INDIAN INSTITUTE OF INFORMATION TECHNOLOGY VADODARA-ICD

CS441 COMPUTER PROGRAMMING AND PROBLEM SOLVING

SMART ATTENDANCE SYSTEM

Submitted By
SUBHAM KUMAR MOHANTY

Contents

1 Introduction

1.1 Abstract

Traditional attendance system take time and are error-prone. Manual attendance can lead to proxy attendance, manipulation, and data loss.

To solve this, we developed a Smart Attendance System using Socket Programming in C, where students mark attendance using their devices (phone or laptop) connected to classroom Wi-Fi.

Attendance is recorded on the teacher's system only if:

- Student is within the allowed time window
- Student submits valid roll number and name
- Attendance is not duplicated (only one attendance per roll)

Attendance is stored in attendance.txt with

- Roll Number
- Student Name
- Date & Timestamp

1.2 Problem Statement

Traditional classroom attendance is slow, manual, and prone to errors or proxy marking. There is no reliable way to verify whether a student is actually present in the classroom at the time of attendance.

To solve this, we need a simple automated system that records attendance **only within a fixed time window** and **only from students connected to the classroom network**. Each entry must include **roll number, name, IP address, date, and time**, and duplicate submissions should be blocked.

This project aims to build a **C-based Smart Attendance System** using **socket programming and file handling** to make attendance faster, more accurate, and secure.

Key Issues Addressed :

1. **Manual attendance tracking is slow and unreliable.**
2. **Difficulty in verifying student authenticity during attendance.**
3. **Possibility of proxy attendance (students marking for others).**
4. **Lack of digital record-keeping and timestamped evidence.**

Goal :

1. **Marks attendance automatically within a defined time window.**
2. **Accepts attendance only from devices connected to the same classroom network.**
3. **Stores roll number, name, date, time, and IP address in a file for verification.**
4. **Prevents multiple or duplicate entries from the same device (IP).**

1.3 Scope of Work

The scope of this project includes the design and implementation of a **network-based attendance system** using the C programming language. The system allows students to submit their roll number and name from their device, and the server validates and records the attendance.

The work covered in this project includes:

I. Server-Side Functionality :

- Hosting an attendance server using **TCP sockets**.
- Accepting multiple student requests one by one.
- Verifying student presence through **IP address** (classroom network check).
- Enforcing **time-based attendance window**.
- Blocking **multiple submissions** from the same IP.
- Saving attendance to a file with:
  - Roll number
  - Name
  - Time
  - Date

II. Client-Side Functionality

- Connecting to the teacher’s server using IP address.
- Taking roll number and name as input.
- Sending attendance data to the server via sockets.
- Receiving confirmation or rejection messages.

III. File Handling

- Storing attendance records permanently in **attendance.txt**.
- Ensuring proper formatting and error-free entry storage.

IV. System Validation

- Handling invalid IPs, incorrect server address, and connection failures.
- Ensuring only **legitimate** students connected to the classroom Wi-Fi can mark attendance.

V. C Programming Concepts Used

- Socket programming (TCP)
- File handling
- Time & date API
- Strings and buffers
- Control structures

2 Objectives

The main objective of this project is to develop a **secure and automated Smart Attendance System** using the C programming language and socket programming. The system aims to simplify the attendance process while ensuring accuracy and preventing misuse.

- Automate Attendance Collection

  To eliminate manual entry and reduce time by enabling students to mark attendance digitally through a network.

- Ensure Authenticity of Attendance

  To verify that attendance is marked only by students physically present in class by checking their **IP address and Wi-Fi network**.

- Implement Time-Based Validation

  To allow attendance submissions **only within a predefined time window**, preventing late or early entries.

- Prevent Duplicate Entries

  To ensure that each student/IP can submit attendance **only once**, improving integrity of the record.

- Store Data Securely

  To save attendance records (roll number, name, date, time, IP) in a file for future use and verification.

- Demonstrate Practical Use of C Programming Concepts

  To implement a real-world solution using:

  - Socket programming
  - File handling
  - Time & date functions
  - Strings and data structures

3 Tools/Environment Used

A. Programming Language

C Language used for developing both server and client programs, implementing socket communication, file handling, and time-based validation.

