CS310
ASSIGNMENT 1

ID	Name	Lab	Campus
S11198967	Karan Parmar	Thursday 11-1 pm	Laucala
S11195863	Prashant Sharma	Tuesday 9 -11 am	Laucala
S11199815	Savindya Fernando	Thursday 11-1 pm	Laucala

Table of Contents

List of Tables 1

List of Figures 1

1. Introduction 2

2. Important Emphasis 3

2.1 Application prototype 3

2.2 Implementation of raw TCP sockets in the prototype 9

3. System Interface 11

4. Implementation Issues 14

5. Conclusion 15

Meet the Team 16

Resources 17

Mark Allocation Table 18

List of Tables

Table 1: The primary socket API functions and methods.....9

List of Figures

Figure 1: The sequence of socket API calls and data flow for TCP 10

1. Introduction

The selected scenario for this assignment is to create a prototype for a chat application by establishing a client-to-client communication using socket programming. There is no a “best programming language” for a chat application. Therefore, the assignment was implemented using Python Programming language to send messages simultaneously allowing users to connect and disconnect the chat at their convenience among the users.

There are many crucial functionalities that is required from the chosen scenario of creating the protype chat application such as,

- Ability to send and receive messages among the users real-time.
- Ability to send an acknowledgement message when a user has joined and left the conversation.

We also implemented some additional features such as,

- A user-friendly Graphical User Interface which enables efficiency of the application which let users’ complete tasks more quickly and easily.
- Availability of the time feature which is visible to the sender and the receiver to check the time that the message has been sent.
- A welcome introductory message to the users to orientate themselves and making the first impression towards the user-friendly application.

2. Important Emphasis

2.1 Application prototype

Source Code

Client.py

```
import socket
import threading
import tkinter
from tkinter import simpledialog, messagebox
from tkinter import scrolledtext
from datetime import datetime

HOST = '127.0.0.1'  # The server's hostname or IP address
PORT = 49746  # The port used by the server


class client:
    def __init__(self):
        # Create a socket object
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.connect((HOST, PORT))

        # GUI
        window = tkinter.Tk()
        window.withdraw()

        while window:

            self.username = simpledialog.askstring("Username", "Please Enter your username", parent=window)


            if self.username:
                self.gui_done = False
                self.running = True

                gui_thread = threading.Thread(target=self.GuiClient)
                receive_thread = threading.Thread(target=self.client_receive)
                gui_thread.start()
                receive_thread.start()
                window.mainloop()
            else:
                messagebox.showwarning("WARNING !", "Username is Required")
```

```
    #  List box for online users
    #  self.clients_list = tkinter.Listbox(self.ui, height=10, width=50)
    #  self.clients  list.pack()

    # create a Label widget to display the number of connected clients
    #  self.num_clients_label = tkinter.Label(self.win, text="Number of clients: 0")
    #  self.num_clients_label.pack()

    self.chatlabel = tkinter.Label(self.ui, text=" WELCOME TO THE CHATAPP ", bg="lightgray")
    self.chatlabel.config(font=("Arial", 12))
    self.chatlabel.pack(padx=20, pady=5)

    self.textarea = tkinter.scrolledtext.ScrolledText(self.ui, bg="#ffffff")
    self.textarea.pack(padx=20, pady=5)
    self.textarea.config(state='disabled')

    self.msglabel = tkinter.Label(self.ui, text="Type Your Message ", bg="lightgray")
    self.msglabel.config(font=("Arial", 12))
    self.msglabel.pack(padx=20, pady=5)

    self.inputarea = tkinter.Text(self.ui, height=5, bg="#ffffff")
    self.inputarea.config(font=("Arial", 12))
    self.inputarea.pack(padx=20, pady=5)

    self.connectButton = tkinter.Button(self.ui, bg="#ffff80", text="Send", command=self.client_send)
    self.connectButton.config(font=("Arial", 12))
    self.connectButton.pack(padx=20, pady=5)

    self.gui_done = True
    # to terminate the window
    self.ui.protocol("WM_DELETE_WINDOW", self.stop)
    self.ui.mainloop()

def stop(self):
    self.running = False
    self.ui.destroy()
    self.sock.close()
    exit(0)

def client_send(self):

    #  get msg
    message = f"\n{self.t_label}\n{self.username}: {self.inputarea.get('1.0', 'end')}"

    # Empty Message handling
    if not len(self.inputarea.get('0.0', 'end')) == 1:
        #  send msg
        self.sock.send(message.encode('utf-8'))
        self.inputarea.delete('1.0', 'end')
    else:
        #  show error
        messagebox.showwarning("Message", "Please Enter a Message")
```

```
    def client_receive(self):

        while self.running:
            try:
                message = self.sock.recv(1024).decode('utf-8')
                if message == 'username?':
                    self.sock.send(self.username.encode('utf-8'))

                # elif message.startswith("[") and message.endswith("]"):
                # clients = eval(message)
                # self.clients_list.delete(0, tkinter.END)
                # for client in clients:
                # self.clients_list.insert(tkinter.END, client)

                # num_clients = len(clients)
                # self.num_clients_label.config(text=f"Number of clients: {num_clients}")

                else:
                    if self.gui_done:
                        self.textarea.config(state='normal')
                        self.textarea.insert('end', message)
                        self.textarea.yview('end')
                        self.textarea.config(state='disabled')

            except:
                self.sock.close()
                exit(0)


client = client()
```

