

# **Computer Network Term Project**

Using TCP Communication, KakaoTalk Messenger  
2022. 12. 14. Wes

Team A

# Members – Team A



Park Jaemin  
201835457

**Backend API &  
DB Development**



Pyeon SeoHee  
201937453

**Backend  
Development API**



Jeong MinGyu  
202135577

**Frontend  
Development**

# Tables of contents

---

**Part 01** Introduction

**Part 02** Structure

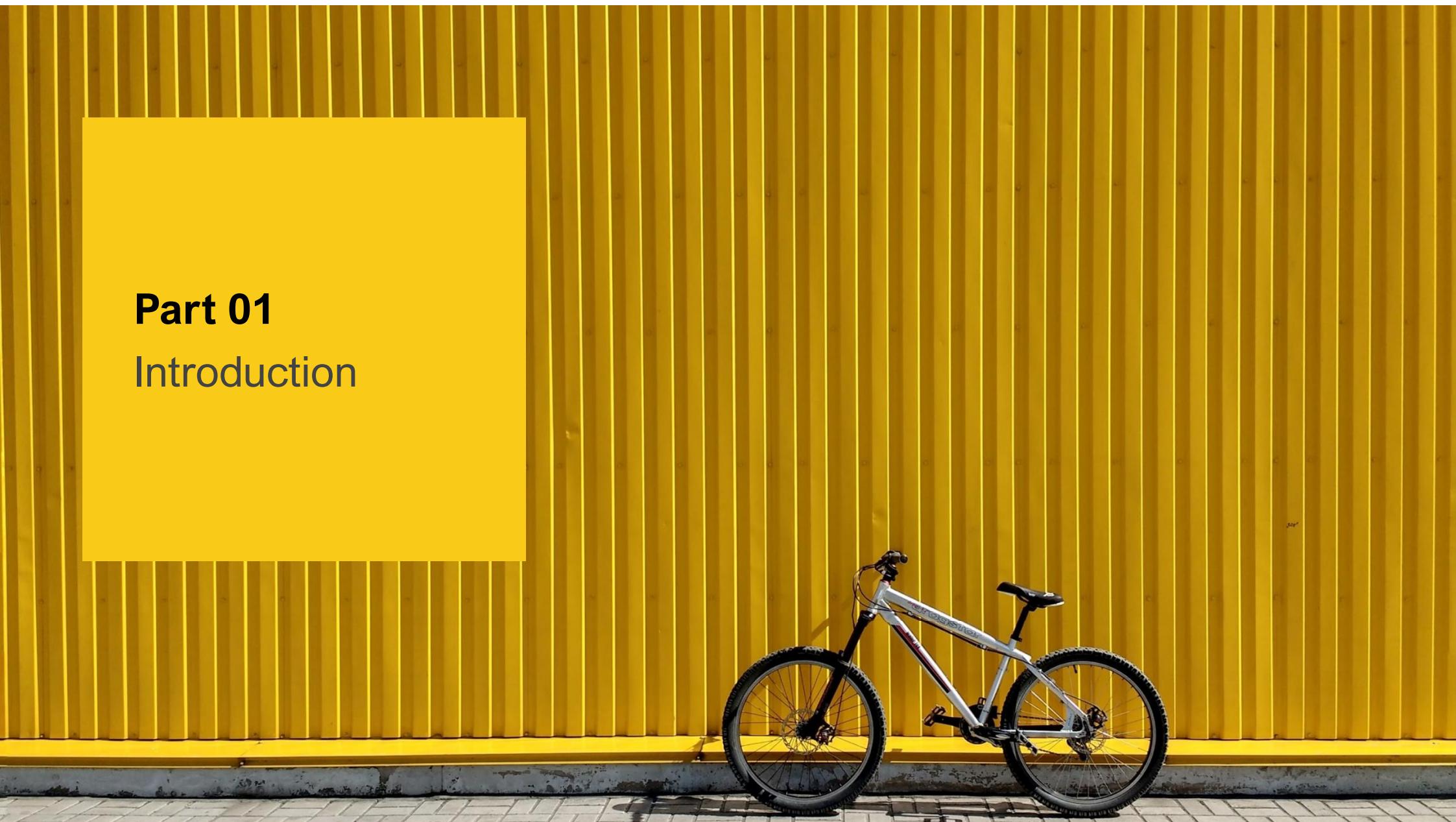
**Part 03** API description

**Part 04** Result



# **Part 01**

## Introduction



# Simple Online Messenger

## Requirements

1. Register & Unregister
2. Login
3. Manage my status
4. Manage friend status
5. Chatting room(real-time and multi)
6. Using Public data





## Part 02, Structure

# **REST ( = Representational state Transfer)**

---

## **1. What is REST?**

- 1) REST refers to sending and receiving the state of a resource, separated by a name.
- 2) It's an architectural style that takes full advantage of the Web's advantages by leveraging the Web's existing technology and HTTP protocol.
- 3) It's one of the communication methods between Client and Server on a network.

## **2. Characteristics of REST**

- 1) It has 'Server – Client Structure', and the one who has the resource becomes the Server, and the one who requests the resource become the Client.
- 2) Less dependence on each other's dependence.

# REST API ( = Representational state Transfer API)

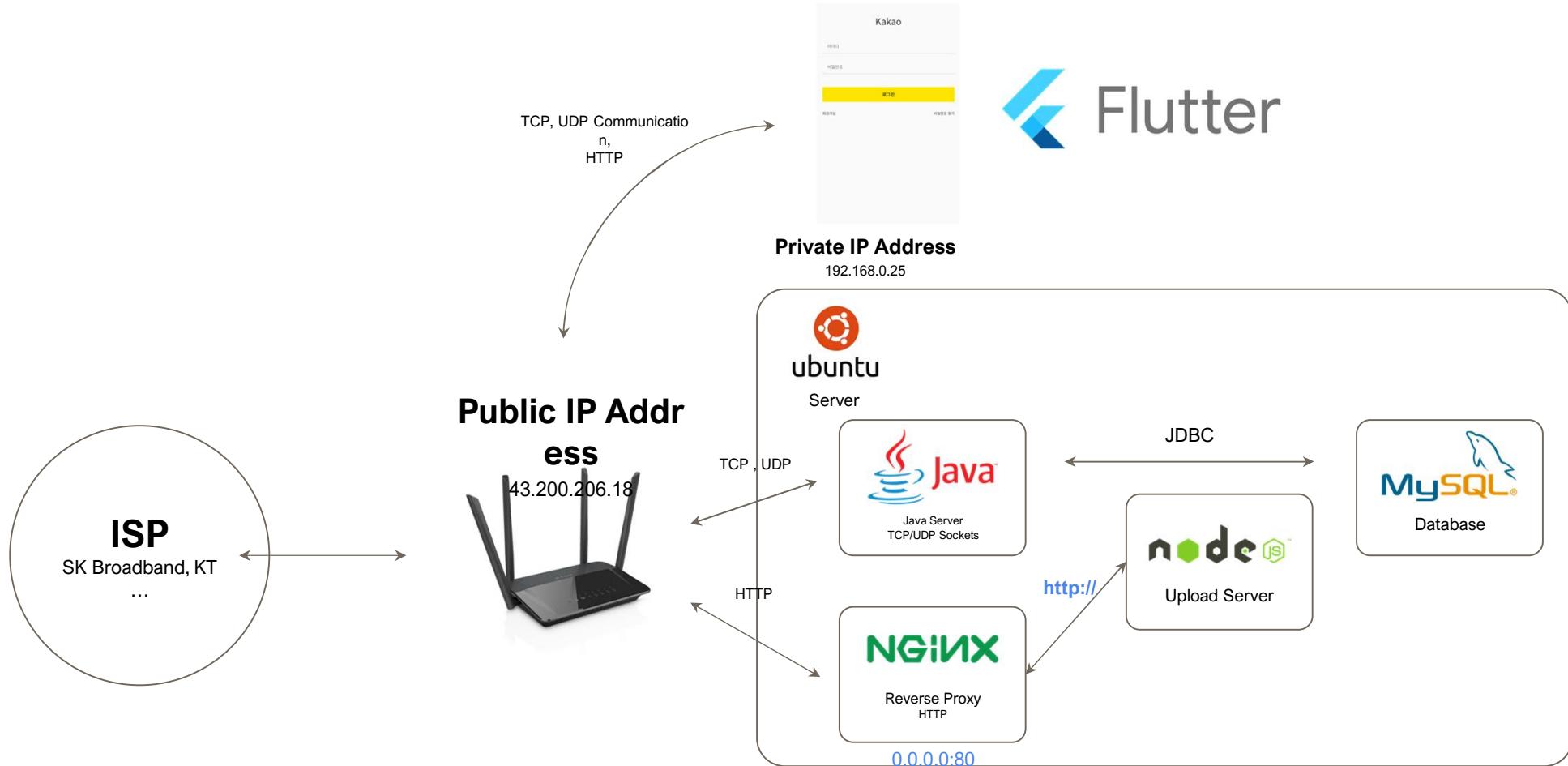
## 1. Characteristics of REST API

It's implemented based on HTTP standards, allowing clients and servers to be implemented in program languages that support HTTP.

