```
var db = require("mongojs").connect("localhost:27017/integ",
["users","courses","testdb"])
var x = require("./XOR/XOR")
var bc = require("bcrypt")

//Creates a user and grants him an authentication token
var createUser = function(name,email,username,pass,ipaddress,cb) {
        db.users.find({"data.username":username},function(err, dob) {
                if (dob.length != 0) {
                        cb(101,"");
                        return;
                }
                db.users.find({"data.email":email}, function(err, dob) {
                        if (dob.length != 0) {
                                cb(102, "");
                                return;
                        }
                        var authToken =
x.toB64(x.XOR(username+Math.floor(Math.random()*1000000)+x.toB64(username),ipaddress)
);
                        var passHash = bc.hash(pass,4,function(err,hash) {
                                user = {
                                        "data": {
                                                "username": username,
                                                "password": hash,
                                                "email": email,
                                                "name":name
                                        },
                                        "courses":[1,2],
                                        "private": {
                                                "authToken":[authToken]
                                        }
                                }
                                console.log(user);
                                db.users.save(user);
                                cb(null,authToken);
                                console.log("Specified authToken "+authToken);
                        })
                })
        })
}

//Bogus method for creating an example course. Used for debugging
var genTestCourse = function() {
        var a = {
                title: "Feesiks",
                UID : 1,
                assignments: [{
                        title: "Various Practice Problems",
                        UID : 1,
                        questions: [
                        {
                                title: "Sally, Bobby, and Apples",
                                content: [
                                "Sally and Bobby are going on a picnic, as you can
see in the diagram below.",
                                "<img
src='http://latinasypunto.files.wordpress.com/2010/09/picnic.jpg' height=250 />",
                                "Bobby has 4 apples and Sally has 3 apples.",
                                {
```

```
                                        text: "How many apples do they have put
together?  (Check all that apply.)",
                                        type: "check",
                                        options: [
                                        "7",
                                        "6",
                                        "Kyle",
                                        "-2",
                                        "What's an apple?"
                                        ],
                                        answer: "0",
                                },
                                "Now suppose that Bobby gives Sally 16 of his
apples.",
                                {
                                        text: "How many apples does Bobby have now?",
                                        type: "select",
                                        options: [
                                        "16 kg",
                                        "wut?",
                                        "African or European?"
                                        ],
                                        answer: "1",
                                },
                                {
                                        text: "How many apples does Sally have now?",
                                        type: "input",
                                        answer: "19",
                                },
                                {
                                        text: "What is the mass of the moon with
respect to teslas?  (Use the following variables as necessary: \\(a, x_0, theta,
\\vec{r}\\).)",
                                        type: "symbolic",
                                        variables: ["a", "x_0", "theta", "apple",
"rvector", "ihat"],
                                        range: [[1,2],[1,2],[1,2],[1,2],[1,2],[1,2]],
                                        steps: 3,
                                        answer: "a*x_0/apple(theta)",

                                },
                                {
                                        text: "This is the last question.",
                                        type: "check",
                                        options: [
                                        "yes"
                                        ],
                                        answer: "0",
                                }
                                ]
                        },
                        {
                                title: "Solenoids (Tipler 6 28.P.079)",
                                content: [
                                "A long solenoid has \\(n\\) turns per unit length
and carries a current that varies with time according to \\(I = I_0 sin \\omega t\\).
The solenoid has a circular cross section of radius \\(R\\). Find the induced
electric field, at points near the plane equidistant from the ends of the solenoid,
as a function of both the time \\(t\\) and the perpendicular distance \\(r\\) from
the axis of the solenoid for the following. (Use the following as necessary: \\(n, r,
R, t, \\mu_0, I_0, \\omega\\).)",
```

```
                                {
                                        text: "\\(r < R\\)",
                                        type: "symbolic",
                                        variables: ["n", "r", "R", "t", "mu_0",
"I_0", "omega"],
                                        range: [[1,2], [1,2], [1,2], [1,2], [1,2],
[1,2], [1,2]],
                                        steps:3,
                                        answer: "mu_0*n*I_0/r"
                                },
                                {
                                        text: "\\(r > R\\)",
                                        type: "symbolic",
                                        variables: ["n", "r", "R", "t", "mu_0",
"I_0", "omega"],
                                        range: [[1,2], [1,2], [1,2], [1,2], [1,2],
[1,2], [1,2]],
                                        steps:3,
                                        answer: "mu_0*n*I_0/R"
                                }
                                ]
                        }
                        ]
                }]
        }
        db.courses.save(a);
}

//Makes sure it doesn't add the test course multiple times
db.courses.find({"UID":1},function(err, dob) {
        if (dob.length == 0) {
                genTestCourse();
        }
})

//Checks to see if a username/authToken pair is valid
var isValid = function(username, authToken, cb) {
        db.users.find({"data.username":username, "private.authToken": authToken},
function(err, dob) {
                if (err || dob.length == 0) {
                        cb(false)
                        return
                }
                cb(true)
        })
}

//gets all courses associated with a course UID
var getCourses = function(uids,cb) {
        db.courses.find({"UID":{$in:uids}},function(err, dob) {
                console.log(dob)
                cb(dob);
        })
}

//Revokes an authentication token
var endSession = function(username, authToken) {
        db.users.update({"data.username":username}, {"$pull":
{"private.authToken":authToken}})
}
```

```
//Updates user data
var updateUser = function(username, authToken, data, cb) {
        var obj = ["username","email","name"];
        for (i in data) {
                if (obj.indexOf(i) < 0) {
                        cb(109);
                        return;
                }
                var tmpobj = {};
                tmpobj["data."+i] = data[i];
                db.users.update({"data.username":username,
"private.authToken":authToken}, {$set:tmpobj})
        }
        if (cb) {
                cb(null, authToken);
        }
}

//Returns user data
var getUserData = function(username,authToken,cb) {
        db.users.find({"data.username":username,"private.authToken":authToken},functi
on(err, dob) {
                if (dob.length == 0) {
                        cb(202,"");
                        return
                }
                user = {
                        data: {
                                name: dob[0].data.name,
                                username: dob[0].data.username,
                                email: dob[0].data.email,
                        },
                        courses: dob[0].courses,
                }

                if (cb) {
                        cb(null,user);
                }
                //console.log(user)

                //db.users.remove({"private.authToken":{$in:[authToken]}})
        })
}

//Checks a username with the password hash and if it is valid, grants an
authentication token
var authUser = function(username,pass,ipaddress, cb) {
        console.log ("Looking for "+username);
        db.users.find({"data.username":username},function(err,dob) {
                if (dob.length == 0) {
                        cb(201);
                        return
                }
                bc.compare(pass, dob[0].data.password ,function(err, res) {
                        if (err) {
                                console.error(err);
                        }
                        if (res) {
                                var authToken =
x.toB64(x.XOR(username+Math.floor(Math.random()*1000000)+x.toB64(username),ipaddress)
);
```

```
                                db.users.update({"_id":dob[0]._id}, {"$push":
{"private.authToken":authToken}},function() {
                                        if (cb) {
                                                cb(null,authToken);
                                        }
                                })
                        } else {
                                cb(202,"");
                        }
                })
        })
}

//Specifies the functions to be exported as a node module
module.exports = {
        "getUserData": getUserData,
        "authUser": authUser,
        "updateUser": updateUser,
        "endSession": endSession,
        "createUser": createUser,
        "isValid": isValid,
        "getCourses": getCourses,

}
```