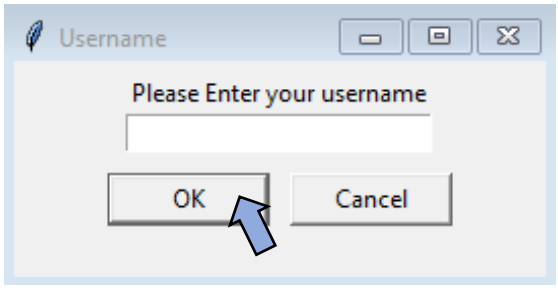
Server.py

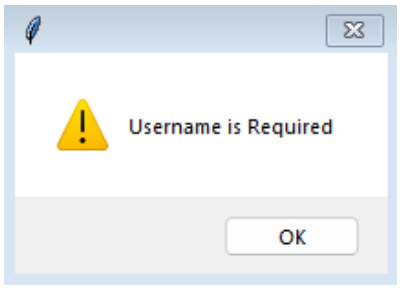
```
import socket
import threading

HOST = '127.0.0.1'  # Symbolic name meaning all available interfaces
PORT = 49746  # Arbitrary non-privileged port

# Create a socket object
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind socket to a specific address and port
server.bind((HOST, PORT))

# Listen for incoming connections
server.listen(1)
clients = []
usernames = []

#  def sendClientList():
#  client_list = str(clients)
#  for username in usernames:
# username.send(bytes(client_list, "utf-8"))

def broadcast(message):
    for client in clients:
        client.send(message)
```

```
def hande_client(client):
    #  sendClientList()
    while True:
        try:
            message = client.recv(1024)
            broadcast(message)
        except:
            index = clients.index(client)
            clients.remove(client)
            client.close()
            username = usernames[index]
            broadcast(f'{usernames} has left the chat room!'.encode('utf-8'))
            usernames.remove(username)
            #  sendClientList()
            break


def receive_message():
    while True:
        print('Server is running and listening ... ')
        client, address = server.accept()
        print(f'connection is established with {str(address)}')
        client.send('username?' .encode('utf-8'))
        username = client.recv(1024)
        usernames.append(username)
        clients.append(client)
        print(f'the nickname of this client is {username}'.encode('utf-8'))
        broadcast(f'{username} has connected to the chat room'.encode('utf-8'))
        client.send('you are now connected!'.encode('utf-8'))
        thread = threading.Thread(target=hande_client, args=(client,))
        thread.start()



if __name__ == "__main__":
    receive_message()
```