## 2. REST API Design Rules

CRUD	HTTP verbs	Route
Display a list of resources	GET	/resource
resource Displays the contents of one resource	GET	/resource/:id
Create resource	POST	/resource
Modify resource	PUT	/resource/:id
Delete resource	DELETE	/resource/:id

# Our Project Structure



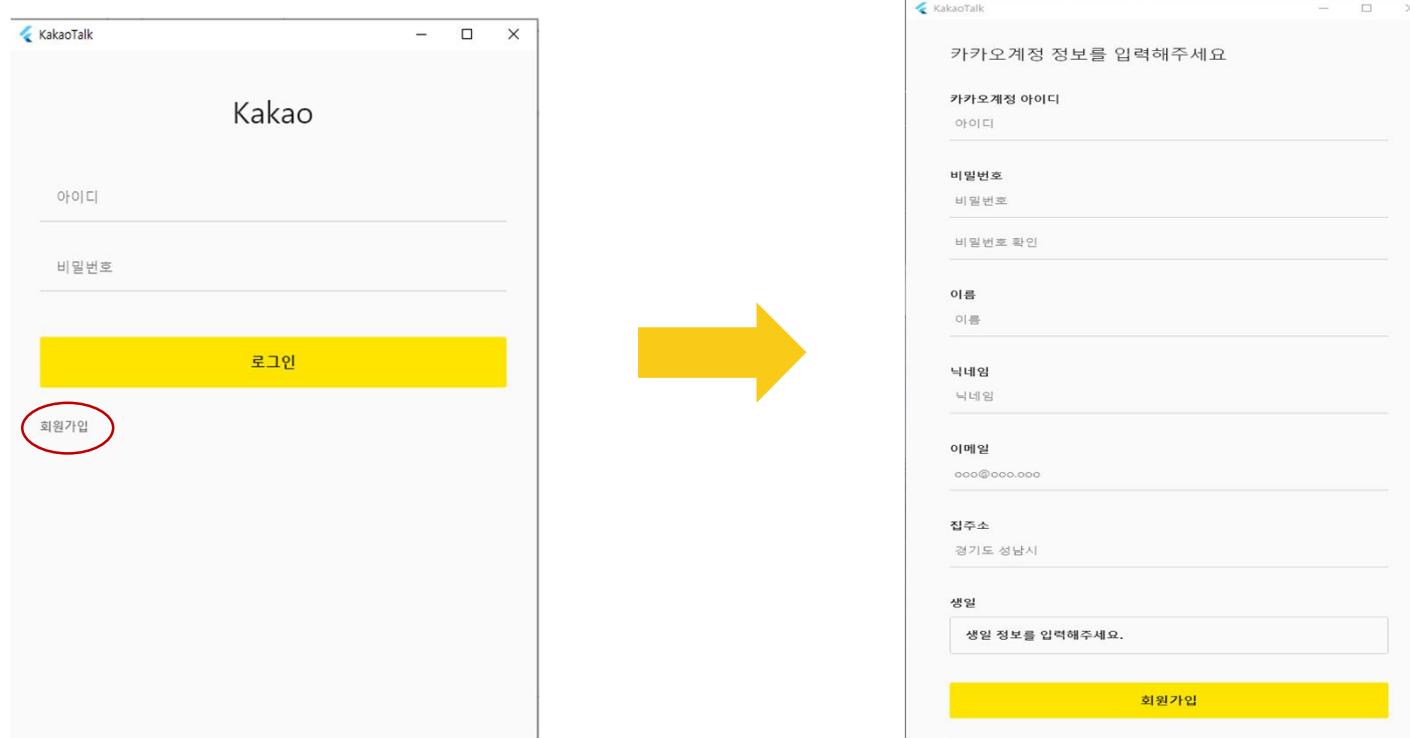
## **Part 3**

### API description



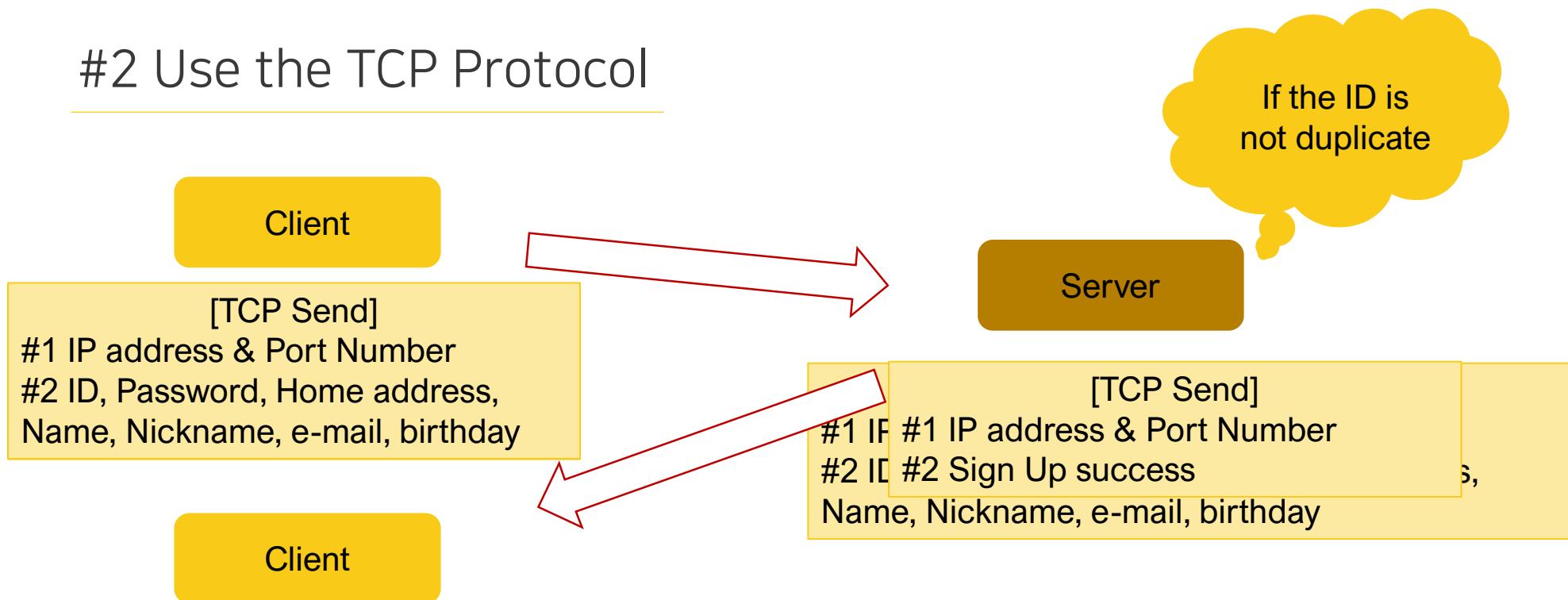
## 1. Register API

### #1 Sign Up Page



## 1. Register API

### #2 Use the TCP Protocol



# 1. Register API

## #3 REST API Structure

KakaoTalk

카카오계정 아이디	gachon02
비밀번호	..... .....
이름	gachon02
닉네임	modangE2
이메일	gachon02@gachon.ac.kr
집주소	경기도 성남시
생일	2016-5-11

회원가입



Password  
Encryption  
Processing!

```
[TCP Receive]
IP:/127.0.0.1 Port#:52428
method: POST
route: register
{"birthday":"2016-5-11","pass":"8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c9","homeaddress":"경기
도 성남시","name":"gachon02","nickname":"modangE2","id":"gachon02","email":"gachon02@gachon.ac.kr"}
```

```
[TCP Send]
IP:/127.0.0.1 Port#:52428
statusCode: 200 statusMessage: OK
data:
null
```

```
[TCP Receive]
IP:/127.0.0.1 Port#:52429
method: POST
route: login
{"password":"8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92","id":"gachon02"}
```

```
[TCP Send]
IP:/127.0.0.1 Port#:52429
statusCode: 200 statusMessage: OK
data:
>{"birthday":"2016-05-11","name":"gachon02","nickname":"modangE2","bio":"","id":"gachon02","email":"gachon02@gachon.ac
.kr","isonline":true}
```

## 1. Register API

#4 User Table data set

	*	ID varchar(30)	PassWord text	*	Name text	*	EMail text	Birthday date	TellNum int	HomeAddress varchar(100)
1	123123	XSF4q0+tp2JXNwbn	123456	123456		2022-11-13	(NULL)	123456		
2	123456789	II8YtWgN35U7YNgqa	점1	11		2022-08-12	(NULL)	11		
3	Gachon	RxG81pMnK+JGkaMic	gachon	gachon@gachon.ac.kr	2012-03-02	(NULL)	경기도 성남시 수정구 성			

A yellow arrow points from the text "Encryption using SHA256-CBC." to the "PassWord" column header in the table.

Encryption using SHA256-CBC.

## 2. Unregister API

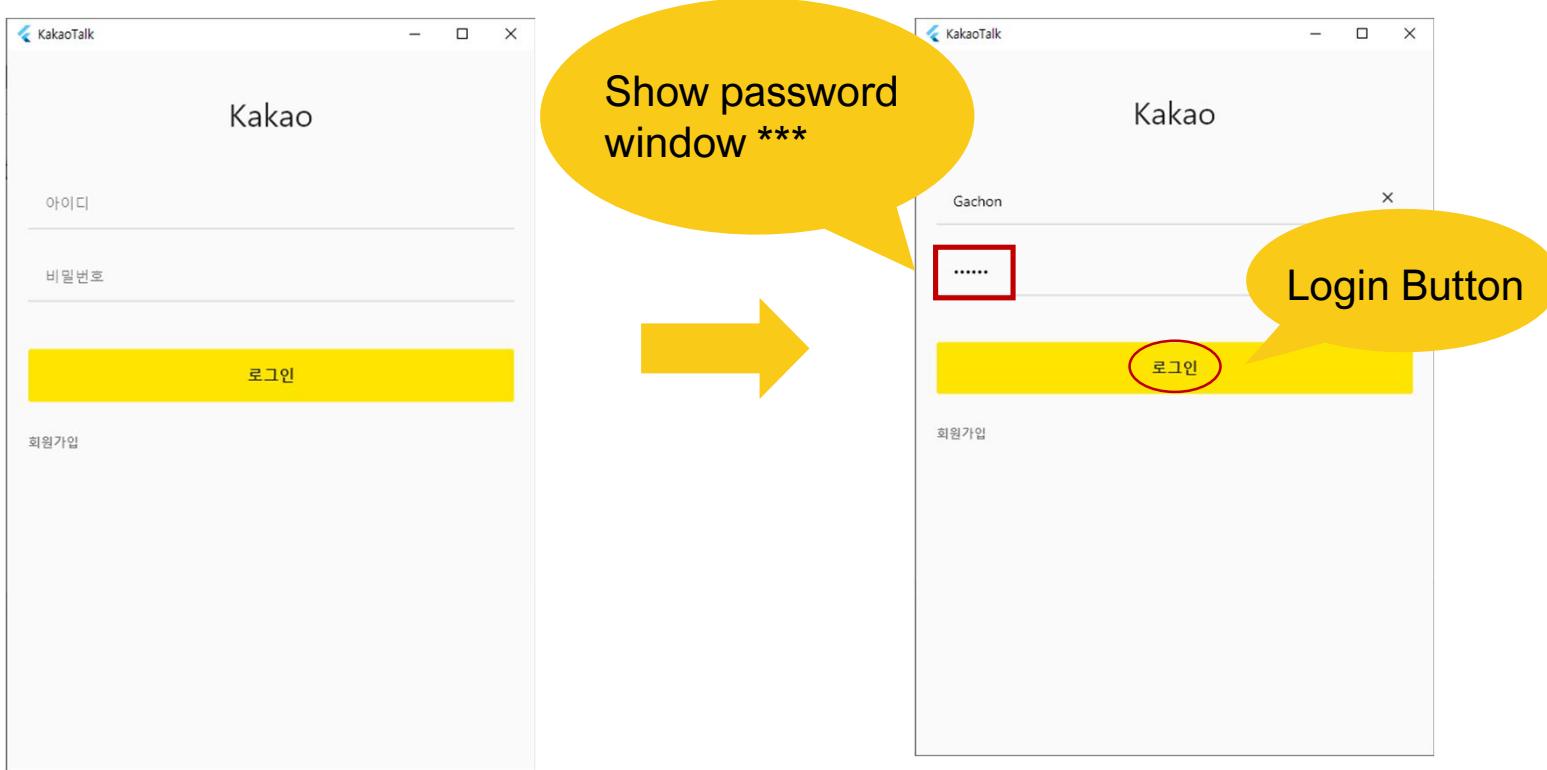
4. Unregister confirmed

```
[TCP Receive]
IP:/127.0.0.1 Port#:52036
method: POST
route: login
{"password": "230073977b57bd2aa9250b600ec2a4113a8525f3f555034c7a501ad5b4c57833", "id": "gachon08"}
```

```
[TCP Send]
IP:/127.0.0.1 Port#:52036
statusCode: 2 statusMessage: Please sign up for membership first
data:
null
```

