일정 관리 메신저 만들기

유승훈

일정 관리 메신저

- 카카오톡의 메신저기능을 참고하여 최대한 많은 기능을 지원할 수 있도록 작성하였다.
- 단순히 메신저의 기능을 넘어서, 메신저를 통한 개인의 일정관리 시스템을 적용하였다.
- 자신의 일정을 메신저에 적용함으로서 사람들과 소통 중 따로 달력을 사용하지 않고도 자신의 일정을 정확히 알고 이후 일정을 정하는것에 도움을 줄 수 있다.
- 해당 기능을 ios의 ‘항공편 문자 수령 시 달력에 적용되는것 ' 에서 착안하였다.

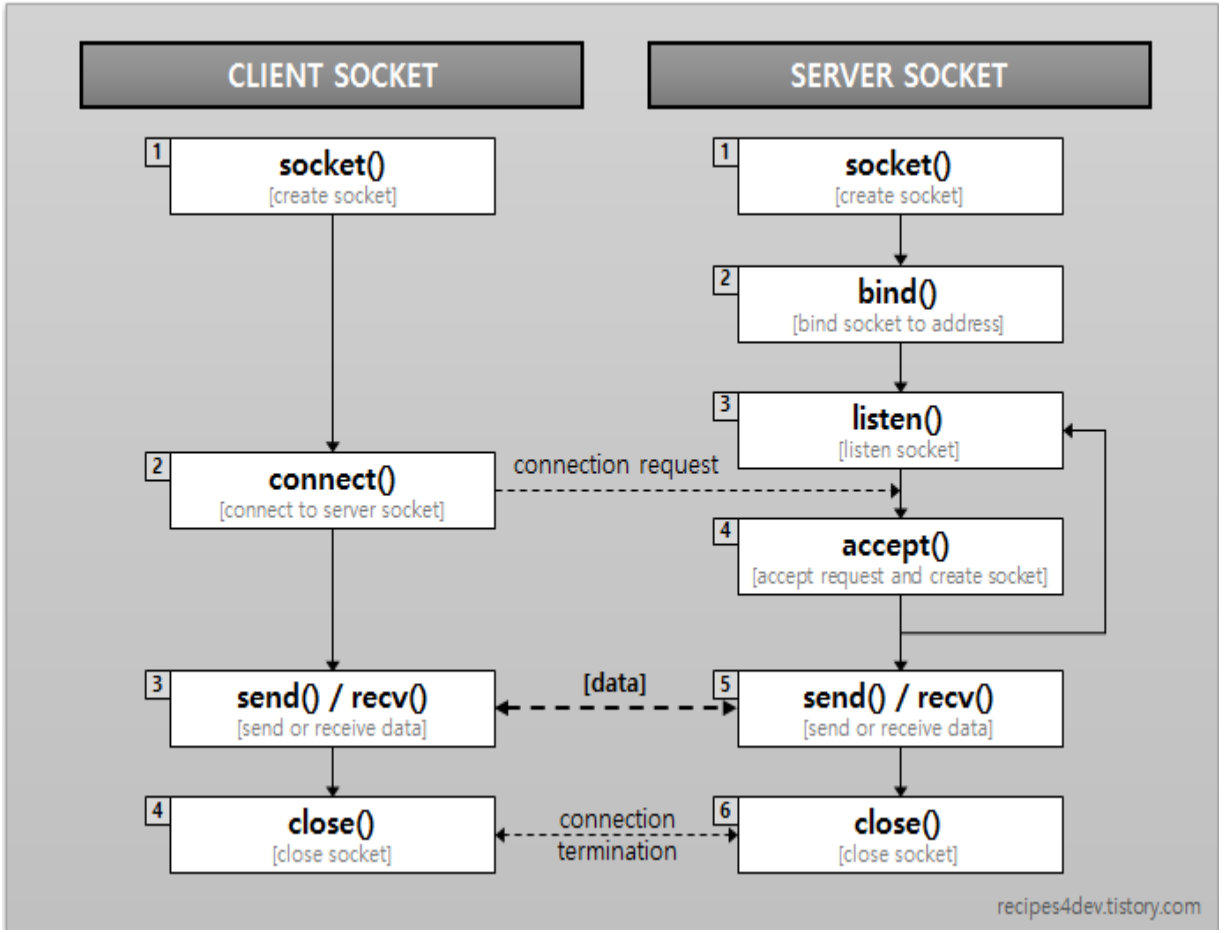
주요기능

- 메신저의 유저로서 회원가입,로그인 할 수 있다.
- 자신의 일정을 달력형태로 확인할 수 있다.
- 친구의 id를 통해 친구추가 할 수 있다.
- 자신이 원하는 사람들과 채팅방을 형성할 수 있다
- 특정 채팅 읽은 사람 기능을 지원한다.
- 채팅은 실시간으로 접속중인 사용자들에게 알람이 간다.
- 채팅방을 접속 시 자신이 읽지 않았던 채팅부터 로그를 띄워준다.
- 채팅을 일주일이 지나면 서버에서 삭제된다.
- 채팅방에서 일정을 잡기위해 !일정잡기! 메시지를 보낼 수 있다.
- !일정잡기! 메시지를 통해 일정을 잡을 수 있으며 이는 서버 내부 개인의 일정에 맞춰 수락,거절 될 수 있다.

Socket programing

- Pc간 인터넷을 이용한 통신,os에서 구현된 소켓을 이용하면 단순 패킷(데이터)를 통한 통신을 구현할 수 있다.