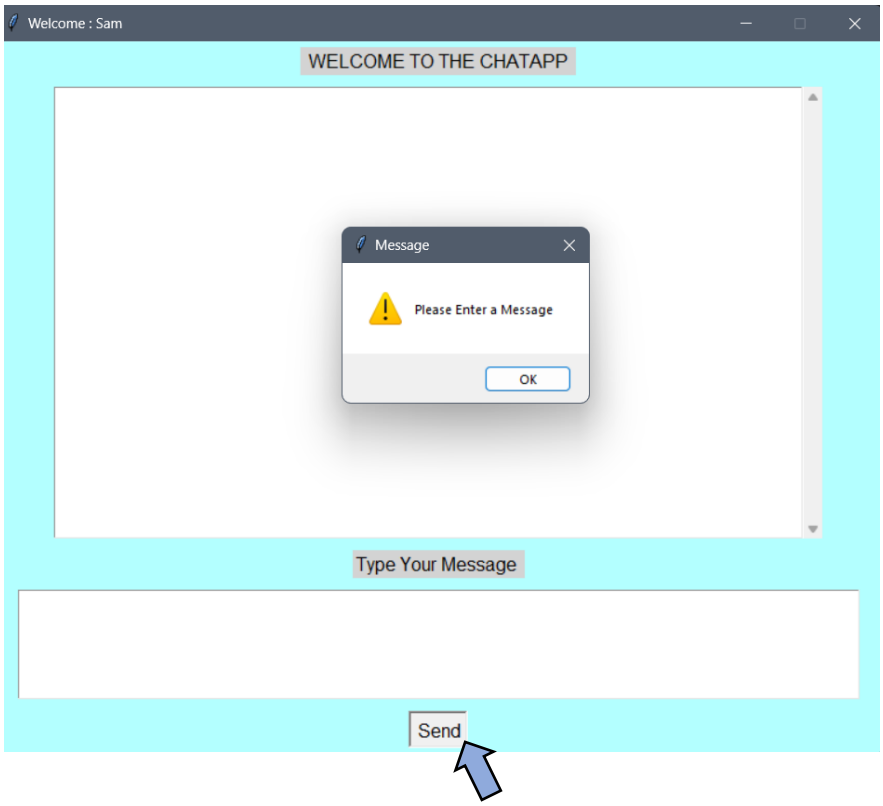
Test Report

Handling Empty Inputs for the Username





Handling Blank Messages



2.2 Implementation of raw TCP sockets in the prototype

Sockets and the socket API are utilized to facilitate communication between processes across a network, whether it be a logical and local connection or a physical connection with access to other networks.
This assignment involves creating a simple socket server and client, as well as an enhanced version capable of handling multiple connections simultaneously. Additionally, a client-client application using a socket-based system will be developed.

This assignment has used primary socket API functions and methods in the Application.

.socket()	Creates a new socket
.bind()	Binds a unique local name to the socket with descriptor socket.
.listen()	listens for connections from clients.
.accept()	When a client connects, the server calls **.accept()** to accept, or complete, the connection.
.connect()	To establish a connection to the server and initiate the three-way handshake.
.send() and .recv()	data is exchanged between the client and server using calls to **.send()** and **.recv()**.
.close()	the client and server close their respective sockets.

Table 1: The primary socket API functions and methods
Source: https://realpython.com/python-sockets/

Python provides a convenient and trustworthy API that comes with built-in classes that make it easier to use low-level socket functions through its standard library.

In our assignment, we utilized the **socket.socket()** method to construct a socket object, specifying the socket type as **socket.SOCK_STREAM**, and the internet address family as

IPv4 with **socket.AF_INET**. By default, the Transmission Control Protocol (TCP) was used for communication in the application.