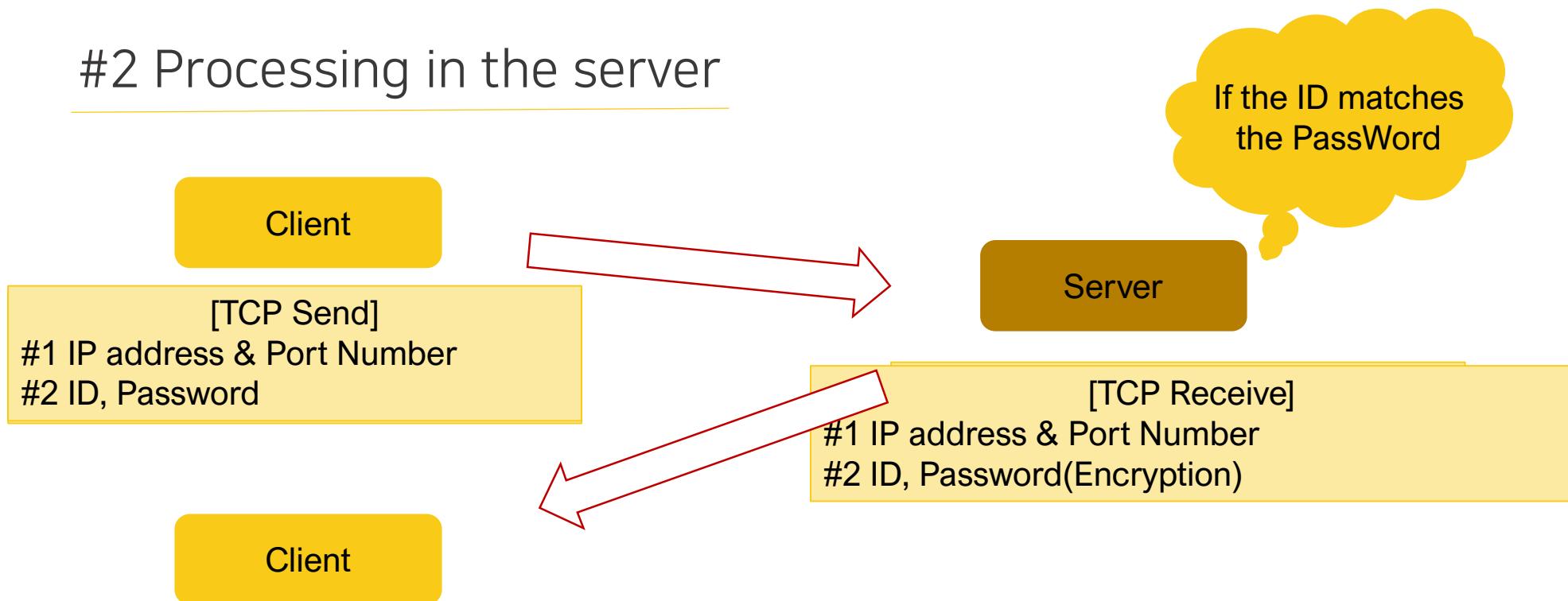
### 3. Login API

#### #1 Implement Login Window



### 3. Login API

#### #2 Processing in the server



## 3. Login API

### #3 REST API Structure

1. Login connection

```
[TCP Receive]  
IP::127.0.0.1 Port#:54618  
method: POST  
route: login  
{"password":"03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4","id":"gachon01"}
```

2. Find matching ID

```
[TCP Send]  
IP::127.0.0.1 Port#:54618  
statusCode: 200 statusMessage: OK  
data:  
{"birthday":"2022-09-14","name":"gachon01","nickname":"modangE1","bio":"","id":"gachon01","email":"gachon01@gachon.ac.kr","isonline":true}
```

3. Compare to encrypted password

```
[TCP Receive]  
IP::127.0.0.1 Port#:54619  
method: CONNECT  
route: userConnect  
{"userID":"gachon01"}  
-----  
[TCP Send]  
IP::127.0.0.1 Port#:54619  
statusCode: 200 statusMessage: OK  
data:  
null
```

4. Login connection

```
[TCP Receive]  
IP::127.0.0.1 Port#:54626  
method: GET  
route: friendList  
{"id":"gachon01"}  
-----  
[TCP Send]  
IP::127.0.0.1 Port#:54626  
statusCode: 300 statusMessage: Not Founded User Friend Data  
data:  
null
```

## 3. Login API

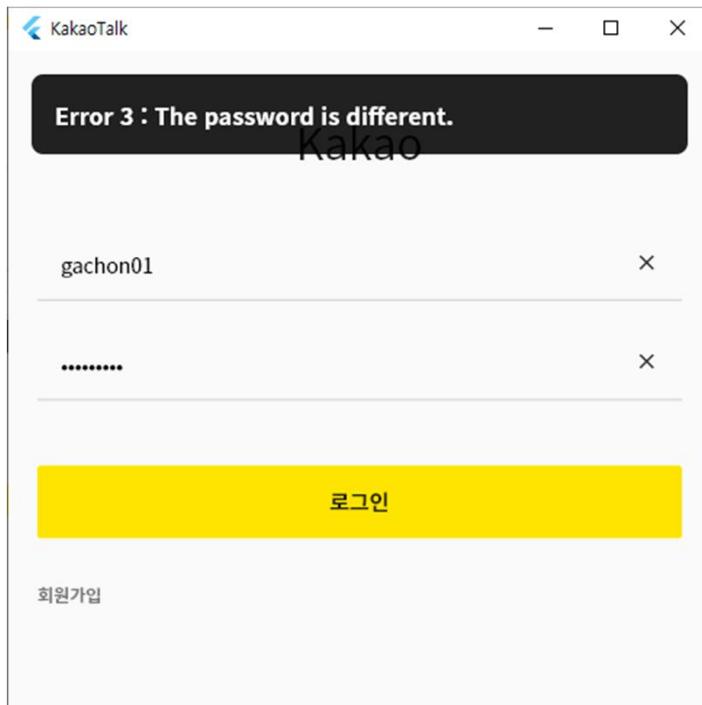
### #3 REST API Structure

ID	NickName	StatusMessage	ConnectionStatus	ResentlyConnectionTime	NumberofLogins	ResentlyLogOutTime	profile_image_id	profile_background_id
z	z		online	2022-12-14 13:27:33	2	2022-12-14 13:27:31	NULL	NULL

ID	NickName	StatusMessage	ConnectionStatus	ResentlyConnectionTime	NumberofLogins	ResentlyLogOutTime	profile_image_id	profile_background_id
z	z		offline	2022-12-14 13:27:33	2	2022-12-14 13:27:57	NULL	NULL

### 3. Login API

#### #4 Login Error Processing (Password is different)

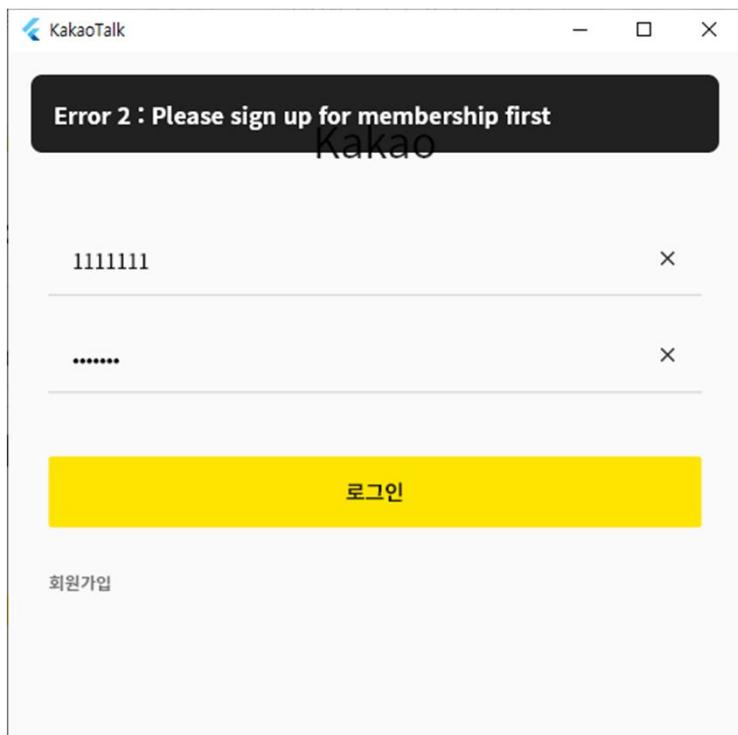


```
[TCP Receive]  
IP:/127.0.0.1 Port#:55179  
method: POST  
route: login  
{"password":"15e2b0d3c33891ebb0f1ef609ec419420c20e320ce94c65fbc8c3312448eb225","id":"gachon01"}  
-----  
[TCP Send]  
IP:/127.0.0.1 Port#:55179  
statusCode: 3 statusMessage: The password is different.  
data:  
null
```

Error!

