Network Project

P2P Interactive Whiteboard

Mohamed moghazy 19100650

Amr Ahmed Hussein 19101042

Ibrahim Mostafa 19103851

Ayten Sherif 19100260

The following code implements a Peer-to-Peer (P2P) Whiteboard application using Python and the Tkinter library for the creation of the graphical user interface (GUI). The application allows multiple users to connect and draw on a shared whiteboard. Each user can do anything on his whiteboard, and that thing is synchronized among all connected clients. The application follows a client-server architecture, where one user acts as the server and others connect as clients to share and update the things they are doing.

The frameworks and protocols that we used in this project are:

Tkinter : for the gui including the canvas and the buttons .

Socket: the socket module is used for establishing network communication between the client and the server.and also it offers the essential TCP/IP functions for sending and receiving data over a network.

Threading : used to handle concurrent operations In the application. It also allows the server to listen for incoming connections while the gui remains responsive.

Struckt: is used for packing and unpacking binary data . it is used to convert the drawing coordinates into a structured byte format for sending over network.

Functions and classes

First , the whiteboard class:

The white board class represents the main application and encapsulates the functionality of the whiteboard

Init(self) :the constructor method that initializes the gui components . It asks the user to enter a username and uses the start\_server method to launch the server in a separate thread.

accept\_connections: Runs in a separate thread and accepts incoming client connections. For each new client, it creates a separate thread to handle client's messages.

handle\_client: it Runs in a separate thread for the connected clients. It continuously reads messages from the client's input stream and then process them.

draw\_on\_canvas: Callback function bound to the "" event of the canvas. It draws an oval shape on the canvas at the specified coordinates and sends the draw message to all connected clients.

draw\_on\_canvas\_remote: Draws an oval shape on the canvas at the specified coordinates received from a remote client.

send\_draw\_message: Sends the draw message to all connected clients by moving over the output streams and writing the message to each stream.

send\_draw\_message\_remote: Sends the draw message to a specific remote client the second client in the output\_streams list.

send\_clear\_message: Sends a clear message to all connected clients by iterating over the output streams and writing the "CLEAR" message to each stream.

clear\_canvas: Clears the canvas by deleting all items on it and sends a clear message to all connected clients.

process\_message: Processes the received message. If it is a draw message, it extracts the action, x, and y coordinates and calls the draw\_on\_canvas\_remote method. If it is a clear message, it calls the clear\_canvas method.

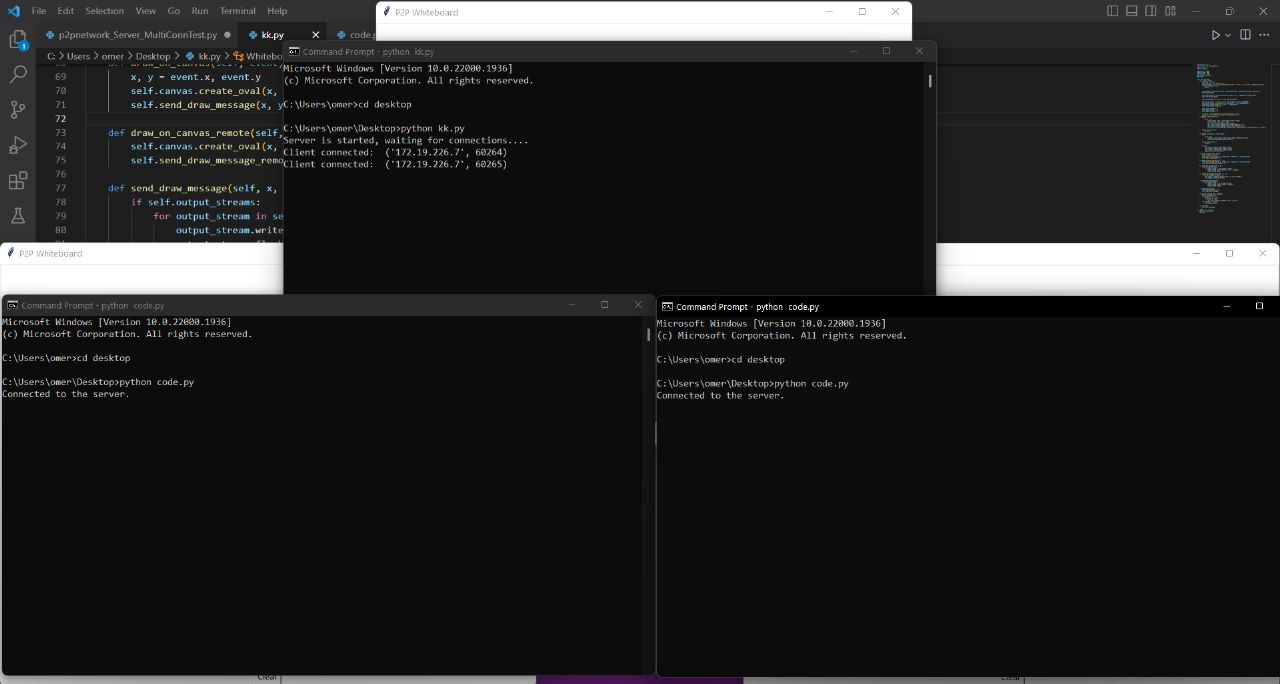
run: Starts the program.