- 서버의 소켓이 클라이언트의 소켓을 연결받아 그 둘사이의 recv(수신) ,send(송신)을 수행하며 통신이 가능하다.



통신 패킷 정하기

- 통신 패킷은 컴퓨터 네트워크가 전달하는 데이터의 형식화된 블록. 이를 설정함으로서 서버의 구현이 가능
- 기본적으로 한 개의 버퍼(path[255])를 송,수신하며 통신이 이루어짐
- 버퍼의 구조는 다음과 같음

  - 000 : messanger close
  - 001 : 친구요청 수락
  - 002 : 친구요청 송신
  - 003 : 채팅요청 수락
  - 004 : 채팅방 생성
  - 005 : 메시지 수신
  - 006 : 알람 송신
  - 007 : 일정 초대
  - 008 : 일정 수락
  - 009 : 일정 취소(개인)
  - 010 : 일정 취소(그룹)
  - 011 : 일정 거절

통신코드: 0~2	통신할 메시지 : 3~254

```
if (strcmp(code.c_str(), "000") == 0) {// MessangerClose 입력받음
        tool::TxtToSocket("c:/server/Id_Socket_map.txt", socket_info);
        cout << "id:" + Id + " is now logout" << endl;
        iter = socket_info.find(Id);
        if (iter != socket_info.end()) {
                socket_info.erase(iter);
        }
        tool::SocketToTxt("c:/server/Id_Socket_map.txt", socket_info);
        return 0;
}
else if (strcmp(code.c_str(), "001") == 0) {
        char* recv_id = strtok(msg, " ");
        recv_id = strtok(NULL, " ");
        AddFriends Add(client, Id, recv_id);
        Add.Delete_invite(client, Id, recv_id);
}
else if (strcmp(code.c_str(), "002") == 0) {//SendInvite 입력받음
        cout << "Sending invite message" << endl;
        char* recv_id = strtok(msg, " ");
        recv_id = strtok(NULL, " ");
        vector<string> V_id;
        tool::TxtToVector("c:/server/id_index.txt", V_id);
        if (find(V_id.begin(), V_id.end(), recv_id) == V_id.end()) {
                Send(client, "001");
        }
```

한 서버가 여러 개의 사용자를 수용하기 위해

- 이번에 선택한 방법은 멀티 쓰레드 프로그래밍 입니다. 이는 하나의 프로세스에 여러 개의 쓰레드를 만들어 사용합니다. 주의점으로는 여러 개의 쓰레드가 공유 자원에 접근할 때 주의할 필요가 있습니다.
- 서버는 지속적으로 소켓을 열어 사용자를 받아들이며 각 소켓은 쓰레드를 통해 비 동기적으로 실행됩니다.

```
int main() {
	while (1) {
		WSADATA data;
		WORD ver = MAKEWORD(2, 2);
		map<string, SOCKET>socket_info;
		ofstream wFile("c:/server/Id_Socket_map.txt");
		for (map<string, SOCKET>::const_iterator iterator = socket_info.begin(); iterator != socket_info.end(); ++
			wFile << iterator->first << "|" << iterator->second;
			wFile << "\n";
		}
    if (strcmp(buf, "Signin") == 0) {//client의 login 요청
       //thread Signin
            thread SIGN(&Sign, sock);
            SIGN.detach();
    }
    else if (strcmp(buf, "Login") == 0) {
            thread LOGIN(&Log, sock);
            LOGIN.detach();
    }
    else if (strcmp(buf, "TimeTable") == 0) {
            cout << "TimeTable in" << endl;
            buf[MAX_BUFFER_SIZE];
            ZeroMemory(buf, MAX_BUFFER_SIZE);
            int byteIn = recv(sock, buf, MAX_BUFFER_SIZE, 0);
            if (byteIn <= 0) {
                    closesocket(sock);
                    FD_CLR(sock, &Fd);
            }
            else {
                    thread TimeTable(&Table, sock, buf);
                    cout << "접속:" << buf << endl;
                    TimeTable.detach();
            }
    }
```

멀티 쓰레드의 동기화 문제

- 모든기능은 한 서버에서 관리됩니다 이에 따라 여러 개의 쓰레드가 동시에 자료에 접근하여 문제가 생기는 경우를 발견하였습니다.(채팅로그 추가 등)
- 이러한것들은 mutex lock을 통해 해결해보았습니다. 뮤텍스를 통해 특정파일들(채팅방 채팅로그) 은 한번에 한 개의 쓰레드만이 접근 가능하도록 설정하였습니다.

```
static mutex mtx;
static void mux(SOCKET client, string msg) {
        mtx.lock();
        Sleep(10);
        send(client, msg.c_str(), MAX_BUFFER_SIZE, 0);
        mtx.unlock();
}
```

프로그램의 취약점

- 모든 통신에 보안을 넣지 않은 점, 비밀번호 등이 그대로 노출될 수 있음

- 쓰레드의 동기화 문제 : 동기화가 그렇게까지 완벽하게 동작하지 않은 점 , 분명 채팅로그 파일에 뮤텍스를 걸어놨지만 write가 꼬여버리는 현상이 간혹 발견되었음, 또한 뮤텍스의 특성 상 서버가 한번에 여러프로세스를 처리하지 못하는 경우가 발생하므로 이는 불필요한 지연이라고 생각됨.

- 통신에러 : 통신이 정상적으로 완료되지 않는경우가 발견된 점, 정해진 통신 패킷에서 벗어난 통신이 프로그램에 에러를 발생시켰음