COLLAB-CODE

Real-Time Collaborative coding with Chat

CS18B036 SRIPRANAV MANNEPALLI
CS18B032 S.R.V.S.MAHESWARA REDDY

ABSTRACT:

COLLAB-CODE's Coding Platform provides an Editor which enables users to code together in Real-Time. The Platform also has a chatting feature where users can chat in a group. The platform also shows the currently logged-in users. We used socket programming to develop the application.

INTRODUCTION:

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server. We used socket programming to develop a collaborative coding feature, group chatting feature, and related features.

DESIGN:

We have a client and a server. Multiple clients can connect to one server. The Server acts as a bridge between clients.

- Server Design:
    - Whenever the server receives a connection request for the first time from a client, the server sends a message asking for a username. The client then responds by sending its username, the server stores the client's username and creates a thread to handle requests from this client. Multiple clients can connect with the server in a similar manner. The server acts as a bridge between different clients and performs various functionalities.
    - When the server is run, then it binds to the localhost and a port and starts listening to the clients trying to connect.

- The server class has the following main methods:
  - Receive:
    - Through this method, the server listens to the incoming connections. When a new client connects to the server, the server stores the username and client instance and creates a thread for the client.
  - Handle:
    - Through this method, the server listens to the particular client on which it is called upon.
  - Broadcast:
    - Through this method, the server broadcasts a message to all the connected clients.

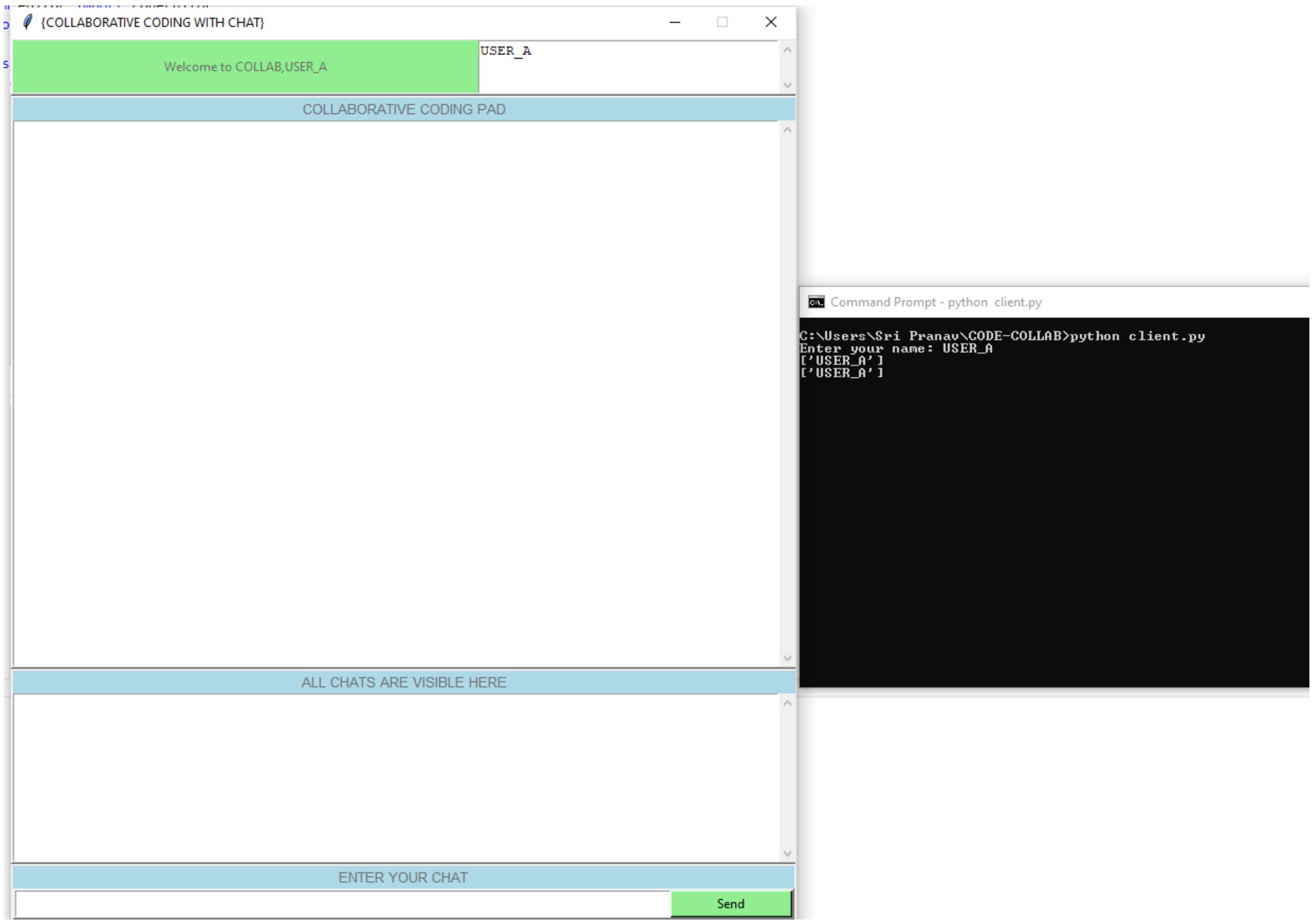
- Client Design:
  - Whenever a client instance is created, then we ask the client to enter the username. Then, this username is sent back to the server and the client makes a connection with the server. The client creates two threads to run the "Send" and "Receive" from the server.
  - The client will connect to the server via the localhost and the port number and starts communication.
  - The client class has the following main methods:
    - RecieveCodeAndChat:
      - Through this method, the client receives the messages from the server. Then, the client decodes the type of message (chat/code etc) and performs the required functionality.
    - Send:
      - Through this method, the client sends information to the server.
      -