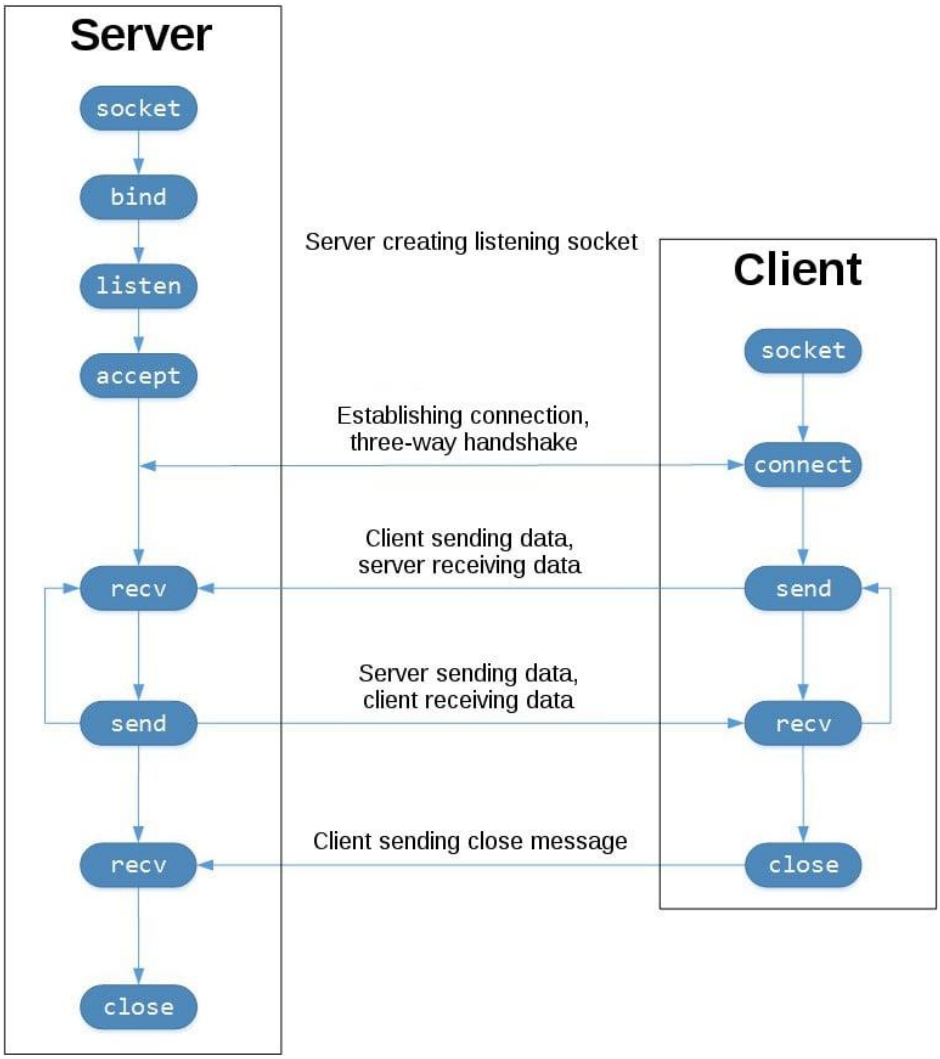


Figure 1: The sequence of socket API calls and data flow for TCP

Source: https://realpython.com/python-sockets/

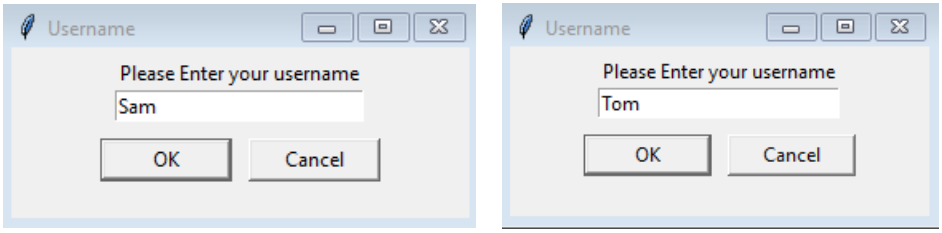
3. System Interface

System Interface is the graphical or visual representation of the application that enables user interaction for a developed application.

