**Mumtaz Ali Section A** Date:4/1/2024

**21MDBCS124 Project**

**Operating system**

Client Chat System

Simple Chat Client in Python using Sockets

This Python script demonstrates a basic chat client using

sockets. The client connects to a server and allows the user to

send and receive messages in a chat-like fashion.

1. Importing Necessary Modules

import socket

import sys

import errno

These are the required modules:

**socket**: Provides low-level networking functionality.

**sys**: Offers access to interpreter-specific variables and functions.

**errno**: Defines symbolic error codes.

2. Constants and Configuration

HEADER\_LENGTH = 10

IP = "192.168.43.165"

PORT = 5555

**HEADER\_LENGTH**: Specifies the fixed length of the message header.

**IP and PORT**: Define the IP address and port number for the server.

3. User Input

my\_username = input("Username: ")

Prompts the user to input their desired username.

4. Creating and Connecting Client Socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect((IP, PORT))

client\_socket.setblocking(False)

Creates a client socket using IPv4 and TCP.

Connects to the specified IP and port.

Sets the socket to a non-blocking state.

5. Sending Username

username = my\_username.encode('utf-8')

username\_header = f"{len(username):<{HEADER\_LENGTH}}".encode('utf-8')

client\_socket.send(username\_header + username)

Encodes the username into bytes.

Prepares a header of fixed size with the encoded length.

Sends the encoded username and header to the server.

6. Sending Messages in a Loop

while True:

message = input(f'{my\_username} > ')

if message:

message = message.encode('utf-8')

message\_header = f"{len(message):<{HEADER\_LENGTH}}".encode('utf-8')

client\_socket.send(message\_header + message)

Enters an infinite loop to continuously send messages.

Takes user input for messages.

Encodes messages into bytes, prepares headers, and sends them to the server.

7. Receiving and Printing Messages

try:

while True:

username\_header = client\_socket.recv(HEADER\_LENGTH)

if not len(username\_header):

print('Connection closed by the server')

sys.exit()

username\_length = int(username\_header.decode('utf-8').strip())

username = client\_socket.recv(username\_length).decode('utf-8')

message\_header = client\_socket.recv(HEADER\_LENGTH)

message\_length = int(message\_header.decode('utf-8').strip())

message = client\_socket.recv(message\_length).decode('utf-8')

print(f'{username} > {message}')

except IOError as e:

if e.errno != errno.EAGAIN and e.errno != errno.EWOULDBLOCK:

print('Reading error: {}'.format(str(e)))

sys.exit()

continue

except Exception as e:

print('Reading error: '.format(str(e)))

sys.exit()

Enters a loop to receive and print messages from the server.

Handles various exceptions and errors for non-blocking connections.

This document provides a detailed explanation of each section of the provided Python code for a simple chat client using sockets.