Exercise (Instructions): User Authentication with Passport

Objectives and Outcomes

In this exercise, you will explore user authentication with JSON web tokens and the Passport module. You will be able to control access to some routes within your REST server. At the end of this exercise, you will be able to:

- Use JSON web tokens for token-based user authentication
- Use Passport module together with passport-local and passport-local-mongoose for setting up local authentication within your server.

Setting up the Project

- Copy the *rest-server* folder that you created in the first exercise in this module and rename the folder copy as*rest-server-passport*. You will modify this project to set up user authentication support using tokens and Passport.

Installing Passport and jsonwebtoken Node Modules

- Install the Passport related Node modules and the jsonwebtoken module as follows:

```
    npm install passport passport-local passport-local-mongoose --save
    npm install jsonwebtoken --save
```

Set up a config File

- Create a new file named config.js and add the following code to it:

```
module.exports = {
    'secretKey': '12345-67890-09876-54321',
    'mongoUrl' : 'mongodb://localhost:27017/conFusion'
}
```

Updating app.js

- Update app.js as follows:

```
var express = require('express');
```

```
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var mongoose = require('mongoose');
var passport = require('passport');
var LocalStrategy = require('passport-local').Strategy;

var config = require('./config');

mongoose.connect(config.mongoUrl);
var db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function () {
    // we're connected!
    console.log("Connected correctly to server");
});

var routes = require('./routes/index');
var users = require('./routes/users');
var dishRouter = require('./routes/dishRouter');
var promoRouter = require('./routes/promoRouter');
var leaderRouter = require('./routes/leaderRouter');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');
// uncomment after placing your favicon in /public
//app.use(favicon(path.join(__dirname, 'public', 'favicon.ico')));

app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
```

```
// passport config
var User = require('./models/user');
app.use(passport.initialize());
passport.use(new LocalStrategy(User.authenticate()));
passport.serializeUser(User.serializeUser());
passport.deserializeUser(User.deserializeUser());

app.use(express.static(path.join(__dirname, 'public')));

app.use('/', routes);
app.use('/users', users);
app.use('/dishes',dishRouter);
app.use('/promotions',promoRouter);
app.use('/leadership',leaderRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handlers
// development error handler
// will print stacktrace
if (app.get('env') === 'development') {
  app.use(function(err, req, res, next) {
    res.status(err.status || 500);
    res.json({
      message: err.message,
      error: err
    });
  });
}

// production error handler
```

```
// no stacktraces leaked to user
app.use(function(err, req, res, next) {
  res.status(err.status || 500);
  res.json({
    message: err.message,
    error: {}
  });
});

module.exports = app;
```

Setting up User Schema and Model

- In the models folder, create a file named user.js and add the following code to it:

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var passportLocalMongoose = require('passport-local-mongoose');

var User = new Schema({
    username: String,
    password: String,
    admin:   {
        type: Boolean,
        default: false
    }
});

User.plugin(passportLocalMongoose);

module.exports = mongoose.model('User', User);
```

Updating /users route in users.js

- Open *user.js* file in the *routes* folder and update the code as follows:

```
var express = require('express');
```

```
var router = express.Router();
var passport = require('passport');
var User = require('../models/user');
var Verify    = require('./verify');

/* GET users listing. */
router.get('/', function(req, res, next) {
  res.send('respond with a resource');
});

router.post('/register', function(req, res) {
    User.register(new User({ username : req.body.username }),
      req.body.password, function(err, user) {
        if (err) {
            return res.status(500).json({err: err});
        }
        passport.authenticate('local')(req, res, function () {
            return res.status(200).json({status: 'Registration Successful!'});
        });
    });
});

router.post('/login', function(req, res, next) {
  passport.authenticate('local', function(err, user, info) {
    if (err) {
      return next(err);
    }
    if (!user) {
      return res.status(401).json({
        err: info
      });
    }
    req.logIn(user, function(err) {
      if (err) {
        return res.status(500).json({
          err: 'Could not log in user'
        });
```

```
      }

      var token = Verify.getToken(user);
              res.status(200).json({
        status: 'Login successful!',
        success: true,
        token: token
      });
    });
  })(req,res,next);
});

router.get('/logout', function(req, res) {
    req.logout();
  res.status(200).json({
    status: 'Bye!'
  });
});

module.exports = router;
```

Supporting JSON Web Tokens and Verification

- In the *routes* folder create a file named *verify.js* and add the following code to it:

```
var User = require('../models/user');
var jwt = require('jsonwebtoken'); // used to create, sign, and verify tokens
var config = require('../config.js');

exports.getToken = function (user) {
    return jwt.sign(user, config.secretKey, {
        expiresIn: 3600
    });
};

exports.verifyOrdinaryUser = function (req, res, next) {
    // check header or url parameters or post parameters for token
```

```
    var token = req.body.token || req.query.token || req.headers['x-access-token'];

    // decode token
    if (token) {
        // verifies secret and checks exp
        jwt.verify(token, config.secretKey, function (err, decoded) {
            if (err) {
                var err = new Error('You are not authenticated!');
                err.status = 401;
                return next(err);
            } else {
                // if everything is good, save to request for use in other routes
                req.decoded = decoded;
                next();
            }
        });
    } else {
        // if there is no token
        // return an error
        var err = new Error('No token provided!');
        err.status = 403;
        return next(err);
    }
};
```

Controlling Routes with Authentication

- Open *dishRouter.js* and updated the code for the '/' route as follows:

```
dishRouter.route('/')

.get(Verify.verifyOrdinaryUser, function (req, res, next) {

   . . .

})
```

```
.post(Verify.verifyOrdinaryUser, function (req, res, next) {

   . . .

})

.delete(Verify.verifyOrdinaryUser, function (req, res, next) {

   . . .

});
```

- Save the changes and test the server by sending various requests.

Conclusions

In this exercise you used token-based verification together with the Passport module to verify the authenticity of users and control access to routes in a REST API server.