### 3. Login API

#### #4 Login Error Processing (ID that doesn't exist)



```
[TCP Receive]  
IP:/127.0.0.1 Port#:54902  
method: POST  
route: login  
{"password": "2558a34d4d20964ca1d272ab26ccce9511d880579593cd4c9e01ab91ed00f325", "id": "1111111"}  
-----  
[TCP Send]  
IP:/127.0.0.1 Port#:54902  
statusCode: 2 statusMessage: Please sign up for membership first  

```

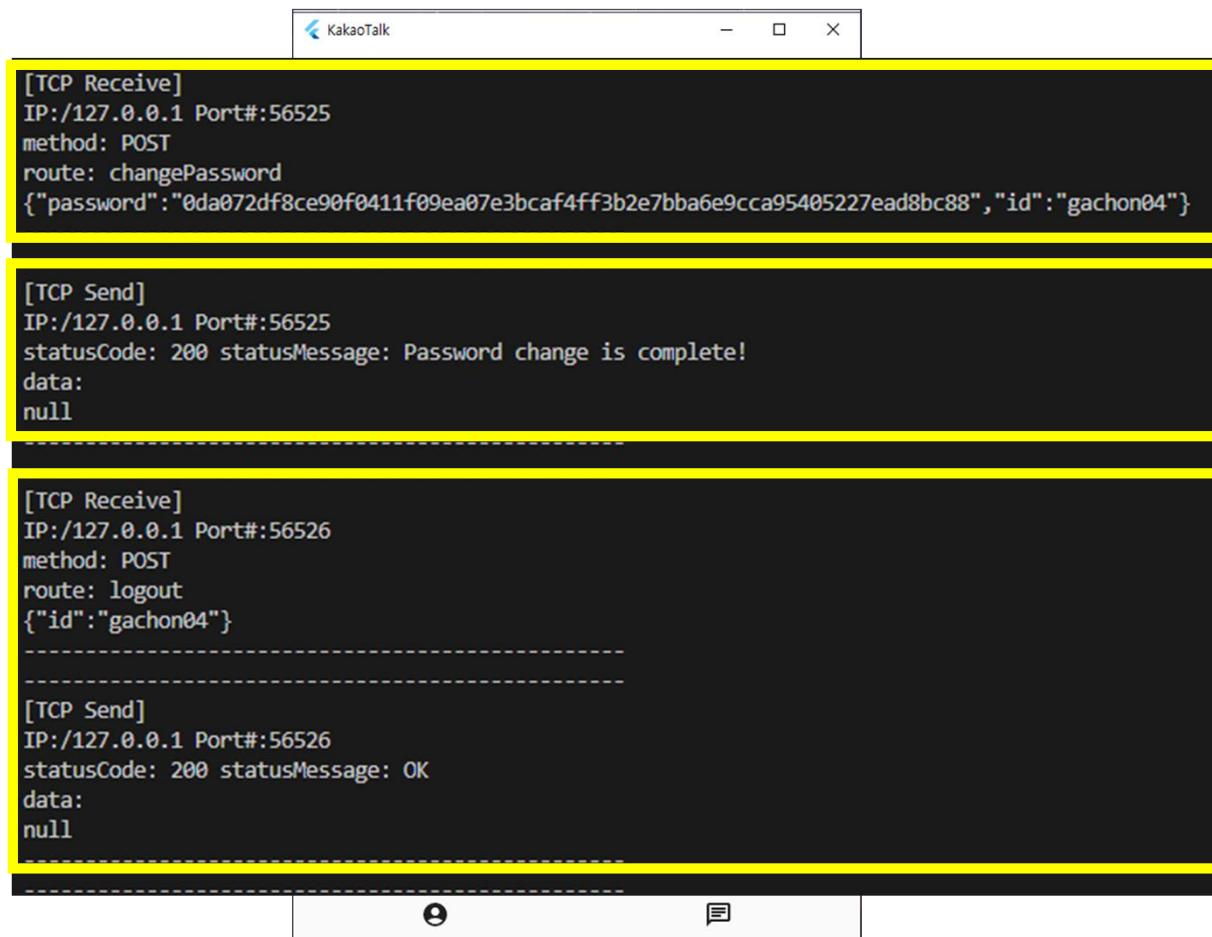
Error!

## 4. Change password API

1. Client's change password connection

2. Server's change password complete

3. Change password successful & log out



The image shows a KakaoTalk window with a dark theme. It displays three messages related to a TCP connection:

- [TCP Receive]**  
IP:/127.0.0.1 Port#:56525  
method: POST  
route: changePassword  
{"password": "0da072df8ce90f0411f09ea07e3bcfa4ff3b2e7bba6e9cca95405227ead8bc88", "id": "gachon04"}
- [TCP Send]**  
IP:/127.0.0.1 Port#:56525  
statusCode: 200 statusMessage: Password change is complete!  
data:  
null
- [TCP Receive]**  
IP:/127.0.0.1 Port#:56526  
method: POST  
route: logout  
{"id": "gachon04"}  
-----  
**[TCP Send]**  
IP:/127.0.0.1 Port#:56526  
statusCode: 200 statusMessage: OK  
data:  
null

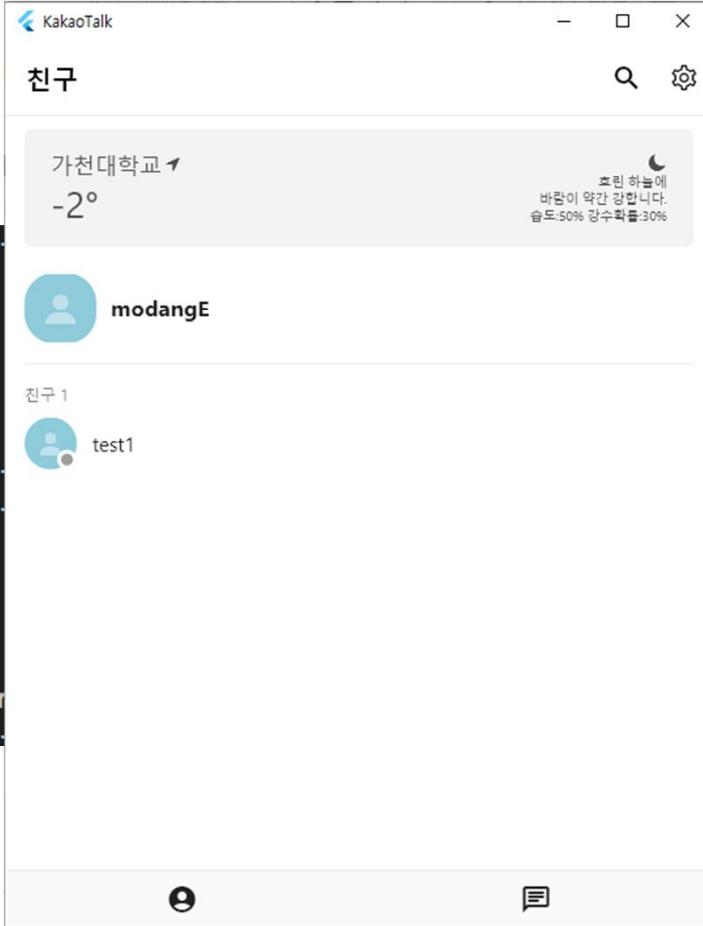
## 5. Add friend API

The screenshot shows three windows. On the left, a KakaoTalk search result for 'test1' shows a contact icon and the name 'test1'. In the center, a search for '888' shows a message '검색 결과가 없습니다.' (No results found). On the right, a terminal window displays a TCP connection between IP:127.0.0.1 Port#:51257 and IP:127.0.0.1 Port#:51619. The client sends a GET request for 'UpdateMyData' with id:888. The server responds with an error message: 'statusMessage: Not Founded User Data' and 'data: null', which is circled in yellow.

```
[TCP Receive]
IP:/127.0.0.1 Port#:51257
method: GET
route: UpdateMyData
{"id":"9"}
-----
[TCP Send]
IP:/127.0.0.1 Port#:51257
statusCode: 200 statusMessage: OK
data:
{"birthday":"2022-08-12","name":"점1","nickname":"test1","bio":"","id":9,"online":false}
-----
[TCP Receive]
IP:/127.0.0.1 Port#:51619
method: GET
route: UpdateMyData
{"id":"888"}
-----
[TCP Send]
IP:/127.0.0.1 Port#:51619
statusCode: 2 statusMessage: Not Founded User Data
data:
null
```

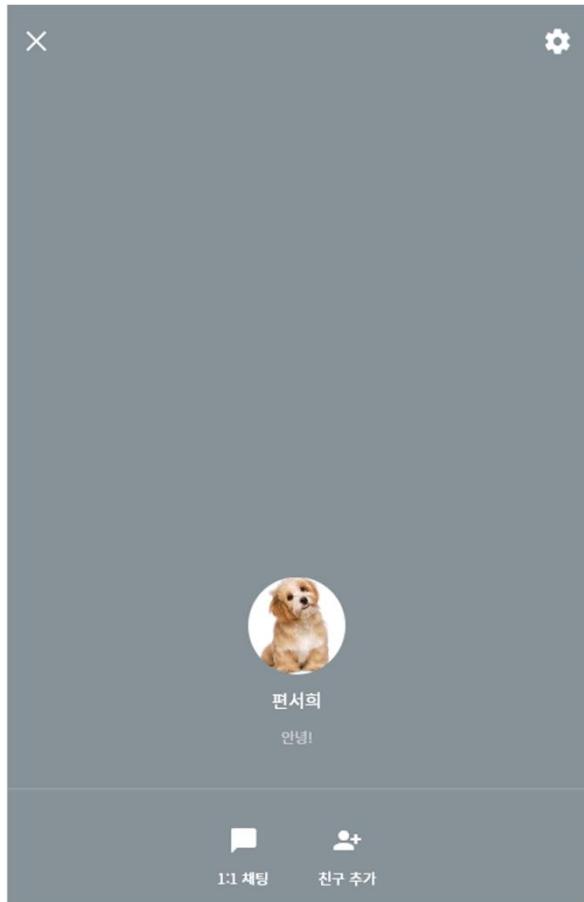
## 5. Add friend API

```
[TCP Receive]
IP:/192.168.0.1 Port#:8290
method: GET
route: friendList
{"id":"q"}  
-----  
[TCP Send]
IP:/192.168.0.1 Port#:8290
statusCode: 200 statusMessage: OK
data:  
{"datas": [{"birthday": "2022-02-14", "name": "modangE", "tel": "123456789", "email": "a", "isonline": false}], "count": 1}
```



The screenshot shows the KakaoTalk application window titled '친구' (Friends). At the top, there is a weather forecast for Gachon University: '가천대학교 ↗ -2°' with a moon icon and text '흐린 하늘에 바람이 약간 강합니다. 습도:50% 강수 확률:30%'. Below the forecast, the friend list displays one friend: 'modangE' with a blue profile picture. The friend count is shown as '친구 1' and the friend's name is 'test1'.

