```
#!/usr/local/bin/node

//The code is build ontop of node's generic HTTP stack. No express or similar
libraries

var http = require("http")
var fs = require("fs")
var querystring = require('querystring');

//sg is our symbolic grader module
var sg = require("./symGrader");

//Users represents the interface with a Mongo no-sql database
var users = require("./users");
var args = process.argv

//Command-Line Options
if (args[2] && ( args[2] == "-h" || args[2] == "--help") ) {
        console.log("Usage servjs [port] [index file]")
        return
}

//Aliases for webpages. I.E. instead of http://localhost/views/login.html, you can do
http://localhost/login
var pages = {
        login:"views/login.html",
        register:"views/register.html",
        problem:"views/assignment.html",
        assignment:"views/assignments.html",

}

//URL's that require a login to access. This is just for user-experience, the actual
security is done when the user goes to access something
var rest = [
        "views/assignment.html",
        "courses"
]

//This function creates the server and handles data from req[uests] and pipes them
into the res[ponse].
http.createServer(function(req,res) {
        var cookies = parseCookie(req.headers.cookie);

        //Handles all GET requests
        if (req.method == "GET") {

                //Chooses the URL. First priority is the url they request, second
priority is the url passed as a command line argument, and the third priority is the
login page
                var url = req.url.substr(1) || args[3] || "views/login.html"

                //Ignores URL options - Those are for the client
                url = url.split("?")[0];
                if (pages[url]) {
                        url = pages[url]
                }

                //Finds the mime type of the page being sought
                var cT = getMime(url.split(".")[1])
```

```
		//Case for a restricted URL
		if (rest.indexOf(url) >= 0) {

			//Checks the validity of the authentication, based on a
username and an authentication token issued on login or register
			users.isValid(cookies.username, cookies.auth, function(valid)
{
				console.log("Request received for restricted page:
"+url+" with a "+valid+" validity");
				if (valid) {

					//This is a special case of a GET request,
because it is being accessed from an XMLHttpRequest, and needs to send data, not a
file.
					res.writeHead(200, {'Content-Type': cT});
					if (url == "courses") {

						//Gets the data, again checks the
authentication of the user
						users.getUserData(cookies.username,
cookies.auth, function(err,data) {
							if (data) {

console.log(">>>>>>>>> "+data.courses);

users.getCourses(data.courses, function(dob) {

res.write(JSON.stringify(dob));
res.end();

})
							} else {
res.end();
							}
						})
					} else {

						//In most cases, it can just get the
file
						getFile(url, {}, function(data) {
							res.write(data);
							res.end();
						})
					}
				} else {

					//If the user is not validly logged in,
redirect them to the login page
					res.writeHead(302, "Redirect",
{"Location":"/login"});
					res.end();
				}

			})
		} else {

			//If there are no restrictions on the page, show it to the
user
			console.log("Request received for unrestricted page: "+url);
			res.writeHead(200, {'Content-Type': cT});
```

```
                        getFile(url, {}, function(data) {
                                res.write(data);
                                res.end();
                        })
                }

        //Handles all the POST requests, coming from both XHR's and form POST's
        } else if (req.method == "POST") {
                fullBody = "";
                req.on('data', function(chunk) {
                        //Append the current chunk of data to the fullBody variable
                        fullBody += chunk.toString();
                });

                req.on('end', function() {
                        console.log("Post data received "+fullBody);


                        //If the data is an HTTP query string, decode it
                        var dec = querystring.parse(fullBody);

                        //Handles login posts
                        if (req.url === "/login") {

                                //Authenticates the user using our database module

users.authUser(dec.username,dec.password,"0.1.2.3",function(err, authToken) {

                                        //If they are not authenticated, send them
back to login, alerting them of the error
                                        if (err) {
                                                res.writeHead(200, "OK", {'Content-
Type': 'text/html'});
                                                getFile("views/login.html", {error:
err}, function(data) {
                                                        res.write(data);
                                                        res.end();
                                                })

                                        //Otherwise redirect them to the problem
page, and issue them an authentication token
                                        } else {
                                                var opt = [["Location","/problem"],
["Set-Cookie","auth="+authToken], ["Set-Cookie","username="+dec.username]]
                                                console.log(opt);
                                                res.writeHead(302, "Redirect", opt );
                                                res.end();
                                        }
                                })

                        //Handles all registration requests
                        } else if (req.url === "/register") {

                                //Creates a user using our database module. Behaves
similarly to login

users.createUser(dec.name,dec.email,dec.username,dec.password,"0.1.2.3"
,function(err, authToken) {
                                        console.log("Given authToken: "+authToken);
                                        if (err) {
