

```
1 var app = require('express') ();
2 var http = require('http').Server(app);
3 var io = require('socket.io')(http);
4 var fs = require('fs');
5 var port = process.env.PORT || 80;
6
7 app.get(/^(.+)$/, function(req, res){
8   res.sendFile(__dirname + '/html' + req.params[0]);
9 });
10
11 http.listen(port, function(){
12   console.log('listening on *: ' + port);
13 });
14
15 var groups = new Array(); // Array of Group
16
17 var nextGroupID = 1; // id to serve to next new group
18
19 /**
20  User Object
21  id-Integer: The id of the the user
22  nick-String: The nickname of the user
23  pass-String: The hash of the user's password
24  status-String: status of user, "online" or "offline"
25  */
26
27 /**
28  Group Object
29  id-Integer: id of group
30  messages-Array[String]: array of messages
31  name-String: name of group
32  */
33
34 /**
35  List of Users
36  key: id-Integer
37  */
38 var users = [];
39 var nextUserID = 1; // id to serve to next new user
40
41 // Load the database into groups, and users
```

```
42 function loadDB(){
43 // load the groups users nextGroupID, and nextUserID from the db
44
45 // Database mock
46 // remove when database code is working
47 /**
48 users.push({
49 id: 0,
50 name: "Bismarck",
51 nick: "Bismarck",
52 isAdmin: true,
53 pass: "password",
54 status: "online"
55 });
56 **/
57
58 /**
59 groups.push({
60 id: 0,
61 name: "General",
62 messages: []
63 });
64 **/
65
66 fs.readFile('server/users.dat', 'utf8', function (err,data) {
67     if (err) {
68         return console.log(err);
69     }
70     var strList = data.split(/\r?\n/);
71     for(var i =0; i<strList.length;){
72         var u = {};
73         u.id = strList[i++];
74         u.name = strList[i++];
75         u.nick = strList[i++];
76         u.isAdmin = strList[i++];
77         u.pass = strList[i++];
78         u.status = "offline";
79         users.push(u);
80     }
81     nextUserID = users[users.length -1].id+1;
82 });
```

```
83
84
85 fs.readFile('server/numGroup.dat', 'utf8', function (err,data) {
86     if (err) {
87         return console.log(err);
88     }
89
90     var numGroup = data;
91
92     for(var i = 0; i<numGroup; i++) {
93         data = fs.readFileSync('server/group'+i+'.dat', 'utf8');
94         var strList = data.split(/\r?\n/);
95         var group = {};
96         group.id = i;
97         group.messages = [];
98         group.name =strList[0];
99         for(var n = 1; n<strList.length; n++){
100             group.messages.push(strList[n]);
101         }
102         groups.push(group);
103     }
104     nextGroupID = groups.length;
105 });
106 }
107
108 loadDB();
109
110
111 // User connection
112 io.on('connection', function(socket){
113     console.log('User Connected');
114
115     socket.on('disconnect', function(){
116         console.log('User Disconnected');
117     });
118
119 /**
120  send message to groups
121  */
122     socket.on('groupMessage', function(user, groupID, message){
123         if(auth(user)){
```

```
124 for (var i = 0; i<groups.length; i++){
125   if(groupID==groups[i].id){
126     groups[i].messages.push(message);
127     // db.insertData();
128   }
129 }
130 fs.appendFileSync('server/group'+groupID+'.dat', '\n'+message);
131 io.emit('groupMessage',user, groupID, message);
132 }
133 });
134
135
136 /**
137  creates a new group
138  */
139 socket.on('newGroup', function(user, groupName){
140   var newGroup = {};
141   newGroup.id = nextGroupID++;
142   newGroup.name = groupName;
143   newGroup.messages = new Array();
144   // write to DB
145   groups.push(newGroup);
146   console.log(groupName);
147   fs.writeFileSync("server/group"+newGroup.id+".dat", groupName);
148   console.log(nextGroupID);
149   fs.writeFileSync("server/numGroup.dat", nextGroupID);
150
151   io.emit('newGroup', groups);
152 });
153
154 /**
155  creates a new user
156  */
157 socket.on('newUser', function(name, pass, isAdmin, callback){