More related functions and constants:

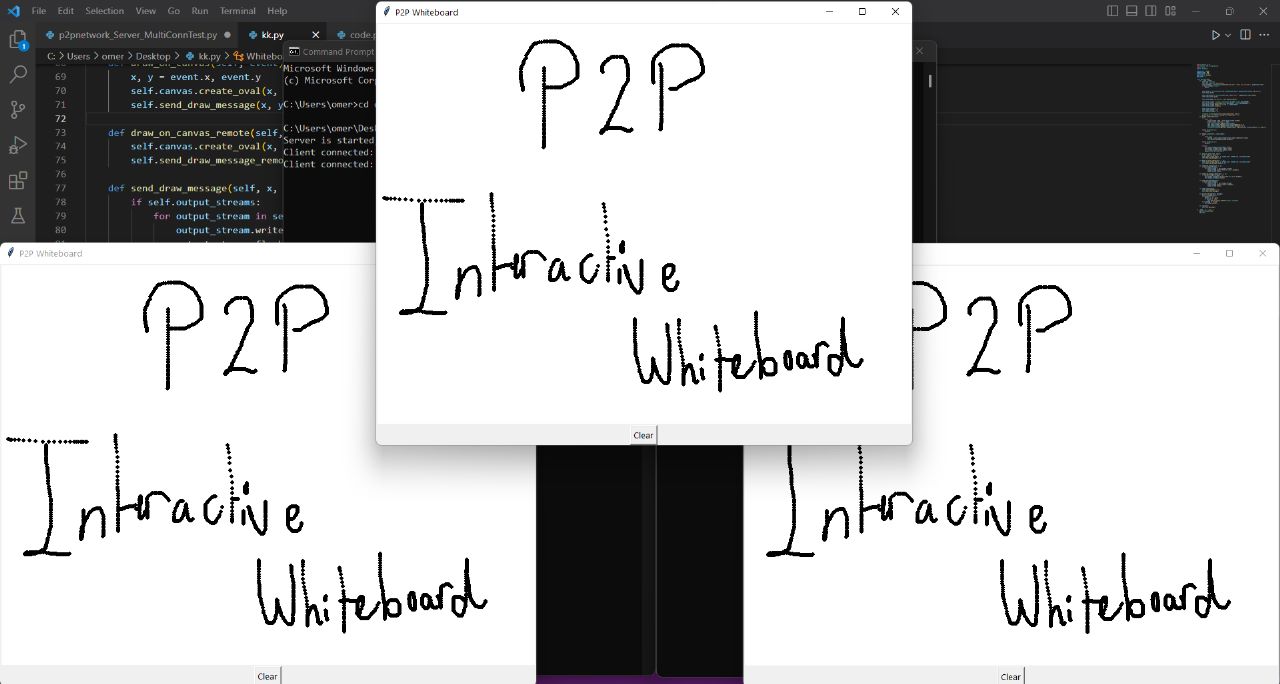
Server port:specifies the port number for the server socket.

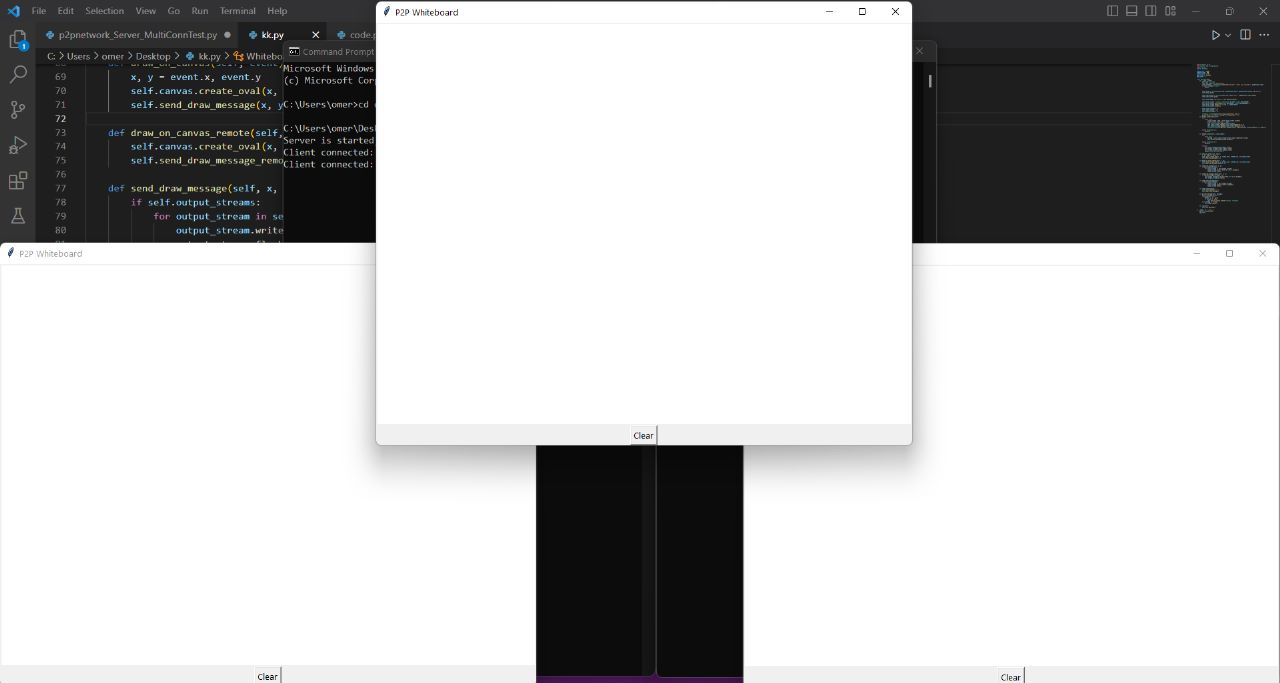
Cancas width , canvas height: constants that determine the size of the whiteboard .

The application also have various helper functions like event hanlder and utility functions for drawing, message processing and networking communication.



output





Tasks divided : ayten and ebrahim: handling gui + poster

Ibrahim and amr : server side

Amr and moghazy : client side

Moghazy: report