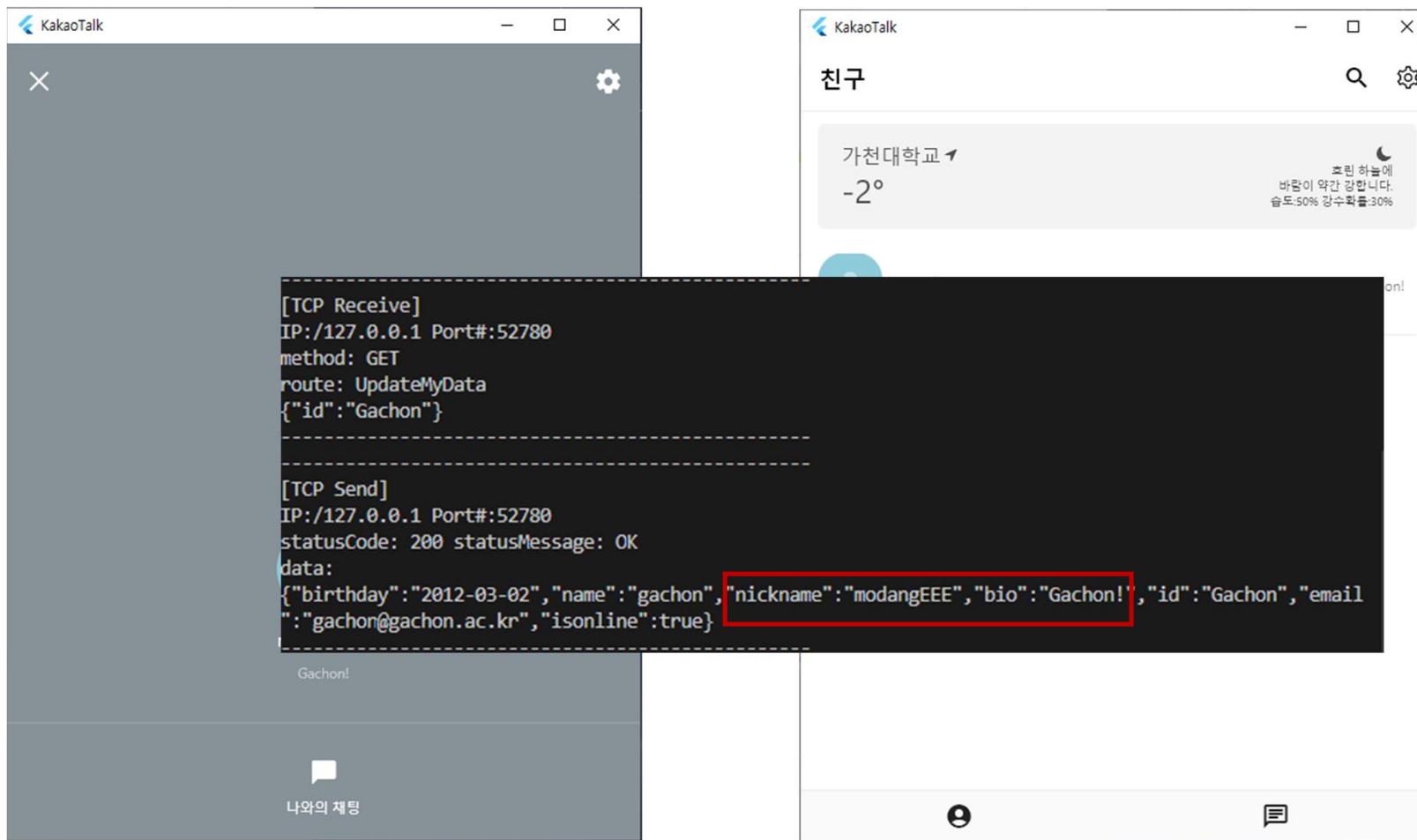
## 6. Delete friend API



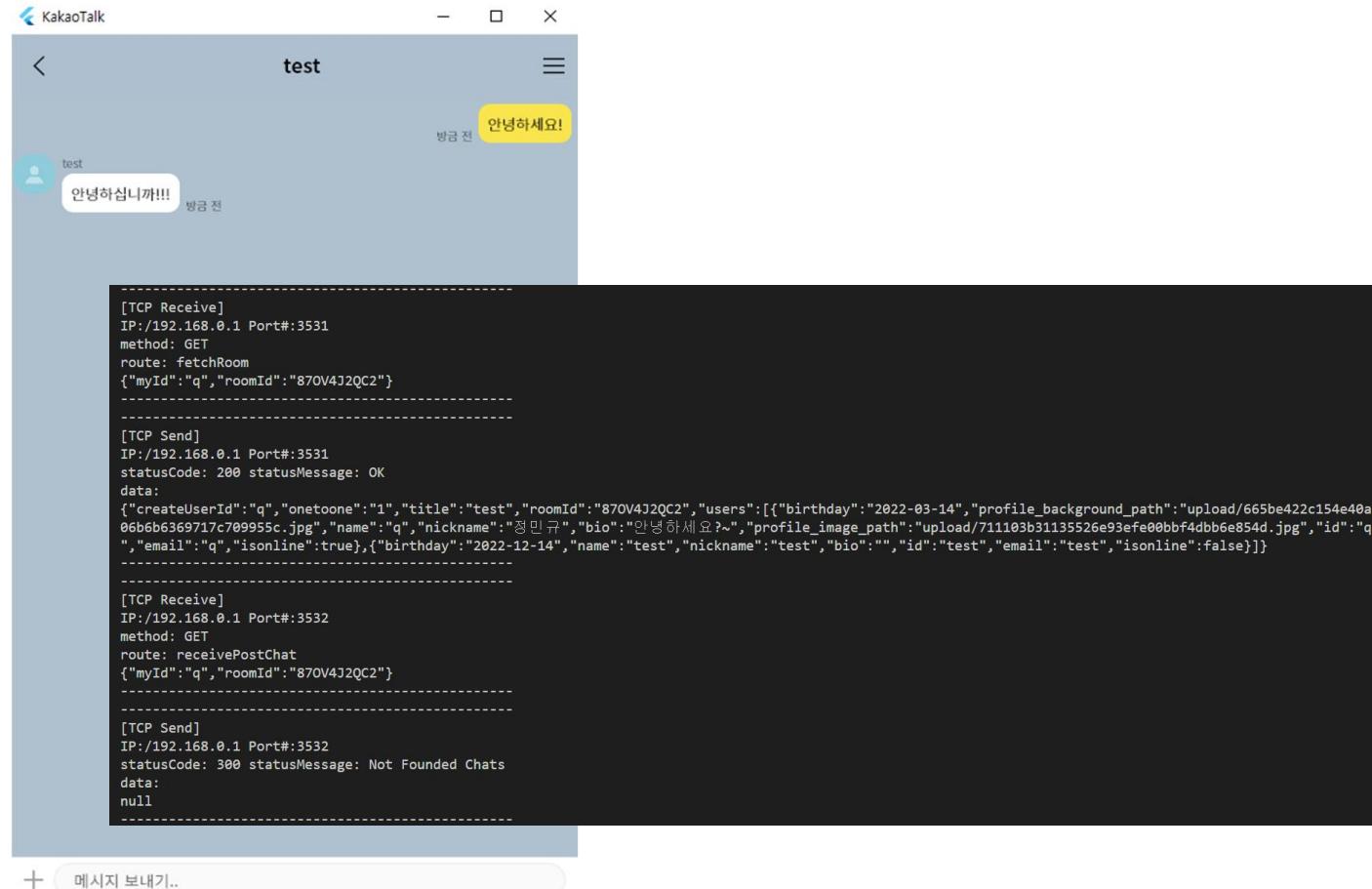
```
[TCP Receive]  
IP:/192.168.0.1 Port#:3444  
method: POST  
route: deleteFriend  
{"myId":"q","friendId":"1234"}  
-----
```

```
[TCP Send]  
IP:/192.168.0.1 Port#:3444  
statusCode: 200 statusMessage: OK  
data:  
null  
-----
```

## 7. Change statusMessage, nickName API



## 8. Create room API



## 9. Invite friend in chatting room API

```
[TCP Receive]  
IP:/192.168.0.1 Port#:3703  
method: POST  
route: createRoom  
{"myId":"q","onetoone":0,"ids":["q","test1234","test"]}  
  
-----  
  
[TCP Send]  
IP:/192.168.0.1 Port#:3703  
statusCode: 200 statusMessage: OK  
data:  
{"roomId":"11H99A6GV0"}
```



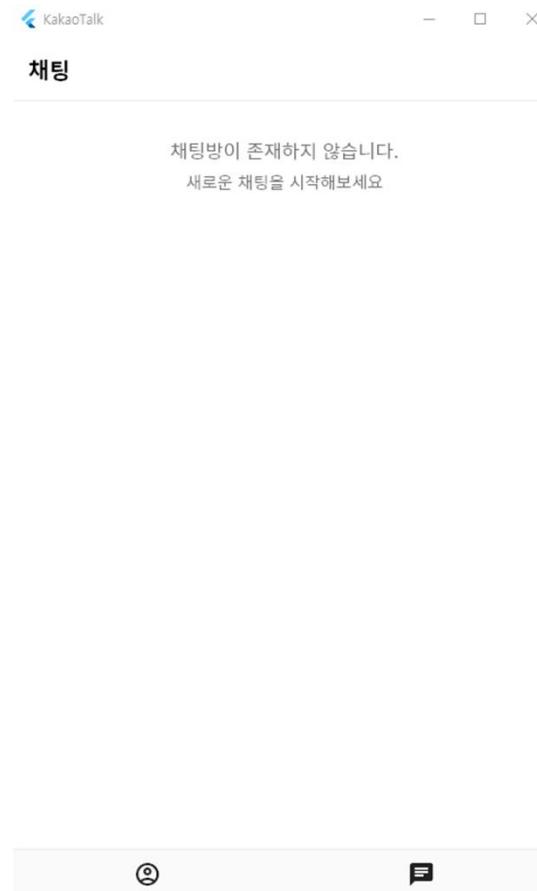
In a one-on-one room, the invitation makes a new room.

```
[TCP Receive]  
IP:/192.168.0.1 Port#:3919  
method: POST  
route: InvitePeople  
{"Id":"123456789","roomId":"11H99A6GV0"}  
  
-----  
  
[TCP Send]  
IP:/192.168.0.1 Port#:3919  
statusCode: 200 statusMessage: OK  
data:  
null
```