B. Compiler

GCC (MinGW / TDM-GCC) used to compile C programs on Windows, supporting Winsock2 and standard C libraries.

C. Networking Library

Winsock2 (ws2_32.lib) provides all functions for socket programming such as *socket(), bind(), listen(), accept(), connect(), send(), and recv().*

D. Operating System

Windows 10/11 required for Winsock2-based networking and execution of both server and client programs.

E. Text Editor / IDE

isual Studio Code (VS Code) used to write and edit C source code files.

F. Network Environment

Classroom Wi-Fi Network ensures that students and teacher are on the same local network for successful communication.

G. File System

attendance.txt used to store attendance records permanently using C file handling.

4 Hardware and Software Requirements

4.1 Hardware Requirements:

A. Teacher Device

- Minimum **Dual Core Processor**
- **4 GB RAM** or higher
- **Wi-Fi connectivity** (must be connected to classroom network)
- At least **50 MB free disk space** (for storing attendance logs)

B. Student Devices

- Any device capable of running C program or a simple client application:
    - **Windows Laptop** (preferred)
    - Alternatively, **Android phone using a terminal app** (if applicable)
- Must be connected to the **same classroom Wi-Fi network** as the server

4.2 Software Requirements:

A. Programming Tools

- **C Programming Language**
  Entire system is developed using C (TCP sockets + file handling).
- **GCC Compiler (MinGW/TDM-GCC)**
  Required for compiling C programs on Windows. Supports Winsock2 networking.

B. Networking Libraries

- **Winsock2 Library (ws2_32.lib)**
  Enables socket programming on Windows. Provides functions like:
  socket(), bind(), listen(), accept(), connect(), send(), recv().

C. Development Environment / Edior

- **Visual Studio Code (VS Code)**
  Used for writing, editing, and managing server and client code.

D. Operating System

- **Windows 10 / Windows 11**
  Required for Winsock-based networking.

E. Terminal / Command Prompt

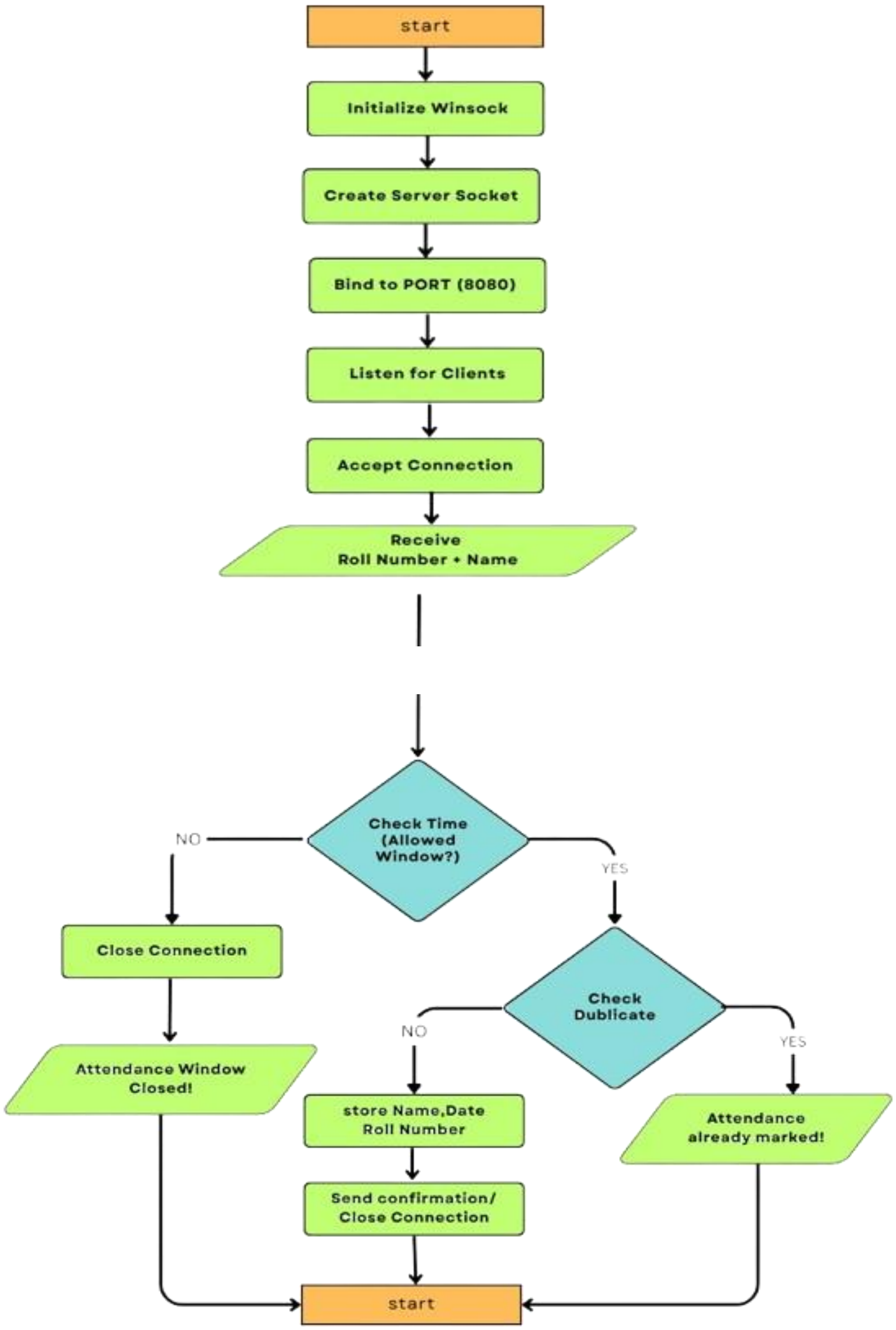
- CMD or PowerShell
  Used for compiling and running the executables: server.exe, client.exe

