### **User Authentication with Passport and JSON Web Token**

#### 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.

#### Installing passport-jwt and jsonwebtoken Node Modules

* 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 tokens and Passport.
* Install the passport-jwt and the jsonwebtoken modules as follows:

npm install passport-jwt jsonwebtoken

#### Updating the App to use JSON Web Tokens

* 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'
* }

#### Supporting JSON Web Tokens and Verification

* Update authenticate.js as follows:
* . . .
* var JwtStrategy = require('passport-jwt').Strategy;
* var ExtractJwt = require('passport-jwt').ExtractJwt;
* 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});
* };
* var opts = {};
* opts.jwtFromRequest = ExtractJwt.fromAuthHeaderAsBearerToken();
* opts.secretOrKey = config.secretKey;
* exports.jwtPassport = passport.use(new JwtStrategy(opts,
* (jwt\_payload, done) => {
* console.log("JWT payload: ", jwt\_payload);
* User.findOne({\_id: jwt\_payload.\_id}, (err, user) => {
* if (err) {
* return done(err, false);
* }
* else if (user) {
* return done(null, user);
* }
* else {
* return done(null, false);
* }
* });
* }));
* exports.verifyUser = passport.authenticate('jwt', {session: false});
* Open users.js file in the routes folder and update the code as follows:
* . . .
* var authenticate = require('../authenticate');
* . . .
* router.post('/login', passport.authenticate('local'), (req, res) => {
* var token = authenticate.getToken({\_id: req.user.\_id});
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json({success: true, token: token, status: 'You are successfully logged in!'});
* });
* . . .
* Update app.js to remove the auth function and the app.use(auth), and update as follows:
* . . .
* var config = require('./config');
* . . .
* const url = config.mongoUrl;
* . . .

#### Controlling Routes with Authentication

* Open dishRouter.js and updated the code for the '/' route as follows:
* . . .
* var authenticate = require('../authenticate');
* . . .
* dishRouter.route('/')
* .post(authenticate.verifyUser, (req, res, next) => {
* . . .
* })
* .put(authenticate.verifyUser, (req, res, next) => {
* . . .
* })
* .delete(authenticate.verifyUser, (req, res, next) => {
* . . .
* });
* dishRouter.route('/:dishId')
* .post(authenticate.verifyUser, (req, res, next) => {
* . . .
* })
* .put(authenticate.verifyUser, (req, res, next) => {
* . . .
* })
* .delete(authenticate.verifyUser, (req, res, next) => {
* . . .
* });
* dishRouter.route('/:dishId/comments')
* .post(authenticate.verifyUser, (req, res, next) => {
* . . .
* })
* .put(authenticate.verifyUser, (req, res, next) => {
* . . .
* })
* .delete(authenticate.verifyUser, (req, res, next) => {
* . . .
* });
* dishRouter.route('/:dishId/comments/:commentId')
* .post(authenticate.verifyUser, (req, res, next) => {
* . . .
* })
* .put(authenticate.verifyUser, (req, res, next) => {
* . . .
* })
* .delete(authenticate.verifyUser, (req, res, next) => {
* . . .
* });
* . . .

* Do similar updates to promoRouter.js and leaderRouter.js.
* Save the changes and test the server by sending various requests.
* Do a Git commit with the message "Passport JWT".

#### 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.