                                                res.writeHead(200, "OK", {"Content-
```

```
Type": "text/html"});
                                                getFile("views/register.html",
{error: err}, function(data) {
                                                        res.write(data);
                                                        res.end();
                                                })
                                        } else {
                                                var opt = [["Location","/problem"],
["Set-Cookie","auth="+authToken], ["Set-Cookie","username="+dec.username]]
                                                res.writeHead(302, "Redirect",opt)
                                                res.end();
                                        }
                                })

                        //Handles requests for submitting answer. Checks the correct
answer against the submitted one
                        } else if (req.url == "/answers" ) {
                                console.log("Received this junk");
                                users.getCourses([1], function(data) {
                                        var post = JSON.parse(fullBody);
                                        var qs =
(data[0].assignments[0].questions[post.question].content)
                                        var iter = 0;
                                        var ended = false;
                                        for (var i = 0; i < qs.length; i++) {
                                                if (typeof qs[i] == "object") {
                                                        if (iter ==
parseInt(post.part)) {
                                                                if (qs[i].answer) {
                                                                        if
(qs[i].type == "symbolic") {
                                                                                var
ans = sg.test(post.answer, qs[i].answer, qs[i].variables, qs[i].range, 3, [], 0)

res.writeHead(200, "OK", {"Content-Type":"text/html"});

res.write((ans).toString());

res.end();
                                                                                ended
= true;

                                                                        } else {

res.writeHead(200, "OK", {"Content-Type":"text/html"});

res.write((qs[i].answer == post.answer).toString());

res.end();
                                                                                ended
= true;
                                                                        }
                                                                }
                                                                console.log(qs[i]);
                                                                console.log(post);
                                                        }
                                                        iter++;
                                                }
                                        }
                                        if (!ended) {
                                                res.writeHead(200, "OK", {"Content-
```

```
Type":"text/html"});
                                                        res.write("false");
                                                        res.end();
                                                }

                                        })

                                //Sending useless data. Ignore it
                                } else {
                                        res.write("Invalid Post Data");
                                        res.end;
                                }
                        });
        }

//Listen on the port specified, or default to 1337. Accepts connections from all IPs
}).listen(args[2] || 1337, '0.0.0.0');

console.log('Server running');


//Helper method to read files
var getFile = function(url,rep, cb) {
        url = url.split("..")
        url = "../public/"+url[url.length-1];
        fs.readFile(url, function(err, data) {
                if (err || !data) {
                        cb("404")
                        return
                }
                cb(fillIn(data,rep));
        })
}

//Helper method to parse cookie strings
var parseCookie = function(cookieString) {
        if (!cookieString) {
                return {}
        }
        var spl = cookieString.split(";");
        var ret = {}
        for (var i in spl) {
                var sple = spl[i].split("=");
                var key = sple[0];
                var value = spl[i].split(key+"=")[1];
                if (key[0] == " ") {
                        key = key.substr(1)
                }
                ret[key] = value;
        }
        return ret;
}

//Helper method to detirmine mime types
var getMime = function(str) {
        if (!str) {
                return "text/plain";
        }
        str = str.toLowerCase();
        switch (str) {
                case "html":
```

```
                        return "text/html"
                case "txt":
                        return "text/plain"
                case "js":
                case "min":
                case "json":
                        return "application/javascript"
                case "gif":
                        return "image/gif"
                case "jpeg":
                case "jpg":
                        return "image/jpeg"
                        case "png":
                case "bmp":
                case "ico":
                        return "image/png"
                case "css":
                        return "text/css"
                case "xml":
                        return "text/xml"
                default:
                        return "text/html"

        }
}

//Helper method to pass the read files with any useful data
var fillIn = function(file,rep) {
        file = file.toString();
        for (var i in rep) {
                file = file.split("{{"+i+"}}").join(rep[i]);
        }
        return file;

}
```