158   var newUser = {};
159   newUser.nick = name;
160   newUser.name = name;
161   newUser.isAdmin = isAdmin;
162   newUser.pass = pass;
163   newUser.status = "online";
164   //newUser.id =nextUserID++;
```

```
165 newUser.id = nextUserID++;
166 users.push(newUser);
167 addUsertoDB(newUser);
168
169 callback(newUser);
170 io.emit('updateUserList', getStrippedUsers());
171 });
172
173 function addUsertoDB(newUser){
174 // write to DB
175 fs.appendFileSync('server/users.dat', '\n'+newUser.id+'\n');
176 fs.appendFileSync('server/users.dat', newUser.name+'\n');
177 fs.appendFileSync('server/users.dat', newUser.nick+'\n');
178 fs.appendFileSync('server/users.dat', newUser.isAdmin+'\n');
179 fs.appendFileSync('server/users.dat', newUser.pass);
180 }
181
182 /**
183 gets groups and messages under it
184 **/
185 socket.on('getGroups', function(callback){
186 callback(groups);
187 });
188
189
190 /**
191 gets list of users that are online
192 **/
193 socket.on('getUsers', function(callback){
194 callback(getStrippedUsers());
195 });
196
197 /**
198 sets the status of the user
199 **/
200 socket.on('setStatus', function(user, status){
201 if(auth(user)){
202 getUserObjectByIDObject(user).status = "online";
203 }
204 io.emit('updateUserList', getStrippedUsers());
205 });
```

```
206
207 /**
208  sets the nickname of the user
209  */
210 socket.on('setNickname', function(user, nick){
211  if(auth(user)){
212    getUserObjectByIdObject(user).nick = nick;
213  }
214  io.emit('updateUserList', getStrippedUsers());
215 });
216
217 socket.on('uploadFile', function(name, file, callback){
218  console.log(name);
219  saveFile(name, file);
220  callback();
221 });
222
223 /**
224  Returns the code of the landing page
225  */
226 socket.on('getLandingCode', function(callback){
227  fs.readFile('server/landing.html', 'utf8', function (err,data) {
228    if (err) {
229      console.log(err);
230    }
231    callback(data);
232  });
233 });
234
235 /**
236  Grabs the user object and returns it.  Null if doesn't exist
237  */
238 socket.on('grabUserObjectByUserPass', function(username, password,
callback){
239  var u = getUserObjectByUserPass(username,password);
240  if(u!=null){
241
242  }
243  callback(u);
244  });
245
```

```
246 socket.on('changePassword', function(user, newPassword){
247 if(auth(user)){
248 getUserObjectByIDObject(user.id).pass = newPassword;
249 }
250 });
251
252 socket.on('', function(user){
253 });
254 });
255
256 /**
257 returns the user object that has the same username and password
258 */
259 function getUserObjectByUserPass(username, password){
260 for(var i =0; i<users.length; i++){
261 if(users[i].name == username && users[i].pass == password){
262 return users[i];
263 }
264 }
265 }
266
267
268 /**
269 gets the user object from the db
270 param: userId
271 */
272 function getUserObjectByIDObject(user){
273 for(var i =0; i<users.length; i++){
274 if(users[i].id == user.id){
275 return users[i];
276 }
277 }
278 }
279
280 /**
281 determines if the user is an authenticated
282 return: Boolean
283 */
284 function auth(user){
285 if(getUserObjectByIDObject(user).password === user.password){
286 return true;
```

```
287 }
288 return false;
289 }
290
291 /**
292  determines if the user is an administrator
293  return: Boolean
294  */
295 function isAdmin(user){
296  return getUserObjectByIDObject(user).isAdmin;
297 }
298
299 // returns users stripped on passwords
300 function getStrippedUsers(){
301  // SECURITY: strip users of passwords before sending to client
302  var clientList = [];
303  users.forEach(function(user) {
304  clientList.push({nick: user.nick, status: user.status});
305  });
306  return clientList;
307 }
308
309 function saveFile(name, file){
310  fs.writeFile('html/upload/' + name, file, function (err,data) {
311  if (err) {
312      console.log(err);
313  }
314  });
315 }
```