### **User Authentication with Passport**

#### Objectives and Outcomes

In this exercise, you will explore user authentication with the Passport module. You will be able to control access to your REST API server. At the end of this exercise, you will be able to:

* Use Passport module together with passport-local and passport-local-mongoose for setting up local authentication within your server.
* Use Passport together with sessions to set up user authentication.

#### Installing Passport

* You will continue working with the Express REST API Server in the conFusionServer folder. You will modify this project to set up user authentication support using Passport.
* Install the Passport and related Node modules as follows:

npm install passport passport-local passport-local-mongoose

#### Updating User Schema and Model

* In the models folder, update user.js by adding the following code to it:
* . . .
* var passportLocalMongoose = require('passport-local-mongoose');
* var User = new Schema({
* admin: {
* type: Boolean,
* default: false
* }
* });
* User.plugin(passportLocalMongoose);
* . . .

#### Adding Passport-based Authentication

* Add a new file named authenticate.js to the project folder and initialize it as follows:
* var passport = require('passport');
* var LocalStrategy = require('passport-local').Strategy;
* var User = require('./models/user');
* passport.use(new LocalStrategy(User.authenticate()));
* passport.serializeUser(User.serializeUser());
* passport.deserializeUser(User.deserializeUser());
* Open users.js file in the routes folder and update the code as follows:
* . . .
* var passport = require('passport');
* . . .
* router.post('/signup', (req, res, next) => {
* User.register(new User({username: req.body.username}),
* req.body.password, (err, user) => {
* if(err) {
* res.statusCode = 500;
* res.setHeader('Content-Type', 'application/json');
* res.json({err: err});
* }
* else {
* passport.authenticate('local')(req, res, () => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json({success: true, status: 'Registration Successful!'});
* });
* }
* });
* });
* router.post('/login', passport.authenticate('local'), (req, res) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json({success: true, status: 'You are successfully logged in!'});
* });
* . . .
* Finally, update app.js as follows:
* . . .
* var passport = require('passport');
* var authenticate = require('./authenticate');
* . . .
* app.use(passport.initialize());
* app.use(passport.session());
* . . .
* function auth (req, res, next) {
* console.log(req.user);
* if (!req.user) {
* var err = new Error('You are not authenticated!');
* err.status = 403;
* next(err);
* }
* else {
* next();
* }
* }
* . . .
* Save the changes and test the server by sending various requests.
* Do a Git commit with the message "Passport".

#### Conclusions

In this exercise you used the Passport module to verify the authenticity of users and control access to routes in a REST API server.