HTTP Session Management

What is a Session?

- HTTP is a **stateless** protocol
- So, we need to have some logic to keep track of the previous requests to the server
- Session is a server-side storage of information to persist throughout the user’s interaction to the web application
- A **unique identifier** is stored on the client side (session id)
- This identifier is passed on every request to the server
- This identifier is matched by the server and retrieves the information attached with the id

Sessions in Node.js

- Sessions in Node.js are stored using 2 ways:
    - Session state Providers:
        - Cookie + backend store
    - Default sessions:
        - client-sessions or express-sessions module

Sessions in Node.js

- Client-sessions npm module provides simple implementation   npm install client-sessions
- These sessions are limited to application scope
- So when the application is restarted, these sessions are invalidated

```
//Include default session
var session = require(‘client-sessions');
```

Sessions in Node.js

```
var session = require('client-sessions');

app.use(session({
        cookieName: 'session',
        secret: 'cmpe273_test_string',
        duration: 30 * 60 * 1000,
        activeDuration: 5 * 60 * 1000,
      }));
```

Sessions in Node.js

```
var session = require('client-sessions');

app.use(session({
        cookieName: 'session’,          //cookie-name stored on browser

        secret: 'cmpe273_test_string', //secret_id stored
        duration: 30 * 60 * 1000,  //how long the session will stay valid in ms

        activeDuration: 5 * 60 * 1000,  //if expiresIn < activeDuration, the
                                        session will  be extended by
                                        activeDuration milliseconds
        }));
```

Sessions

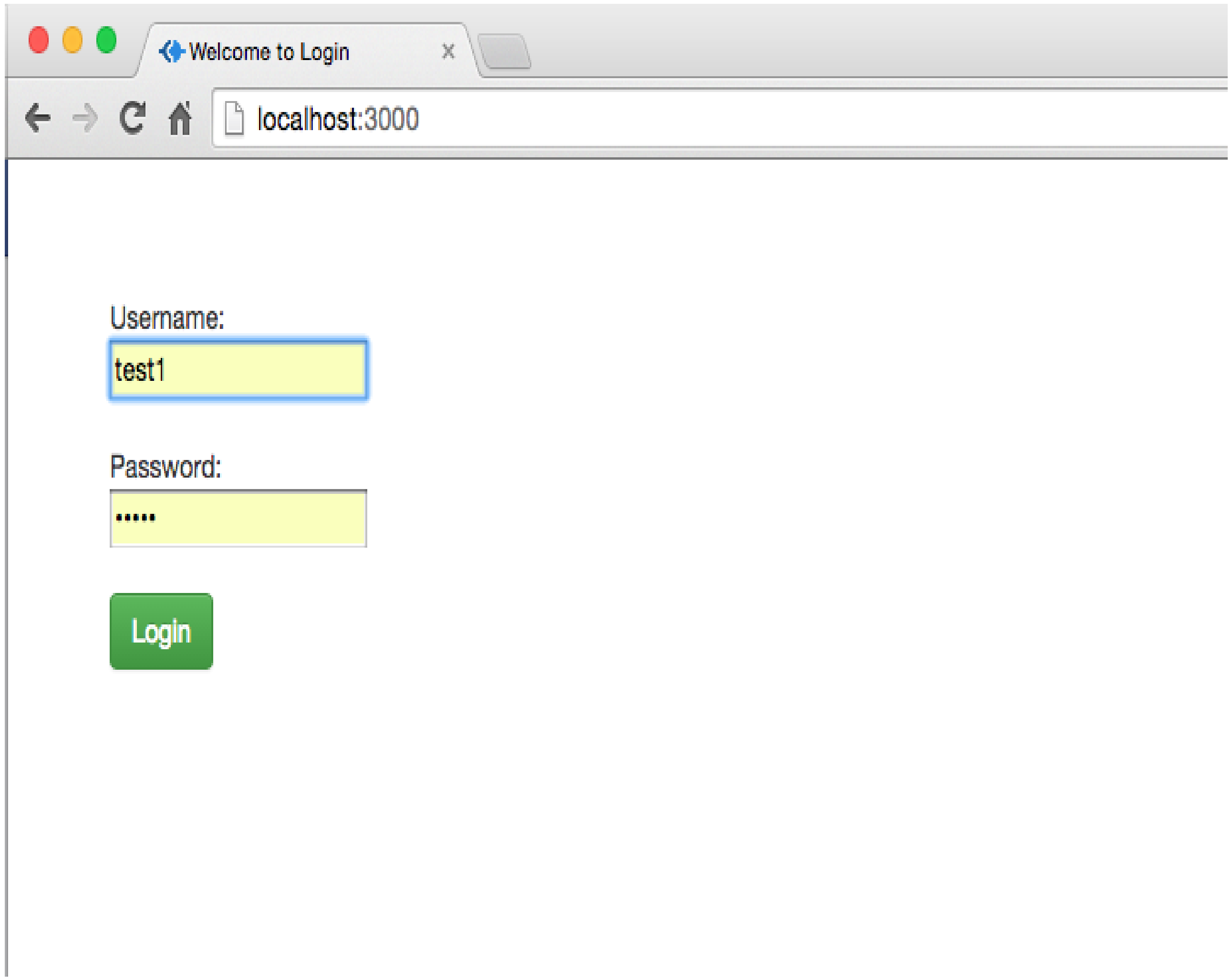
- We can use the create a session using request object

```
//store the username and email
address after successful login
req.session.username = username;
req.session.email_address = email_address;
```