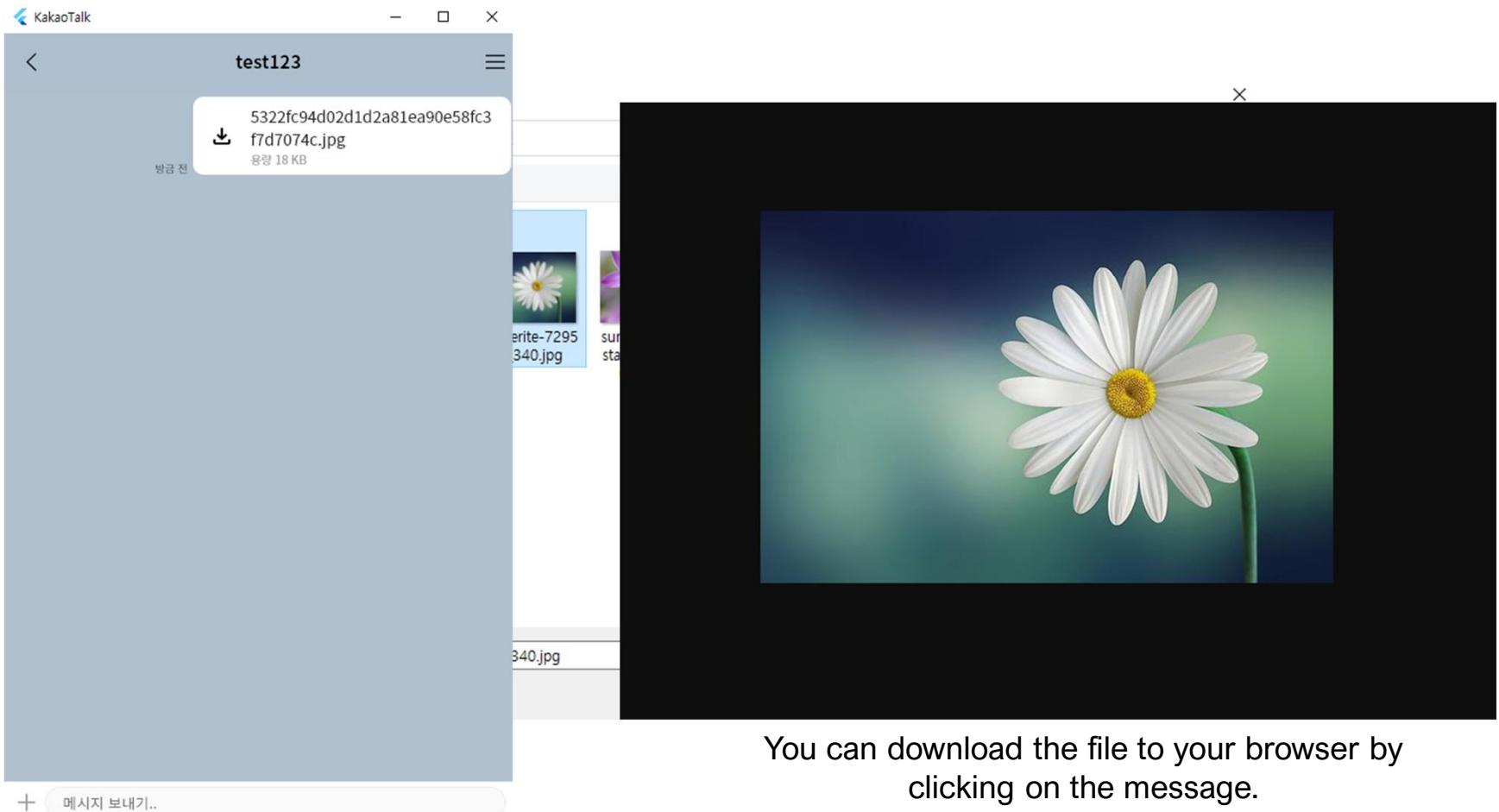
In rooms where many exist, the invitation continues with an additional member in the existing room.

## 10. Exit room API

```
-----  
[TCP Receive]  
IP:/192.168.0.1 Port#:4044  
method: POST  
route: ExitRoom  
{"Id":"q", "roomId":"11H99A6GV0"}  
-----  
-----  
[TCP Send]  
IP:/192.168.0.1 Port#:4044  
statusCode: 200 statusMessage: OK  
data:  
null  
-----
```



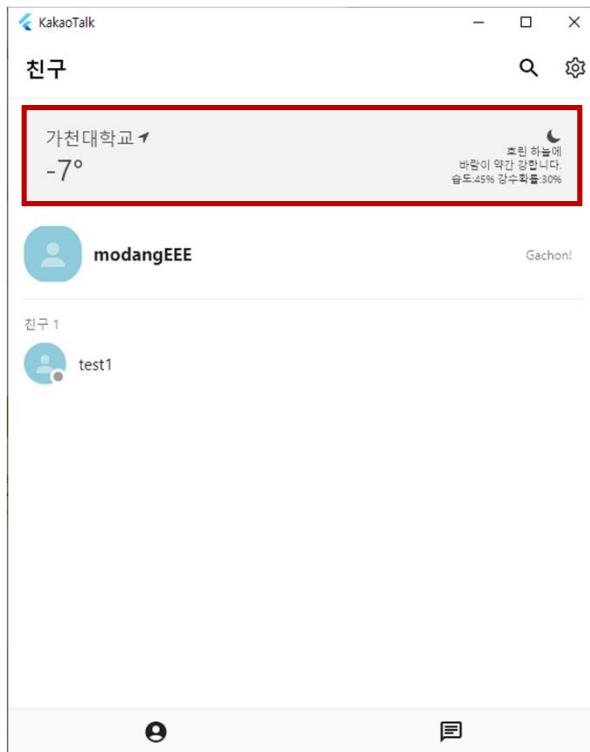
## 11. File Transfer API



You can download the file to your browser by  
clicking on the message.  
(Node.js)

## 12. Using public data

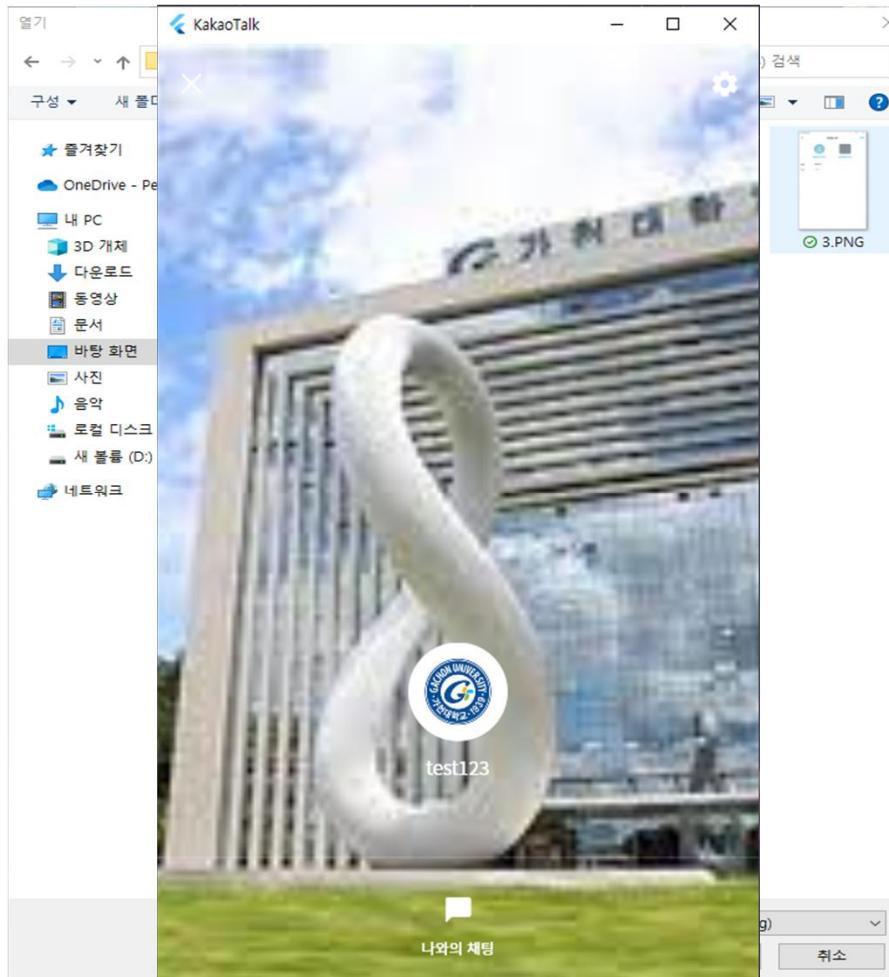
### #1 Today's weather



```
flutter: {시간대: 낮, 오늘온도: -5, 날씨: 화창, 강수형태: , 풍속: 바람이 약합니다., 하늘형태: 구름이 있고, 강수확률: 20, 습도: 30, 시간: 1400, 이미지: d_d_3.jpg}
```

HTTP communication within Flutter receives data from the public data API.

## 13. Change profile background API



## 14. Live chat API

```

[TCP Receive]
IP:/192.168.0.1 Port#:14640
method: POST
route: sendChat
{"myId":"q","message":"안녕하세요","roomId":"MULGBZZN92"}

-----
[TCP Send]
IP:/192.168.0.1 Port#:14640
statusCode: 200 statusMessage: OK
data:
null