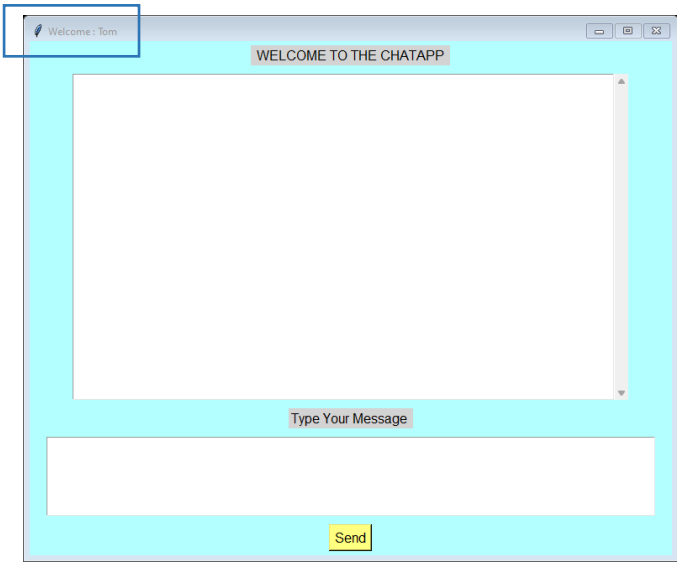
We have implemented the required chat application system interface typically intuitive and user-friendly consists of visual elements that help users navigate the application and carry out different tasks, such as buttons, Dialog Boxes and Time Feature. It also reflects the branding and style of the application, be visually appealing, and be consistent.