F. Terminal / Command Prompt

- Text file: **attendance.txt**
  Automatically created by the server to store attendance logs.

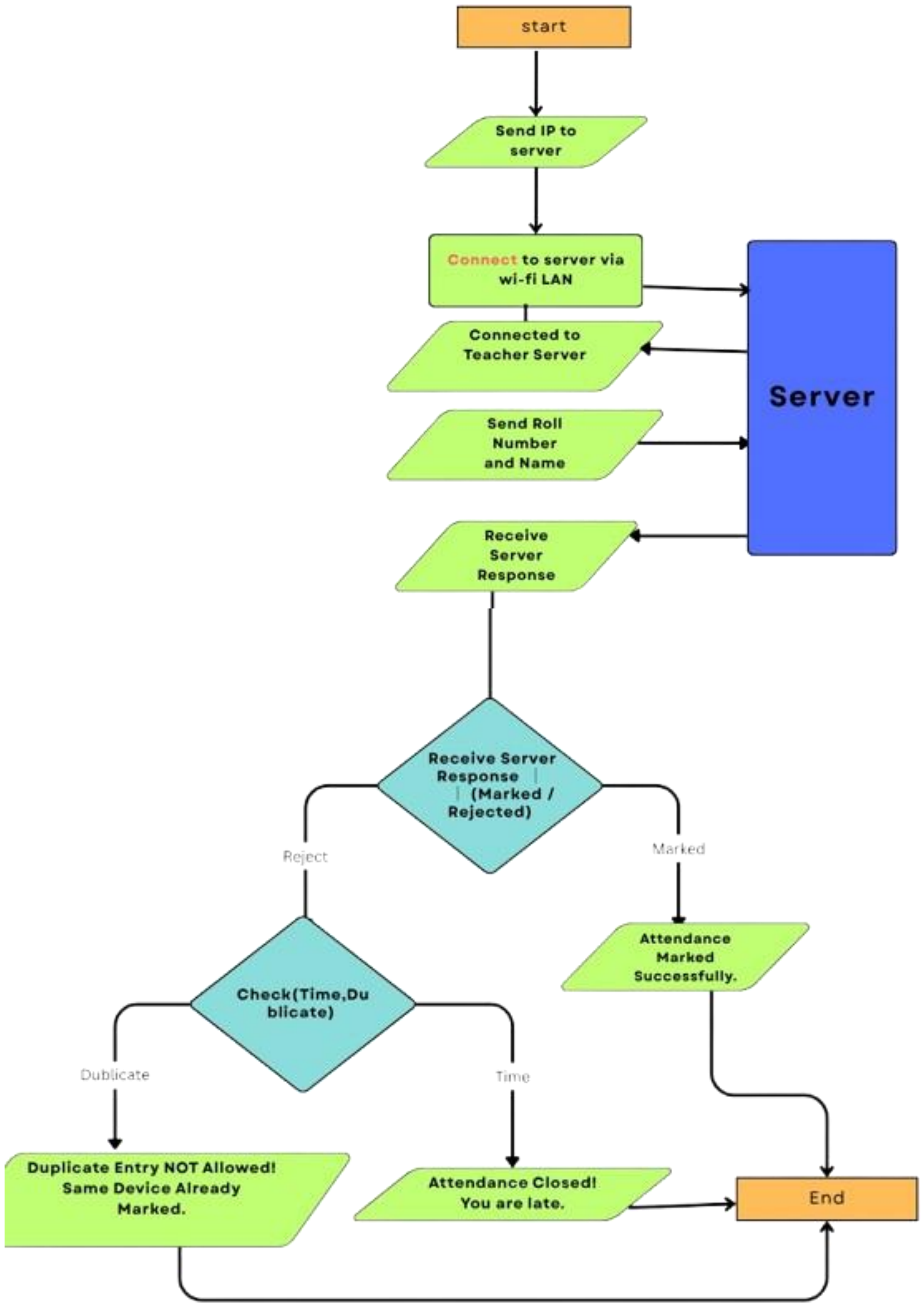
4.3 Flowchart

A. Teacher Side :



....

A. Students Side :


start
Send IP to server
Connect to server via wi-fi LAN
Connected to Teacher Server
Send Roll Number and Name
Server
Receive Server Response
Receive Server Response (Marked / Rejected)
Reject
Marked
Attendance Marked Successfully.
Check(Time,Dublicate)
Dublicate
Time
Duplicate Entry NOT Allowed! Same Device Already Marked.
Attendance Closed! You are late.
End

4.4 Source Code

A. Teacher Side:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <winsock2.h>
#include <time.h>

#pragma comment(lib, "ws2_32.lib")

#define PORT 8080
#define MAX_BUF 512

// Time window (24hr format)
#define START_HOUR 10
#define START_MIN 55
#define END_HOUR 11
#define END_MIN 10

// Check if current time is inside the allowed attendance window
int isAttendanceAllowed()
{
   time_t now = time(NULL);
   struct tm *t = localtime(&now);

   int currentMin = t->tm_hour * 60 + t->tm_min;
   int startMin = START_HOUR * 60 + START_MIN;
   int endMin = END_HOUR * 60 + END_MIN;

   return (currentMin >= startMin && currentMin <= endMin);
}

// Checks if roll already exists in file
int isRollDuplicate(const char *roll)
{
   FILE *fp = fopen("attendance.txt", "r");
   if (!fp)