FUNCTIONALITY:

We have the following features in this application that make use of socket programming.

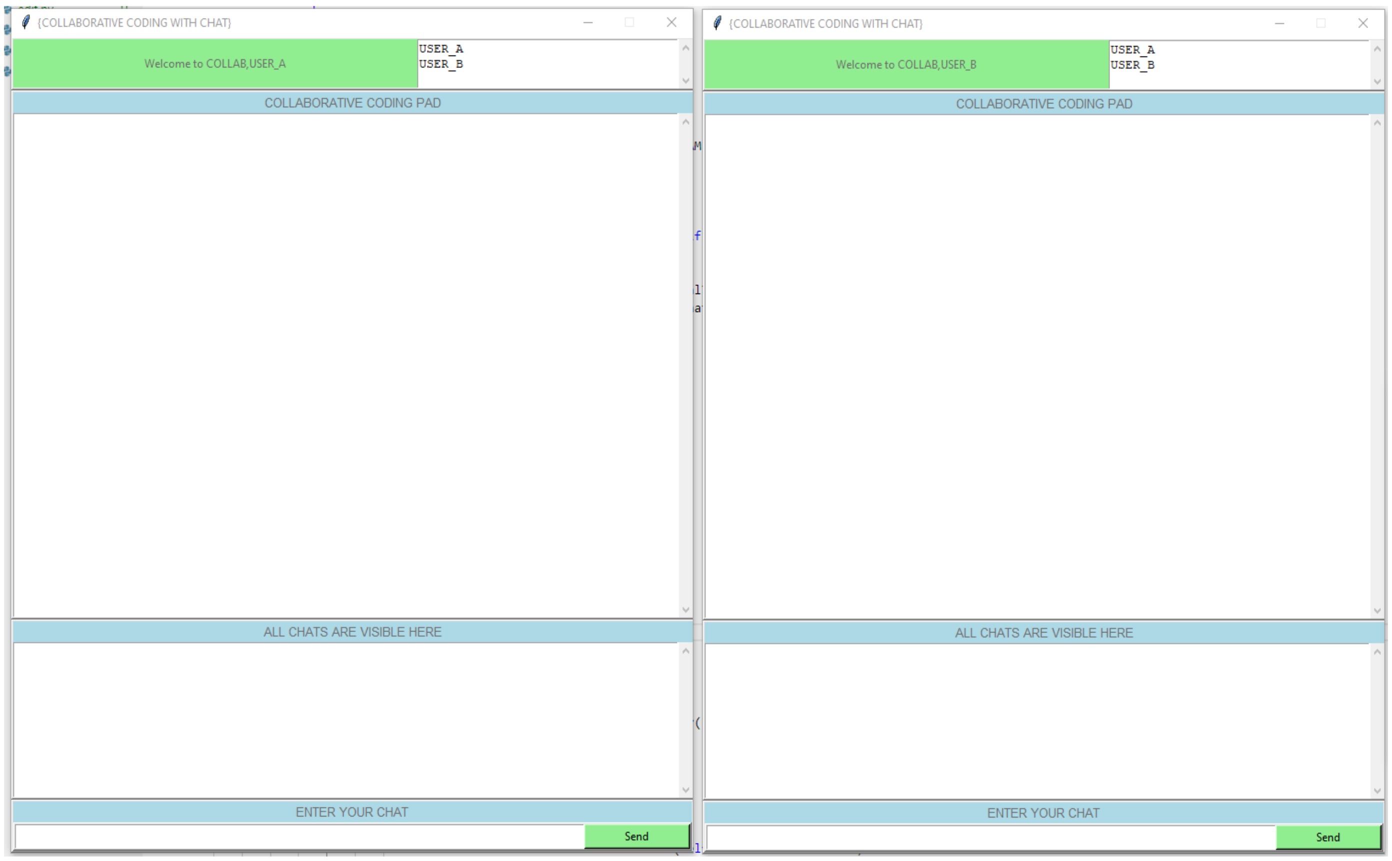
1) COLLABORATIVE CODING:

This feature allows multiple users (clients) to connect to the server and code together.
Let us consider an example to describe this functionality:
Let us say there are 2 clients “USER_A" and “USER_B ". So, when “USER_A" logs in initially, he needs to provide his username, which can be seen in figure A. Then, we can see that the connection is established with the server and a GUI Editor opens with only one user.


{COLLABORATIVE CODING WITH CHAT}
Welcome to COLLAB,USER_A
USER_A
COLLABORATIVE CODING PAD
ALL CHATS ARE VISIBLE HERE
ENTER YOUR CHAT
Send
Command Prompt - python client.py

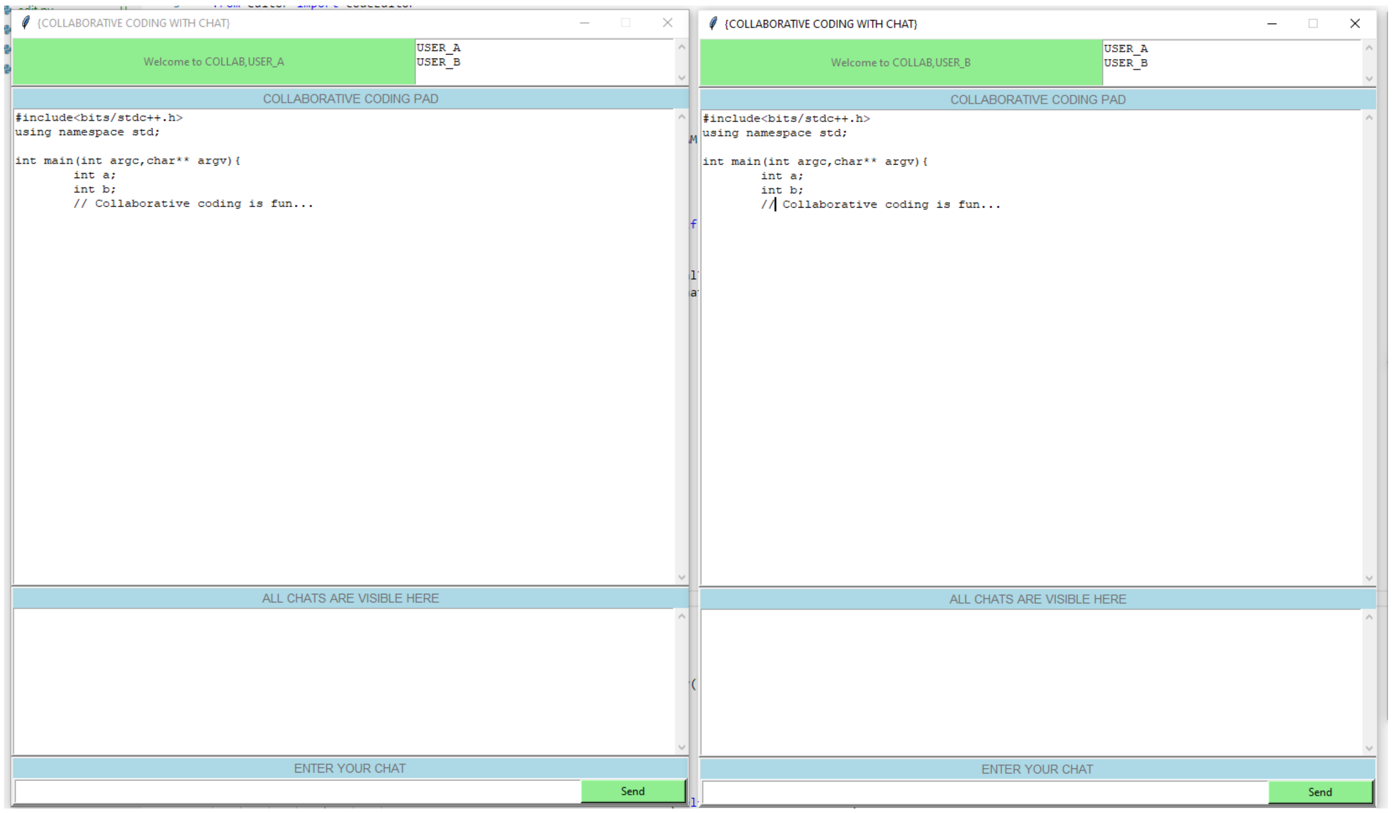
```
C:\Users\Sri Pranav\CODE-COLLAB>python client.py
Enter your name: USER_A
['USER_A']
['USER_A']
```