Front-End

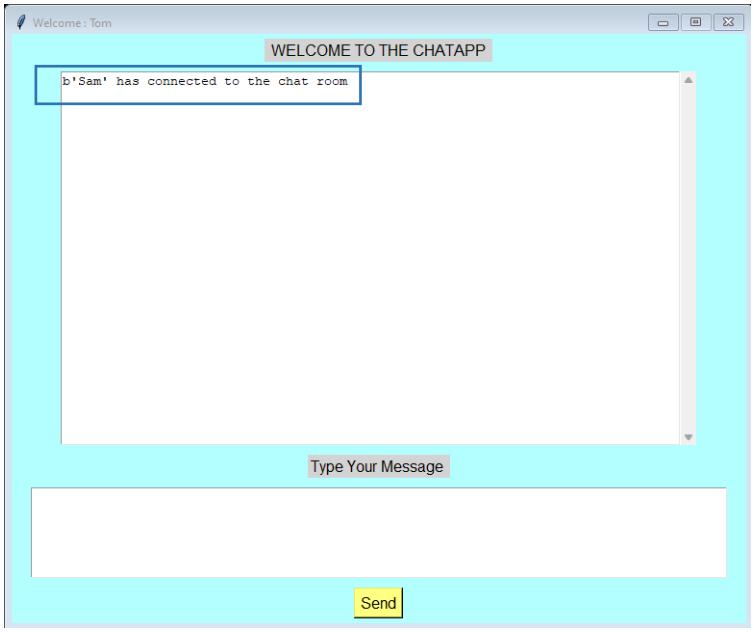
a. Interface to enter the username



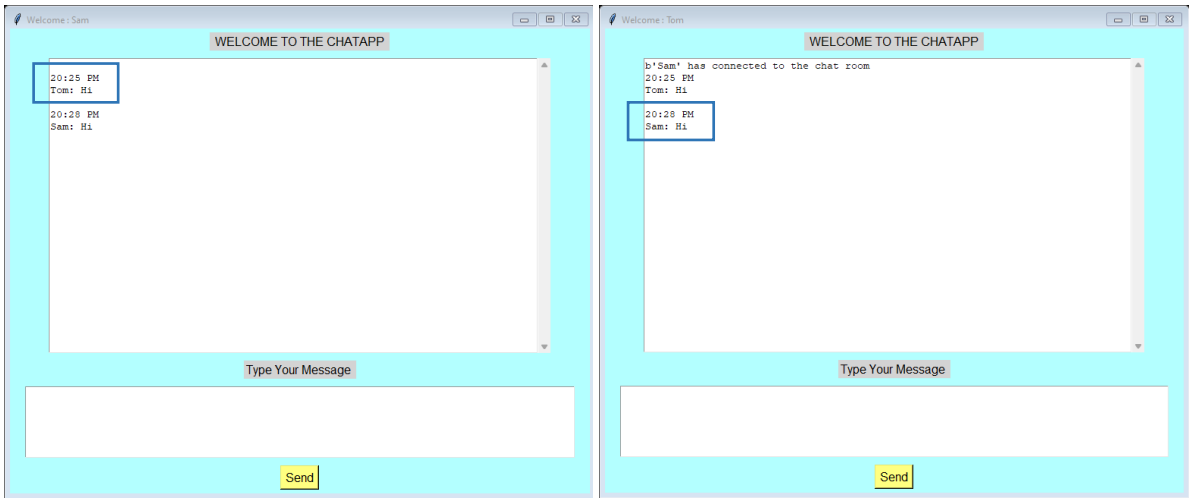
b. Welcome Message to be displayed for Users



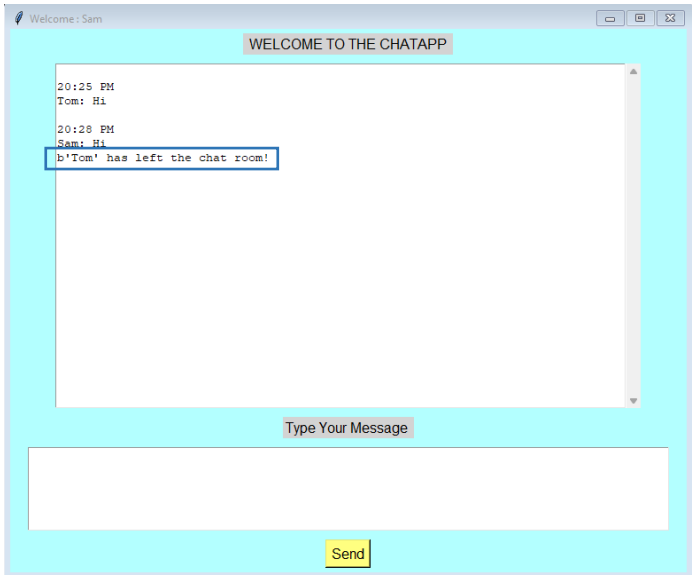
c. Acknowledgment message to be displayed when user joined the chat



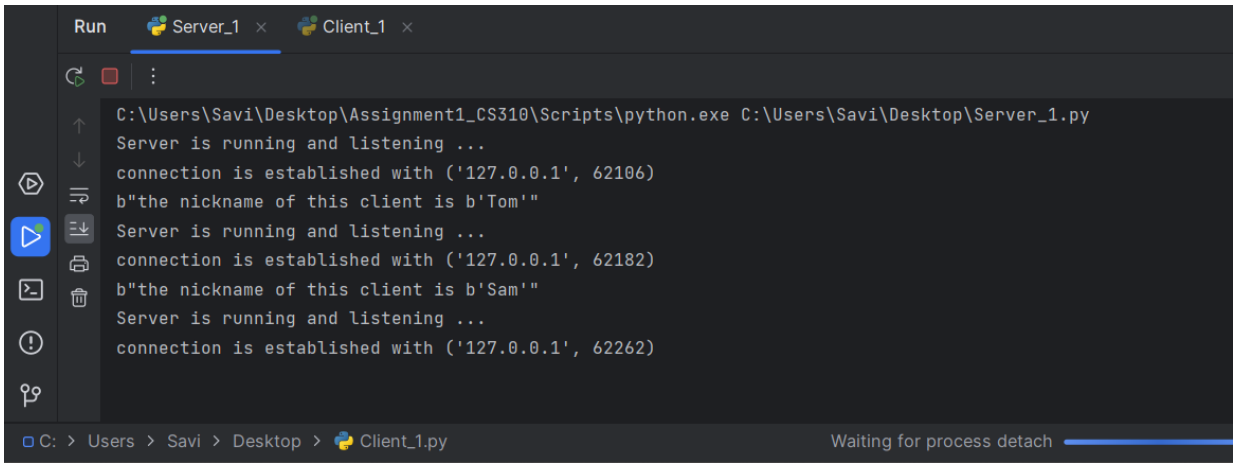
d. Client to Client Communication including the time the message was sent by one client