      return 0;
```

```
    char line[256];
    while (fgets(line, sizeof(line), fp))
    {
        if (strncmp(line, roll, strlen(roll)) == 0)
        {
            fclose(fp);
            return 1;
        }
    }
    fclose(fp);
    return 0;
}

int main()
{
    WSADATA wsa;
    SOCKET serverSocket, clientSocket;
    struct sockaddr_in server, client;
    int c;
    char buffer[MAX_BUF];
    char clientIP[50];

    printf("\n==== SMART ATTENDANCE SERVER STARTED ====\n");

    // Initialize Winsock
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
    {
        printf("WSAStartup failed!\n");
        return 1;
    }

    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (serverSocket == INVALID_SOCKET)
    {
        printf("Socket creation failed!\n");
        return 1;
    }

    server.sin_family = AF_INET;
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons(PORT);
```

```
if (bind(serverSocket, (struct sockaddr *)&server, sizeof(server)) ==
        SOCKET_ERROR)
{
    printf("Bind failed!\n");
    return 1;
}

listen(serverSocket, 3);
printf("Server is waiting for students...\n");

c = sizeof(client);

while (1)
{
    clientSocket = accept(serverSocket, (struct sockaddr *)&client, &c);

    strcpy(clientIP, inet_ntoa(client.sin_addr)); // Get student IP

    memset(buffer, 0, sizeof(buffer));
    recv(clientSocket, buffer, sizeof(buffer), 0);

