Interface

Code Request Summary

The server deciphers each request through a series of codes:

DEFAULT\_USER\_LEVEL = 0

SERVER\_PORT = 80

REGISTER\_ACTION = 100

LOGIN\_ACTION = 101

MESSAGE\_ACTION = 102

LOGIN\_SUCCESS\_CODE = 200

LOGIN\_USER\_NON\_EXIST\_CODE = 201

LOGIN\_EXIST\_CODE = 202

RESISTER\_SUCCESS = 300

REGISTER\_FAIL = 301

GET\_FRIENDS = 400

Server Login Service

The interface between the server and client which allows clients send login request to server.

When user logs in successfully, the user will be assigned a token, which is a unique ID of current session. Whenever the user sends an action to the server, the token must be attached as the authentication.

Message

{

username: string,

password: string,

action: int

}

Return

{

type: string,

code: int,

id: string

}

Example Request

Send

{

“username”: “test”,

“password”: “donttellothers”,

“action”: 101

}

Return

{

"type": "success",

"code": 200,

"id": "9x8869x31134x7906x6x54474x21x18xxx90857x"

}

Server Messaging Service

The interface between the server and client which allows clients send message to others.

The user must be assigned a token first, or log in will be failed.

Message

{

id: string,

action: int

to: string

message: string

}

Return

{

type: string,

code: int,

}

Example Request

Send

{

“id”: “9x8869x31134x7906x6x54474x21x18xxx90857x”

“action”: 201

“to”: “zirenxiao”

“message”: “have you seen my beer Tibbers?”

}

Return

{

"type": "success",

"code": 200,

}

Client Register Service

The interface between the server and client which allows clients to create a new account in database.

Message

{

username: string,

password: string,

email: string,

name: string,

action: int

}

Return

{

type: string,

code: int,

}

Example Request

Send

{

“username”: “test”,

“password”: “donttellothers”,

“email”: “[auser@catchmeifyoucan.com](mailto:auser@catchmeifyoucan.com)”,

“name”: “tester”,

“action”: 100

}

Return

{

"type": "success",

"code": 201

}

Friend List Request Service

The interface between the server and client which allows clients to ask the server to send a JSON list of all the user’s friends. Last online date will be formatted by YYYY-MM-DD HH:MI:SS (24 hours). Note: This obviously can’t send pictures.

Request

{

“id”: “string”,

“action”: int

}

Return

{

"friends" : [

{

"user":string,

"status":"string",

“lastonline”:“string”

},

{

"user":string,

"status":"string",

“lastonline”:“string”

},

...

]

}

Example Request

Send

{

“id”: “zirenxao”

“action”: 400

}

Return

{

"friends" : [

{

"user":”dog\_man300”,

"status":"online",

“lastonline”:“n/a”

},

{

"user":”chicken\_rice”,

"status":"offline",

“lastonline”:“ 2017-08-11 19:32:03”

},

{

"user":”rupert\_pupkin”,

"status":"online",

“lastonline”:“n/a”

}

]

}