e. Acknowledgment message when one user leaves the chat



Back- End



4. Implementation Issues

Coding implementation issues refer to challenges that can arise when writing and deploying software code. When coding the chat application our group came across with some implementation issues. Some of the common issues were,

- **Bugs and errors in the code**

We experienced some bugs and errors can cause unexpected behavior for the output such as crashes and other issues that can affect the usability of the application.

- **Performance issues of the code**

When compiled the program we experienced that the code was running slow runs and was causing performance issues, which affected the interface which includes slow load times, lagging when receiving and sending messages.

- **Issues aroused when using a new software**

We selected PyCharm as our coding platform. We had issues when working with the software which was new for all of us especially when installing new tool packages needed when implementing user interfaces.

5. Conclusion

In this assignment, our group successfully achieved experience in creating a chat application using the necessary instructions provided in the course work briefly, using socket application program interfaces (APIs) where the required function of the chat application enables users to communicate directly with one another.

Furthermore, there are various resources used to complete the assignment such as PyCharm Community Edition 2023 as the software which assisted in completing this assignment. There are other resources such as Geeks for Geeks, Tutorial Point and GitHub which used by the team to gain knowledge about socket programming and also to learn and understand the methods in overcoming the unfamiliarized implementation.

We experienced facing various challenges in working as a team to complete the group activity, as well as understanding the various guidelines that were provided. For example,

- Lack of knowledge and experience
- Error codes
- Time Management
- Debugging resulting in consuming time

As a team there were many steps taken to overcome these difficulties such as,

- Selecting a team leader to coordinate and interlink the different activities team members are to be engaged with enabling clear effective communication among team members to manage time accordingly.
- Using different resources to educate the team members of unfamiliar areas and sharing information and knowledge and revised as a team and practiced the problems faced when coding, through lecture notes using various resources.
- Inquiring difficulties regarding the assignment with the Course Coordinator and the Tutor.

In conclusion, facing various difficulties resulted in gaining experience and working as a team to mitigate the difficulties that raised and developed a successful chat Application.

S11198967

Karan Parmar

BSE

S11195863

Prashant Sharma

BNS

S11199815

Savindya Fernando

BSE

Resources

Pizzo, M. (2022). Benefits of Database Management Systems (DBMS) | ZoomInfo. Pipeline Blog. <https://pipeline.zoominfo.com/operations/6-benefits-of-using-database-management-systems-dbms>

Socket Programming in Python <https://www.geeksforgeeks.org/socket-programming-python/>

Socket Programming in Python < https://realpython.com/python-sockets/ >

Mark Allocation Table

Student Name	Contribution	Justification	Signature
Karan Parmar	100%	• GUI • Client-Client Sending and receiving messages • Documentation	K.P.
Prashant Sharma	100%	• Client-Client Sending and receiving messages • GUI • Documentation	P.S.
Savindya Fernando	100%	• GUI • Time Feature • Documentation	S.F.

Group Name – The Reliable UDP

Date – 4/2/2023