    // Data format = "ROLL,NAME"
    char *roll = strtok(buffer, ",");
    char *name = strtok(NULL, ",");

    printf("\nRequest from IP: %s\n", clientIP);
    printf("Roll: %s, Name: %s\n", roll, name);

    //  Check time restriction
    if (!isAttendanceAllowed())
    {
        char msg[] = "Attendance Window Closed!";
        send(clientSocket, msg, strlen(msg), 0);
        closesocket(clientSocket);
        continue;
    }

    //  Check duplicate roll
    if (isRollDuplicate(roll))
    {
        char msg[] = "Attendance already marked for this roll!";
```

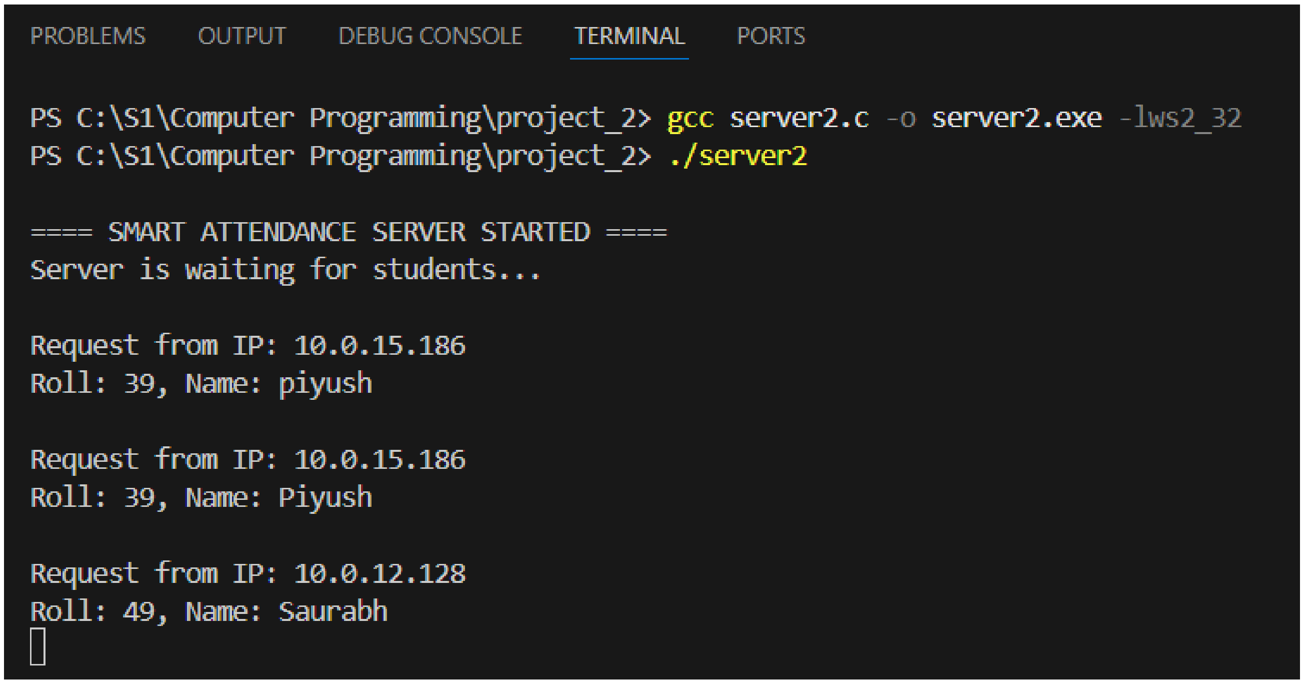
```
            send(clientSocket, msg, strlen(msg), 0);
            closesocket(clientSocket);
            continue;
        }

        //  Write attendance to file
        FILE *fp = fopen("attendance.txt", "a");
        time_t now = time(NULL);
        struct tm *t = localtime(&now);

        fprintf(fp, "%s\t%s\t\t%02d-%02d-%04d   %02d:%02d:%02d\n",
                roll, name,
                t->tm_mday, t->tm_mon + 1, t->tm_year + 1900,
                t->tm_hour, t->tm_min, t->tm_sec);

        fclose(fp);

        char msg[] = "Attendance Marked Successfully!";
        send(clientSocket, msg, strlen(msg), 0);

        closesocket(clientSocket);
    }

    closesocket(serverSocket);
    WSACleanup();
    return 0;
}

// gcc server2.c -o server2.exe -lws2_32
//./server2.exe
```

B. Student's Side:

```
#include <stdio.h>
#include <string.h>
#include <winsock2.h>

#pragma comment(lib, "ws2_32.lib")

#define PORT 8080
#define MAX_BUF 512

int main() {
    WSADATA wsa;
    SOCKET clientSocket;
    struct sockaddr_in server;

    char serverIP[50];
    char roll[50], name[100];
    char sendData[MAX_BUF], response[MAX_BUF];

    printf("\n==== STUDENT ATTENDANCE CLIENT ====\n");

    printf("Enter Teacher (Server) IP: ");
    scanf("%s", serverIP);
    getchar();

    if (WSAStartup(MAKEWORD(2,2), &wsa) != 0) {
        printf("WSAStartup failed!\n");
        return 1;
    }

    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (clientSocket == INVALID_SOCKET) {
        printf("Socket Error!\n");
        return 1;
    }

    server.sin_family = AF_INET;
    server.sin_port = htons(PORT);
    server.sin_addr.s_addr = inet_addr(serverIP);

    if (connect(clientSocket, (struct sockaddr*)&server, sizeof(server)) < 0) {
        printf("Connection failed!\n");
        return 1;
    }

    printf("\nConnected\n");
    printf("Enter Roll Number: ");
    scanf("%s", roll);
    getchar();

    printf("Enter Student Name: ");
    fgets(name, sizeof(name), stdin);
    name[strcspn(name, "\n")] = 0;

    sprintf(sendData, "%s,%s", roll, name); // Combine

    send(clientSocket, sendData, strlen(sendData), 0);

    int recvSize = recv(clientSocket, response, sizeof(response), 0);
    response[recvSize] = '\0';

    printf("\nServer Response: %s\n", response);

    closesocket(clientSocket);
    WSACleanup();