Now, When “USER_B" logs in, we can see both the usernames in both the client editors.



Now, If "USER_A" enters the word "#include<bits/stdc++.h> in his editor, then "USER_B" can see this line in his own editor. Now, if "USER_B" enters "using namespace std;" in his editor, the same appears in the "USER_A"'s code editor as well. In this way, users can collaborate in Real-Time using our application. The same is visible in the below figure.

2) GROUP CHATTING :

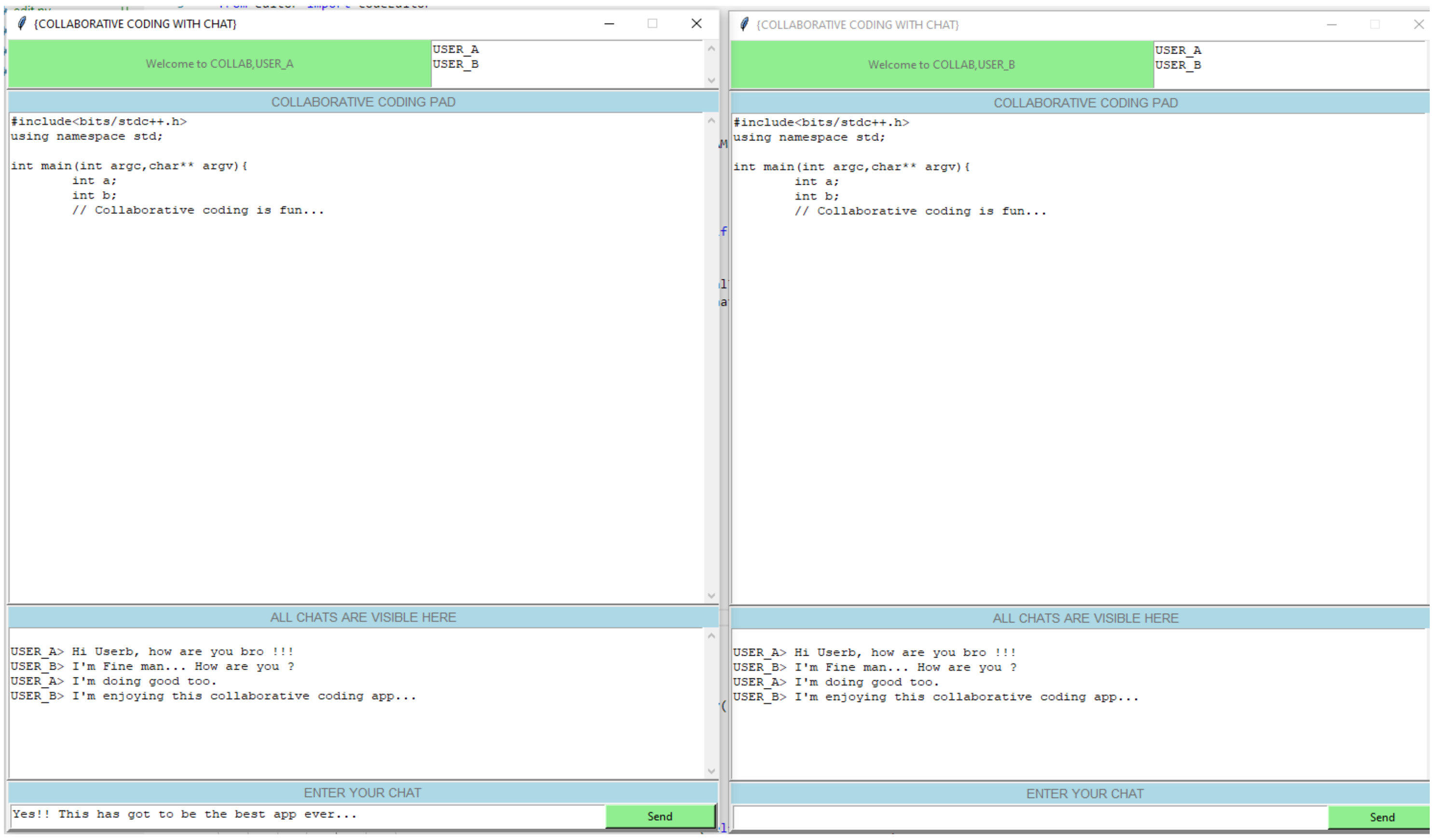


```
#include<bits/stdc++.h>
using namespace std;

int main(int argc,char** argv){
        int a;
        int b;
        // Collaborative coding is fun...
```

```
#include<bits/stdc++.h>
using namespace std;

int main(int argc,char** argv){
        int a;
        int b;
        // Collaborative coding is fun...
```

This feature allows multiple users (clients) to connect to the server and chat together while coding. Let's continue the previous example here. We can see the chat feature in action in the below figure. The user can enter his chat in the bottom box and press send. The chat will be sent to everyone connected on the server. All chats will be visible in the Chatbox. We can see the same in the figure below.

We also included error handling and we terminate a connection whenever an error occurs and show the client that the connection has been terminated.



```
#include<bits/stdc++.h>
using namespace std;

int main(int argc,char** argv){
        int a;
        int b;
        // Collaborative coding is fun...
```

```
USER_A> Hi Userb, how are you bro !!!
USER_B> I'm Fine man... How are you ?
USER_A> I'm doing good too.
USER_B> I'm enjoying this collaborative coding app...
```

```
Yes!! This has got to be the best app ever...
```

TECHNOLOGIES USED :

- Python
- Tkinter

CONCEPTS USED

- Computer Networks, Socket Programming
- Object-Oriented Programming, Software Engineering

CONCLUSION:

We have built a GUI application which we then integrated with the server and client functions which were in turn built using socket programming. This application helps users to collaborate in writing code in real-time situations. We also display all the collaborators and provide a chat feature to provide a communication channel between them.

LEARNINGS:

We have learnt socket programming and saw the application of networking concepts and also GUI programming. We also learnt about building complex softwares.

CONTRIBUTIONS:

We collaborated and contributed equally in all areas so that each of us can maximise our learnings in the topic.

Sripranav Mannepalli: Python Socket programing of Server and Client (receive) and Tkinter GUI building elements and layouts.

Ravi Maheswara Reddy: Python Socket programing of Server and Client (send) and Tkinter GUI building elements and styling.