#include <iostream>

#include <string>

#include <WS2tcpip.h>

#include <fstream>

#include <time.h>

#pragma comment(lib, "ws2\_32.lib")

using namespace std;

void swearwordfilter(string& array, int size, string nextbadword) {

int Bwordpointer = 0, Bwordpos = 0;

for (int i = 0; i < size; i++) {

if (tolower(array.at(i)) == nextbadword[Bwordpointer]) {

if (Bwordpointer == 0) {

Bwordpos = i;

}

Bwordpointer++;

}

else

Bwordpointer = 0;

if (Bwordpointer == nextbadword.length()) {

for (int i = Bwordpos; i < Bwordpos + nextbadword.length(); i++) {

array.at(i) = '\*';

}

Bwordpointer = 0;

}

}

}

void main()

{

cout << "\t\t\t\t\t Hello! Welcome to the chat box!" << endl;

cout << "\n";

string ipAddress; // IP Address of the server

int port = 54000; // Listening port # on the server

cout << "Enter Local IP address to connect to: ";

getline(cin, ipAddress);

// Initialize WinSock

WSAData data;

WORD ver = MAKEWORD(2, 2);

int wsResult = WSAStartup(ver, &data);

if (wsResult != 0)

{

cerr << "Can't start Winsock, Err #" << wsResult << endl;

return;

}

// Create socket

SOCKET sock = socket(AF\_INET, SOCK\_STREAM, 0);

if (sock == INVALID\_SOCKET)

{

cerr << "Can't create socket, Err #" << WSAGetLastError() << endl;

WSACleanup();

return;

}

// Fill in a hint structure

sockaddr\_in hint;

hint.sin\_family = AF\_INET;

hint.sin\_port = htons(port);

inet\_pton(AF\_INET, ipAddress.c\_str(), &hint.sin\_addr);

// Connect to server

int connResult = connect(sock, (sockaddr\*)&hint, sizeof(hint));

if (connResult == SOCKET\_ERROR)

{

cerr << "Can't connect to server, Err #" << WSAGetLastError() << endl;

closesocket(sock);

WSACleanup();

return;

}

// Do-while loop to send and receive data

char buf[4096];

string userInput;

string username;

cout << "Please enter your username: ";

getline(cin, username);

system("CLS"); //clrscreen()

// Username

username = username + " > ";

ofstream myfile;

myfile.open("chatlog.txt", std::ios\_base::app);

// aligning the message new chat log into the center

string align;

for (int i = 0; i < 30; i++) align += '-';

myfile <<endl<<align << "New Chat log" << align << endl;

//Loop begin

do

{

// Prompt the user for some text

cout << "You > ";

getline(cin, userInput);

int Fsent = 0;

// Check for commands

if (userInput == "$exit") break; // Exit condition

if (userInput == "$filetransfer") {

printf("\nType file location:\n");

char filename[150];

string Filebuffer;

ifstream Tfile;

gets\_s(filename);

Tfile.open(filename, std::ios\_base::in);

int sendResult = send(sock, "$file", 5, 0);

while (getline(Tfile, Filebuffer)) {

int len = Filebuffer.length();

int sendResult = send(sock, Filebuffer.c\_str(), len + 1, 0);

if (sendResult != SOCKET\_ERROR)

{

// Wait for response

ZeroMemory(buf, 4096);

// accepts acknowledgement from server

int bytesReceived = recv(sock, buf, 4096, 0);

}

}

int endcheck = send(sock, "end", 4, 0);

userInput = "";

Fsent = 1;

// Accepts final acknowledgement

int bytesReceived = recv(sock, buf, 4096, 0);

cout << "SERVER> " << buf << endl;

}

if (Fsent == 1) continue; // Restarts the loop once file is sent

ifstream pFile;

pFile.open("filteredwords.txt", std::ios\_base::in);

//char buffer[100];

string buffer;

while (getline(pFile, buffer)) {

swearwordfilter(userInput, userInput.size(), buffer);

}

userInput = username + userInput; // Combines username and input

//time for client

time\_t result = time(NULL);

char myTime[26];

ctime\_s(myTime, sizeof myTime, &result);

myfile << userInput << endl << " Sent on: " << myTime << endl;

// Send the text

int sendResult = send(sock, userInput.c\_str(), userInput.size() + 1, 0);

if (sendResult != SOCKET\_ERROR)

{

// Wait for response

ZeroMemory(buf, 4096);

int bytesReceived = recv(sock, buf, 4096, 0);

// Server time

ZeroMemory(myTime, 26);

time\_t result1 = time(NULL);//making another time variable to show updated time

ctime\_s(myTime, sizeof myTime, &result1);

myfile << "SERVER > " << buf << endl << " Sent on: " << myTime << endl;

if (bytesReceived > 0)

{

// Echo response to console

cout << "SERVER> " << string(buf, 0, bytesReceived) << endl;

}

}

} while (true);

myfile << "Connection close";

myfile.close();

// Gracefully close down everything

closesocket(sock);

WSACleanup();

}