**Project Initiation Document**

**QuickChat**

Bowei Yao

July 9, 2017

QuickChat is a program that allows users to join chat rooms without authentication, simply enter a username and begin chatting!

**Tools used:** Unity3d for front end, Heroku + node.js on back end. Websocket for comm

**Rules**

**Username**: length between 3 to 10 characters, must be alphanumeric

**Room specs**: must contain the following the following properties: name, type, max\_p

Name can be between 3 to 10 characters, alphanumeric

Type can be either 1 (public), or 2 (private). If 2, then property pw must exist for password

Password can be between 6 to 12 characters, alphanumeric

Max\_p can be any of the following values: 2, 5, 10, 20

**Message**: message can contain any characters, but must be less than 256 characters

**Message types**

**Server sent messages:**

100: connection result, followed by json { “ret”: 1 }, 1 for success, 0 for failure

101: create username result, same as above

110: create room result, { “ret”: 1, “room\_id”: “24” }

111: join room result, same as above

112: leave room result, same as above

113: request room listing result, return with json of 20 most recent chatrooms

114: get room info result, return with json string containing room info, but for people array, replace the join date with people’s name

120: notify join room, json { “id”: “523”, “name”: “bowragon” }

121: notify leave room, json { “id”: “244” }

130: client send message result, same as above

131: server send message, followed by json string { “msg”: “this is an example msg”, “name”: “boweiyao” }

400: server send error, followed by json string { “err”: “this is an example error msg” }

**Client sent messages:**

201: create username, followed by json string { “name”: “boweiyao” }

210: create room, followed by json string with room specs

211: join room, followed by json string { “room\_id”: 56, “pw”: “1239fdskj” }

212: leave room, followed by empty json

213: request room listing, followed by empty json

214: get room info, followed by json string { “room\_id”: 24” }

230: client send message, followed by json string with { “msg”: “this is an example message” }

**Error message codes:**

1000: Client message is invalid

2000: Failed to create room

2100: Failed to join room

2101: Room has reached maximum number of people

2102: Incorrect password

2200: Failed to leave room

2300: Failed to get room info

2400: Failed to send message

**Appendix**

This is how objects are structured:

In room.js

**Rooms map**

{

"4" : {

"name": "myroom",

"type": 1,

"max\_p": 5,

"people": {},

},

"5": {

"name": "omgholy",

"type": 2,

"pw": "jajaqq",

"max\_p: 20",

"people": {

"12" : "1979-6-12",

"232" : "1980-7-14",

"8498" : "2010-9-12",

"783" : "2008-6-24",

}

},

"6": {

"name": "secret",

"type": 2,

"pw": "loljk",

"max\_p: 2",

"people": {

"13" : "2011-1-4",

"250" : "2005-4-16",

}

}, ...

}