been used). This was intentional, since now all the users must agree to erase the board by themselves if they wish to previously restore it to its previous state.

**Messenger** = This class is responsible for creating a chat between the users using Tkinter text feature. This allows the user to communicate within themselves when using the software

## 