```

```

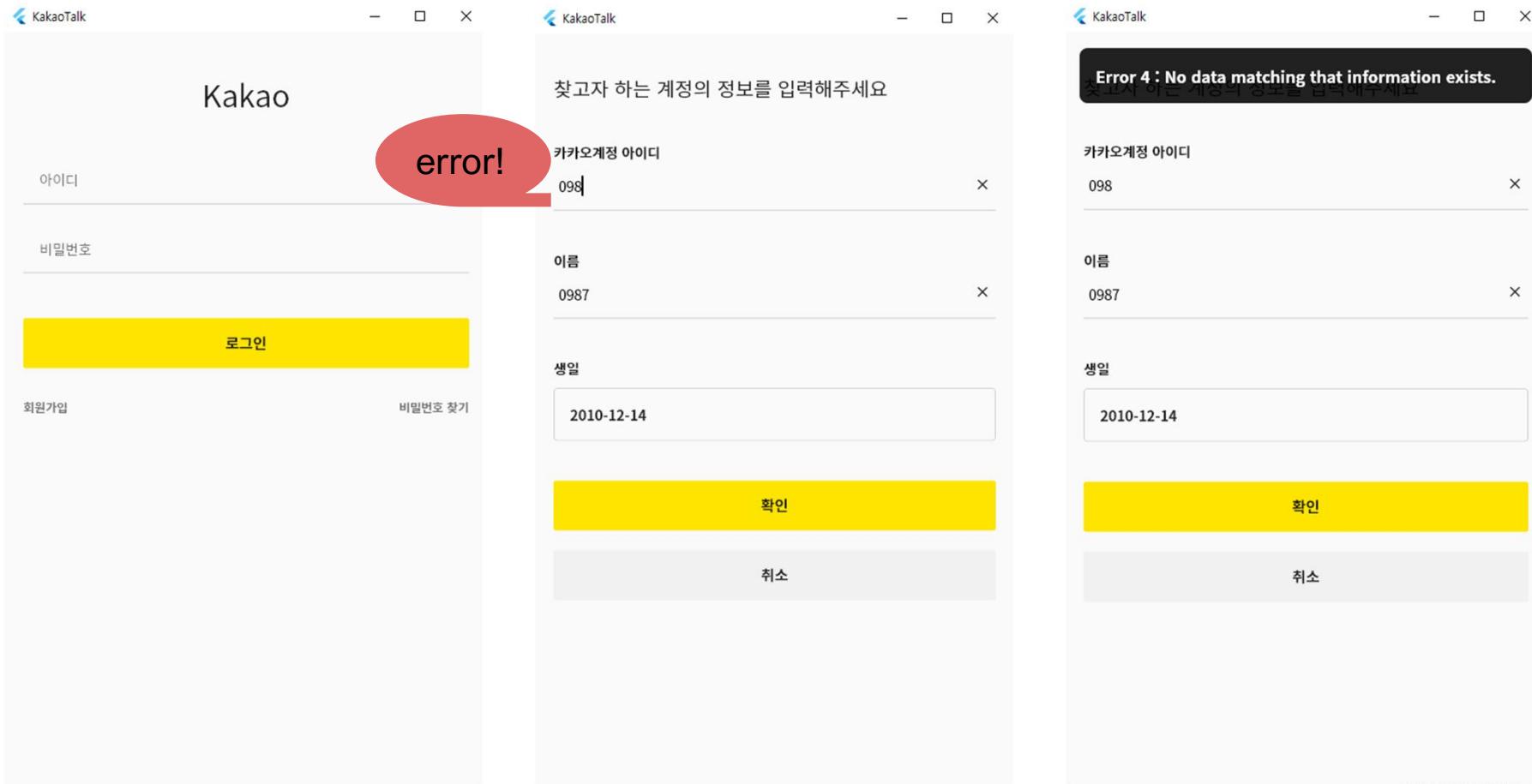
[TCP Receive]
IP:/192.168.0.1 Port#:14592
method: GET
route: receivePostChat
{"myId":"q","roomId":"MULGBZZN92"}

-----
[TCP Send]
IP:/192.168.0.1 Port#:14592
statusCode: 200 statusMessage: OK
data:
{"datas": [{"created_at": "2022-12-14 14:36:15", "message": "안녕하세요", "userid": "1234"}, {"created_at": "2022-12-14 14:23:30", "message": "@a(d|59d566b0c32340d1b0caf4697b2af7fb3f21.jpg|80 KB|upload/59d566b0c32340d1b0caf4697b2af7fb3f21.jpg", "userid": "q"}, {"created_at": "2022-12-14 14:23:11", "message": "@a(d|fa802df5560ba5d417fd1312f071001fc8cc.docx|841 KB|upload/fa802df5560ba5d417fd1312f071001fc8cc.docx", "userid": "q"}, {"created_at": "2022-12-14 14:23:01", "message": "@a(d|c8d037764bc6c80f0cde55f92763153b5183.jpg|6 KB|upload/c8d037764bc6c80f0cde55f92763153b5183.jpg", "userid": "q"}, {"created_at": "2022-12-14 14:01:25", "message": "안녕하세요!", "userid": "1234"}, {"created_at": "2022-12-14 14:01:17", "message": "안녕하세요", "userid": "q"}, {"created_at": "2022-12-14 12:42:51", "message": "@a(d|336a1703262de818ccac9cf130d2571308e8.jpg|4 KB|upload/336a1703262de818ccac9cf130d2571308e8.jpg", "userid": "1234"}], "users": {"q": {"nickname": "정민규", "profile_image_path": "upload/ab11af4403352afeca3a137913110aca0a67.jpg"}}}

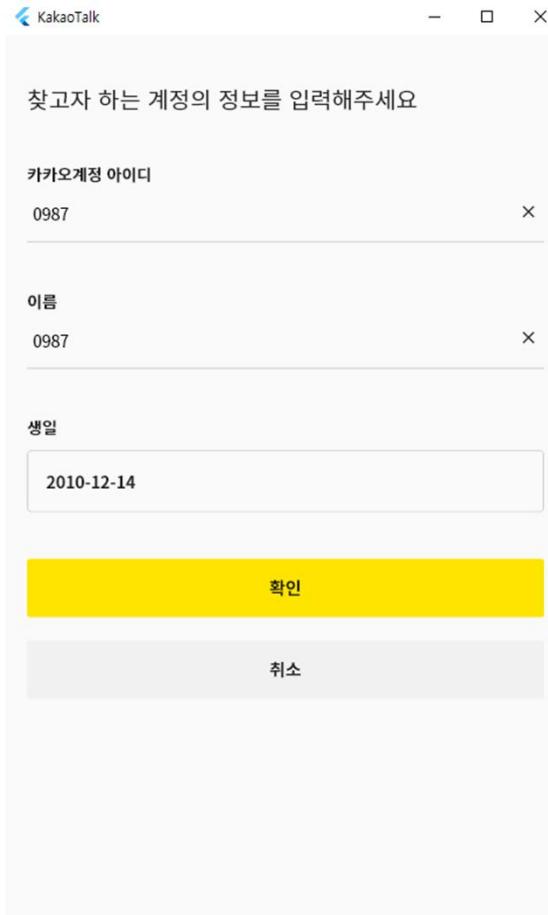
```