    return 0;
}

// gcc client2.c -o client2.exe -lws2_32
// ./client2.exe
```

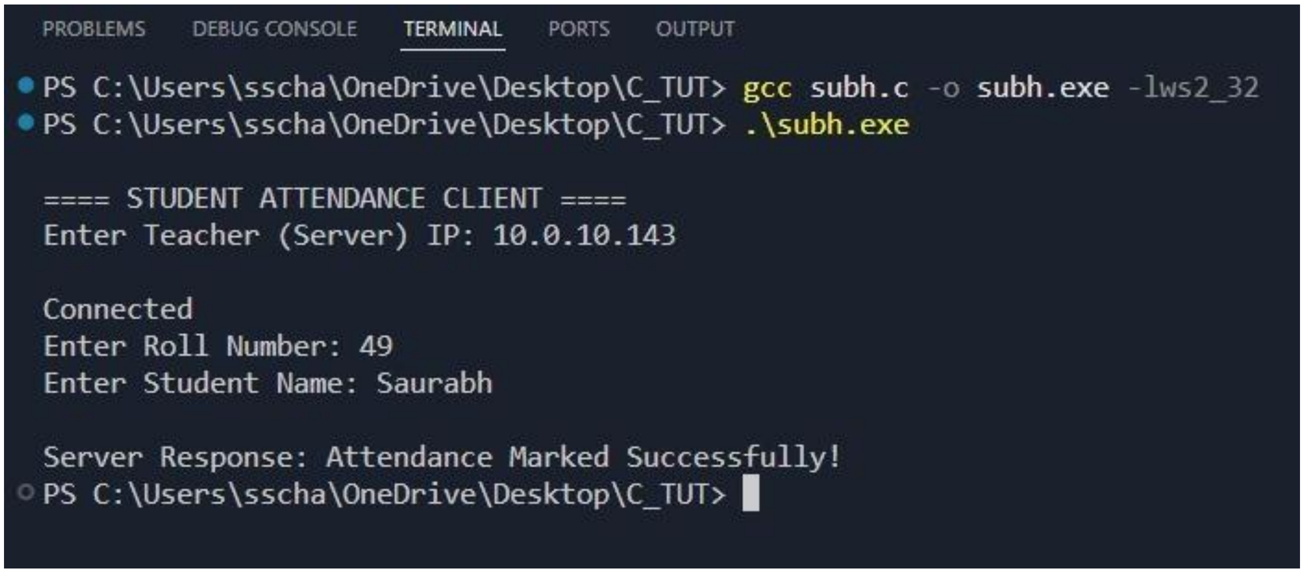
4.5 Output

- **Teacher Side :**



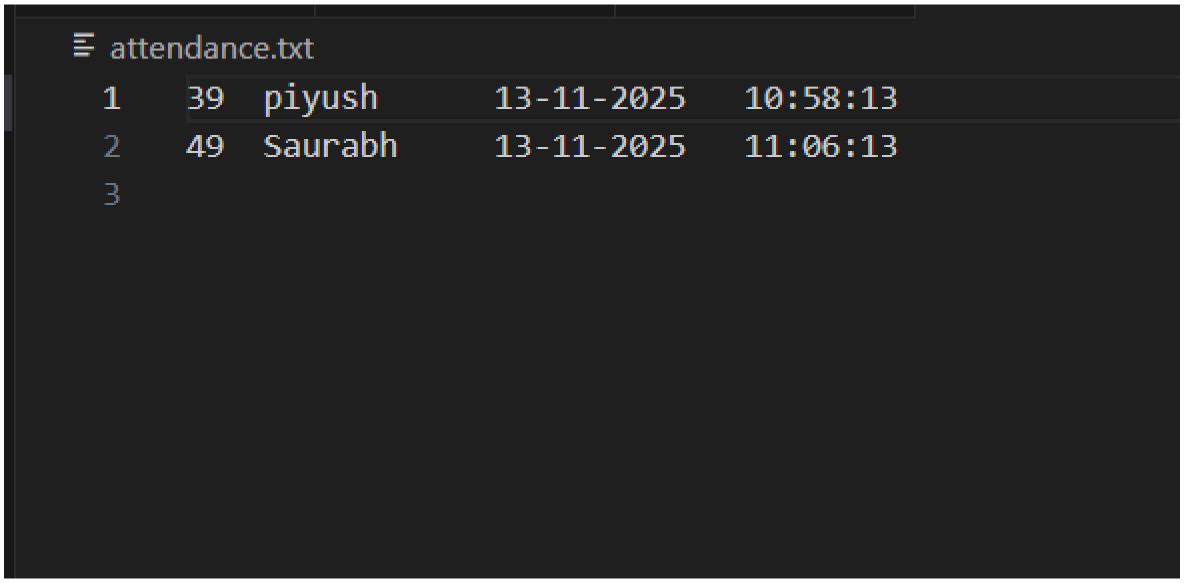
```
PS C:\S1\Computer Programming\project_2> gcc server2.c -o server2.exe -lws2_32
PS C:\S1\Computer Programming\project_2> ./server2

==== SMART ATTENDANCE SERVER STARTED ====
Server is waiting for students...

Request from IP: 10.0.15.186
Roll: 39, Name: piyush

Request from IP: 10.0.15.186
Roll: 39, Name: Piyush

Request from IP: 10.0.12.128
Roll: 49, Name: Saurabh
```

- **Student's Side :**



```
PS C:\Users\sscha\OneDrive\Desktop\C_TUT> gcc subh.c -o subh.exe -lws2_32
PS C:\Users\sscha\OneDrive\Desktop\C_TUT> .\subh.exe

 ==== STUDENT ATTENDANCE CLIENT ====
 Enter Teacher (Server) IP: 10.0.10.143

 Connected
 Enter Roll Number: 49
 Enter Student Name: Saurabh

 Server Response: Attendance Marked Successfully!
PS C:\Users\sscha\OneDrive\Desktop\C_TUT>
```

- **Attendance Record File :**


attendance.txt
```
39  piyush      13-11-2025   10:58:13
49  Saurabh     13-11-2025   11:06:13
```

5. Future Scope

The Smart Attendance System can be improved further with the following enhancements:

- **Multi-client support** using threads to handle many students at the same time.
- **Mobile/Web app integration** to allow easier attendance marking.
- **Database storage** for better record management instead of text files.
- **QR code or OTP-based attendance** for higher security.
- **Geo-location or MAC address verification** to ensure the student is physically present.
- **Real-time dashboard** for teachers to monitor attendance instantly.