```
var app = require('express')();
var http = require('http').Server(app);
var io = require('socket.io')(http);
var fs = require('fs');
var port = process.env.PORT || 80;

app.get(/^(.+)$/, function(req, res){
res.sendFile(__dirname + '/html' + req.params[0]);
});

http.listen(port, function(){
console.log('listening on *:' + port);
});

var groups = new Array(); // Array of Group

var nextGroupID = 1; // id to serve to next new group

/**
User Object
id-Integer: The id of the the user
nick-String: The nickname of the user
pass-String: The hash of the user's password
status-String: status of user, "online" or "offline"
**/

/**
Group Object
id-Integer: id of group
messages-Array[String]: array of messages
name-String: name of group
**/

/**
List of Users
key: id-Integer
**/
var users = [];
var nextUserID = 1; // id to serve to next new user

// Load the database into groups, and users
```

```
function loadDB(){
// load the groups users nextGroupID, and nextUserID from the db

// Database mock
// remove when database code is working
/**
users.push({
id: 0,
name: "Bismarck",
nick: "Bismarck",
isAdmin: true,
pass: "password",
status: "online"
});
**/

/**
groups.push({
id: 0,
name: "General",
messages: []
});
**/

fs.readFile('server/users.dat', 'utf8', function (err,data) {
  if (err) {
    return console.log(err);
  }
  var strList = data.split(/\r?\n/);
  for(var i =0; i<strList.length;){
  var u = {};
  u.id = strList[i++];
  u.name = strList[i++];
  u.nick = strList[i++];
  u.isAdmin = strList[i++];
  u.pass = strList[i++];
  u.status = "offline";
  users.push(u);
  }
  nextUserID = users[users.length -1].id+1;
});
```

```


fs.readFile('server/numGroup.dat', 'utf8', function (err,data) {
  if (err) {
    return console.log(err);
  }

  var numGroup = data;

  for(var i = 0; i<numGroup; i++) {
  data = fs.readFileSync('server/group'+i+'.dat', 'utf8');
var strList = data.split(/\r?\n/);
var group = {};
group.id = i;
group.messages = [];
group.name =strList[0];
for(var n = 1; n<strList.length; n++){
group.messages.push(strList[n]);
}
groups.push(group);
  }
  nextGroupID = groups.length;
});
}

loadDB();


// User connection
io.on('connection', function(socket){
console.log('User Connected');

socket.on('disconnect', function(){
console.log('User Disconnected');
});

/**
send message to groups
**/
socket.on('groupMessage', function(user, groupID, message){
if(auth(user)){
```

```
for (var i = 0; i<groups.length; i++){
if(groupID==groups[i].id){
groups[i].messages.push(message);
// db.insertData();
}
}
fs.appendFileSync('server/group'+groupID+'.dat', '\n'+message);
io.emit('groupMessage',user, groupID, message);
}
});


/**
creates a new group
**/
socket.on('newGroup', function(user, groupName){
var newGroup = {};
newGroup.id = nextGroupID++;
newGroup.name = groupName;
newGroup.messages = new Array();
// write to DB
groups.push(newGroup);
console.log(groupName);
fs.writeFileSync("server/group"+newGroup.id+".dat", groupName);
console.log(nextGroupID);
fs.writeFileSync("server/numGroup.dat", nextGroupID);

io.emit('newGroup', groups);
});

/**
creates a new user
**/
socket.on('newUser', function(name, pass, isAdmin, callback){
var newUser = {};
newUser.nick = name;
newUser.name = name;
newUser.isAdmin = isAdmin;
newUser.pass = pass;
newUser.status = "online";
//newUser.id =nextUserID++;
```

```
newUser.id = nextUserID++;
users.push(newUser);
addUsertoDB(newUser);

callback(newUser);
io.emit('updateUserList', getStrippedUsers());
});

function addUsertoDB(newUser){
// write to DB
fs.appendFileSync('server/users.dat', '\n'+newUser.id+'\n');
fs.appendFileSync('server/users.dat', newUser.name+'\n');
fs.appendFileSync('server/users.dat', newUser.nick+'\n');
fs.appendFileSync('server/users.dat', newUser.isAdmin+'\n');
fs.appendFileSync('server/users.dat', newUser.pass);
}

/**
gets groups and messages under it
**/
socket.on('getGroups', function(callback){
callback(groups);
});


/**
gets list of users that are online
**/
socket.on('getUsers', function(callback){
callback(getStrippedUsers());
});

/**
sets the status of the user
**/
socket.on('setStatus', function(user, status){
if(auth(user)){
getUserObjectByIDObject(user).status = "online";
}
io.emit('updateUserList', getStrippedUsers());
});
```

```

/**
sets the nickname of the user
**/
socket.on('setNickname', function(user, nick){
if(auth(user)){
getUserObjectByIDObject(user).nick = nick;
}
io.emit('updateUserList', getStrippedUsers());
});

socket.on('uploadFile', function(name, file, callback){
console.log(name);
saveFile(name, file);
callback();
});

/**
Returns the code of the landing page
**/
socket.on('getLandingCode', function(callback){
fs.readFile('server/landing.html', 'utf8', function (err,data) {
if (err) {
    console.log(err);
  }
  callback(data);
});
});

/**
Grabs the user object and returns it.  Null if doesn't exist
**/
socket.on('grabUserObjectByUserPass', function(username, password,
callback){
var u = getUserObjectByUserPass(username,password);
if(u!=null){

}
callback(u);
});

```

```
socket.on('changePassword', function(user, newPassword){
if(auth(user)){
getUserObjectByIDObject(user.id).pass = newPassword;
}
});

socket.on('', function(user){
});
});

/**
returns the user object that has the same username and password
**/
function getUserObjectByUserPass(username, password){
for(var i =0; i<users.length; i++){
if(users[i].name == username && users[i].pass == password){
return users[i];
}
}
}


/**
gets the user object from the db
param: userId
**/
function getUserObjectByIDObject(user){
for(var i =0; i<users.length; i++){
if(users[i].id == user.id){
return users[i];
}
}
}

/**
determines if the user is an authenticated
return: Boolean
**/
function auth(user){
if(getUserObjectByIDObject(user).password === user.password){
return true;
```

```
}
return false;
}

/**
determines if the user is an administrator
return: Boolean
**/
function isAdmin(user){
return getUserObjectByIDObject(user).isAdmin;
}

// returns users stripped on passwords
function getStrippedUsers(){
// SECURITY: strip users of passwords before sending to client
var clientList = [];
users.forEach(function(user) {
clientList.push({nick: user.nick, status: user.status});
});
return clientList;
}

function saveFile(name, file){
fs.writeFile('html/upload/' + name, file, function (err,data) {
if (err) {
    console.log(err);
}
});
}
```