↑

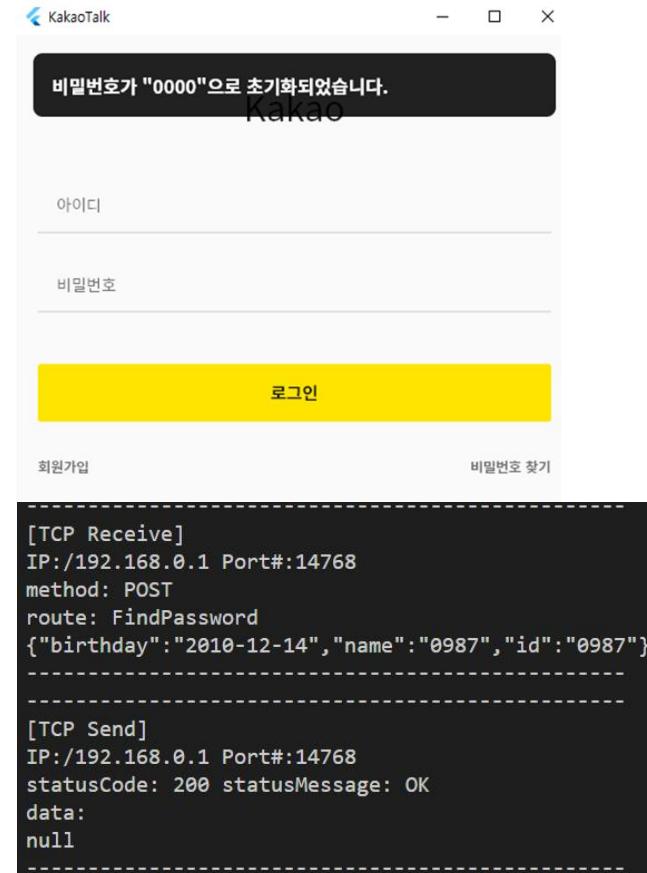
## 15. Change Password

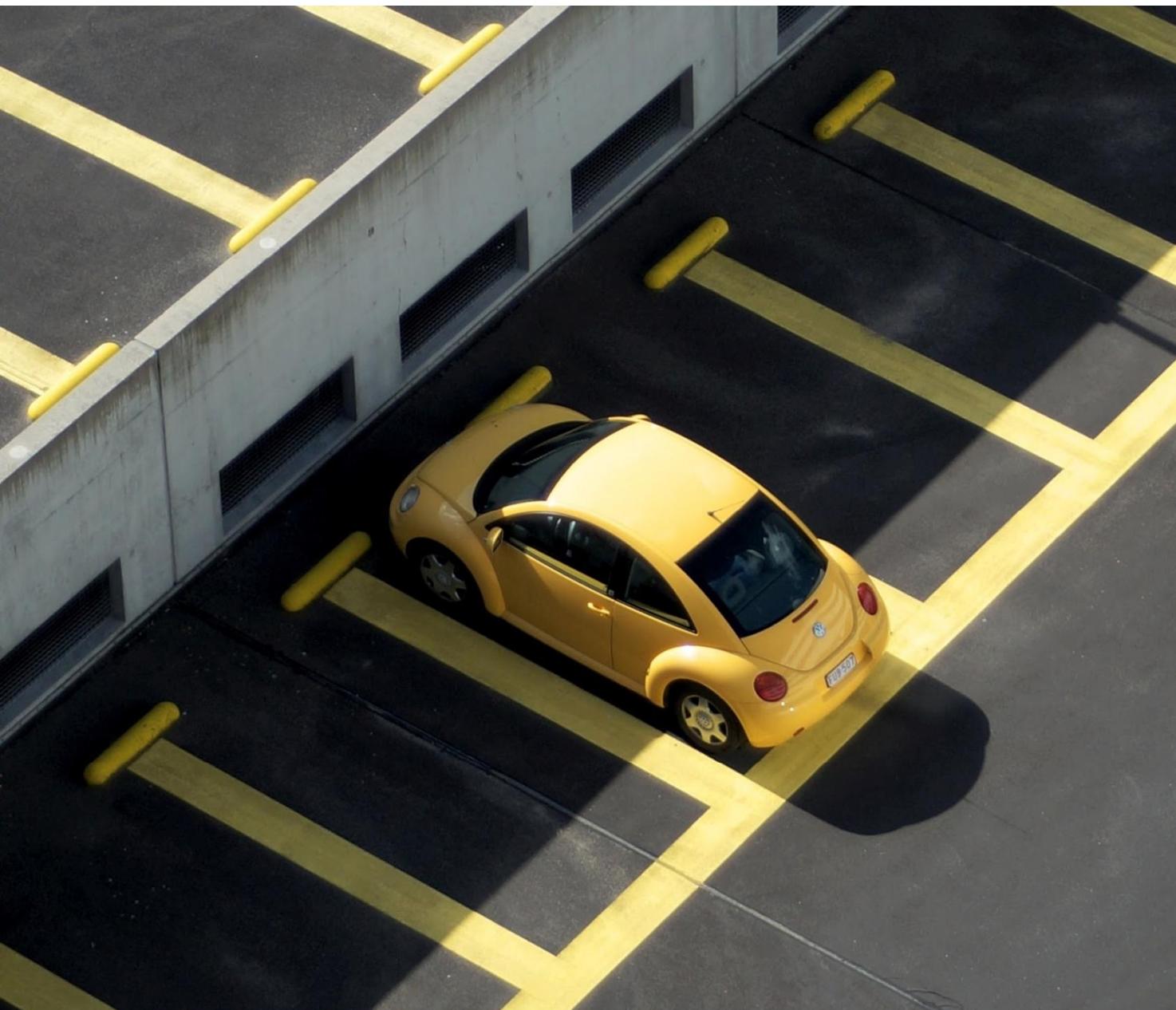


## 15. Change Password



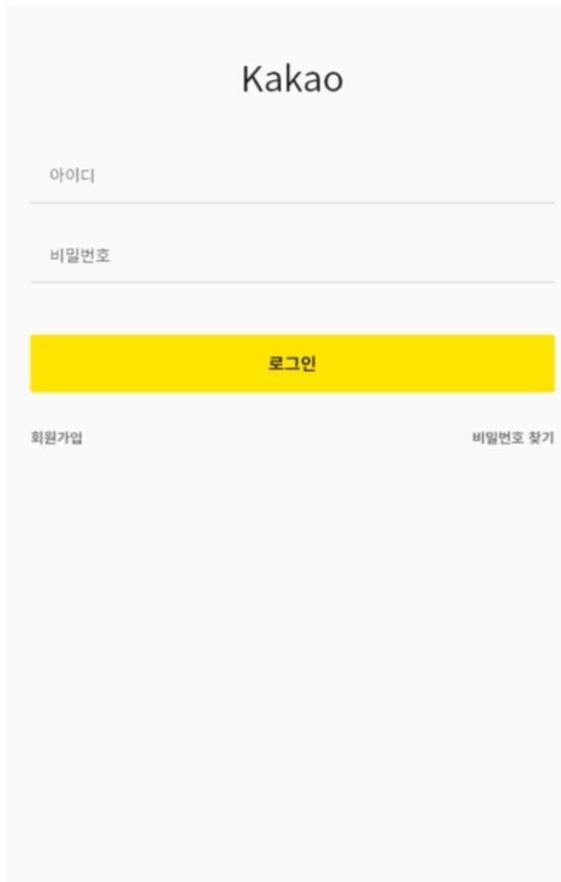
If right information





## Part 4 Result

## Demo video



# Multi-threaded

```
private static class ReceiveUDPThread implements Runnable {
    @Override
    public void run() {
        try {
            DatagramSocket ds = new DatagramSocket(port: 9998);
            while (true) {
                System.out.println(x: "UDP Waiting for a packet reception..");
                Request request = UDP.receive(ds);
                pool.execute(new RequestThread(request, new Network(ds, request.method)));
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

# Multi-threaded

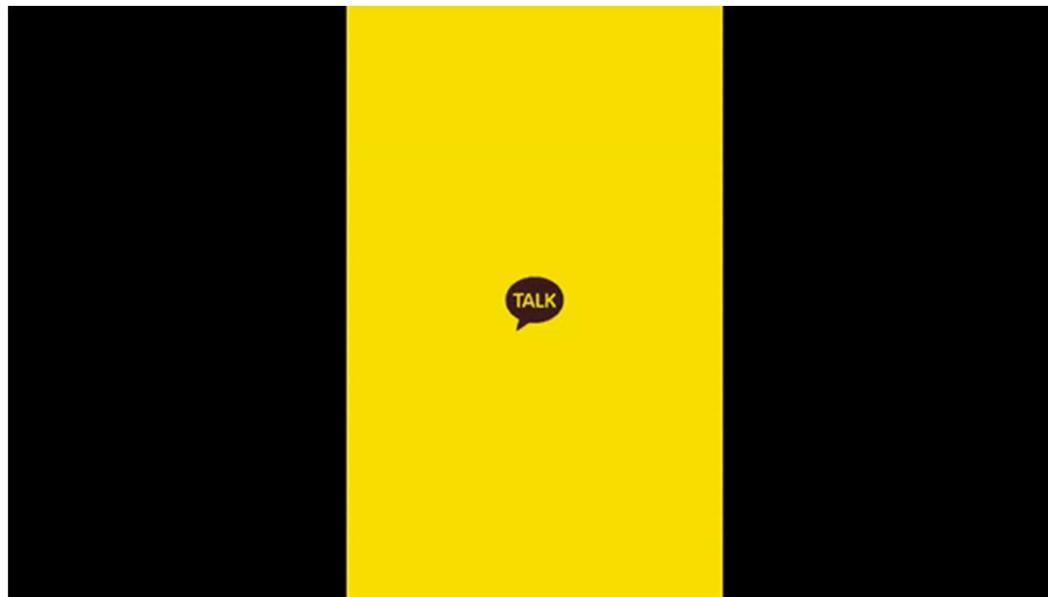
```
private static class ReceiveTCPThread implements Runnable {
    @Override
    public void run() {
        Socket socket = null;
        try {
            ServerSocket listener = new ServerSocket(port: 9997);
            System.out.println(x: "TCP Waiting for a packet reception..");
            while (true) {
                socket = listener.accept();
                Request request = TCP.receive(socket);
                if (request.method.equalsIgnoreCase(anotherString: "CONNECT"))
                    pool.execute(new TcpThread(request, new Network(socket, request.method)));
                else
                    pool.execute(new RequestThread(request, new Network(socket, request.method)));
            }
        } catch (Exception e) {
            if (socket != null)
                try {
                    socket.close();
                } catch (IOException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

# Multi-threaded

```
Run | Debug
public static void main(String[] args) throws Exception {
    String jdbc_url = "jdbc:mysql://43.200.206.18:3306/networkDB";
    // String jdbc_url = "jdbc:mysql://127.0.0.1:3306/networkDB";
    con = DriverManager.getConnection(jdbc_url, user: "root", password: "gachon");
    updatestmt = con.createStatement();
    pool = Executors.newFixedThreadPool(nThreads: 40);
    pool.execute(new ReceiveUDPThread());
    pool.execute(new ReceiveTCPThread());
}
```

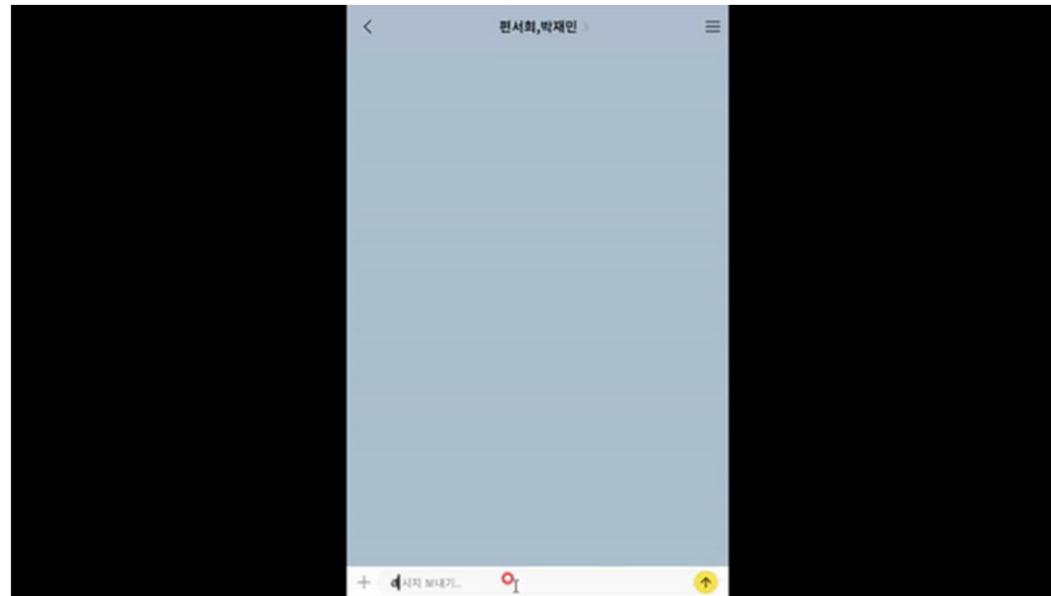
Multi-thread allows for fast  
chatting

## Verify server connectivity



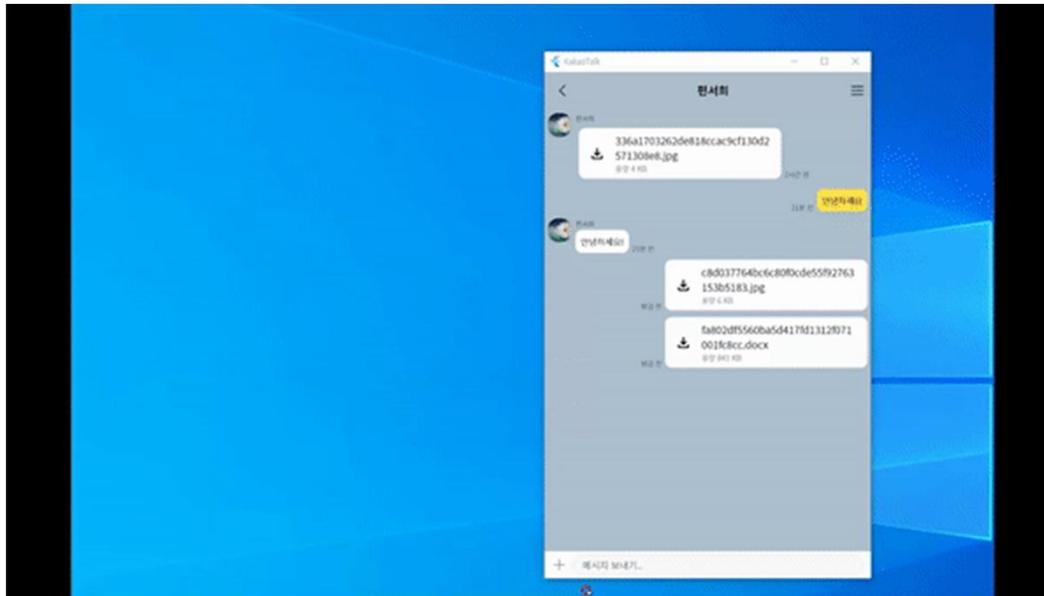
Check if the connection with the server is okay. If you can't connect to the server, you can't escape the Splash Page indefinitely.

# Multiple Users in a Room



BroadCast and MultiThread allow several people to continue conversations at a high speed in one room at the same time.

## File Transfer



You can upload files using node.js Upload Server and receive files through a browser using the URL of the uploaded file.

# Open Source

---

- <https://cloud.google.com/spanner/docs/use-oss-jdbc?hl=ko>
- <https://pub.dev/packages/get/versions>
- [https://pub.dev/packages/udp\(score](https://pub.dev/packages/udp(score)
- [https://pub.dev/packages/cached\\_network\\_image](https://pub.dev/packages/cached_network_image)
- <https://pub.dev/packages/dio>
- <https://pub.dev/packages/http>
- <https://pub.dev/packages/intl/versions/0.17.0-nullsafety.2/install>
- <https://pub.dev/packages/crypto>
- [https://pub.dev/packages/file\\_picker](https://pub.dev/packages/file_picker)
- [https://pub.dev/packages/url\\_launcher](https://pub.dev/packages/url_launcher)

**Thank you for listening**