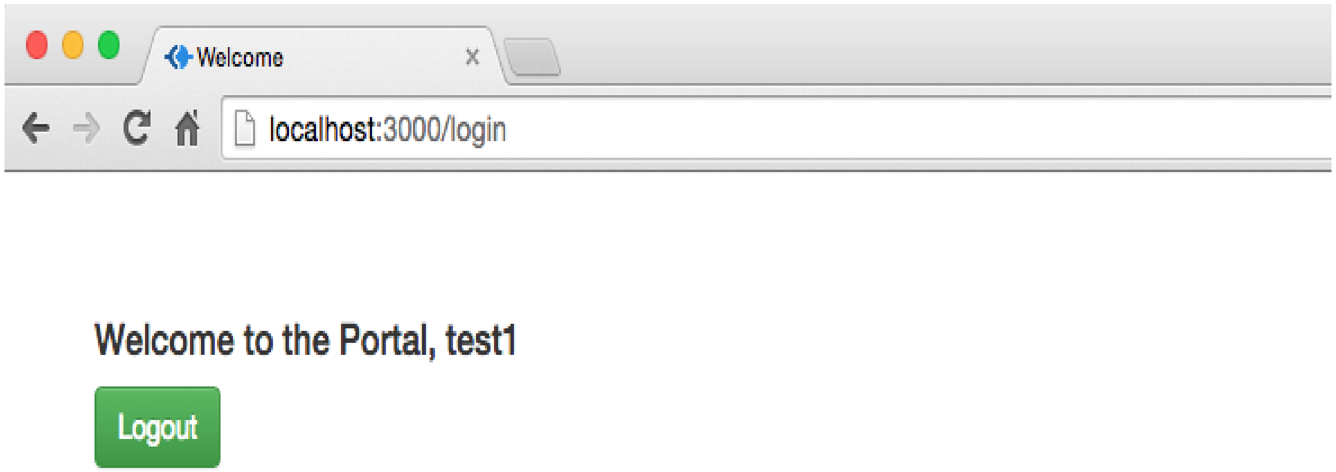
Example

- Login to the application
- Check the session and save the data in session
- Logout after showing information
- After logout, when you press back button, it should not load the home page after login

Example – User Interface



Example – User Interface



Welcome to the Portal, test1

Example – login.js (server side)

```
exports.checklogin = function(req,res)
{
        var username = req.param("username");      //get the username param
        var password = req.param("password");      //get the password param
        var json_response;

        if(username!== ''  && password!== '')             //validate
        {
                if(username === "test1" && password ==="test1")
                {
                        req.session.username = username;
                        json_response = { "statusCode" : 200};
                        res.send(JSON.stringify(json_response));
                }
                else
                {
                        json_response = { "statusCode" : 302};
                        res.send(json_response);
                }
        }
        else
        {
                res.render("index");
        }
};
```

Example – login.js

```
exports.login = function(req,res)  //redirect function to the homepage
{
         if(req.session.username)  //check whether session is valid
         {
                  res.header('Cache-Control', 'no-cache, private, no-store,
must-revalidate, max-stale=0, post-check=0, pre-check=0');
         //disable browser cache
                  res.render("success",{username:req.session.username});
         }
         else
         {
                  res.render("index", { title: 'Welcome to ogin' });
         }
};


exports.logout = function(req,res)                 //logout function
{
         req.session.destroy();                    //destroy session
         res.render("index", { title: 'Welcome to Login' });
};
```

Example – app.js (Session declaration)

```
var session = require('client-sessions');//Require the client-
sessions module

app.use(session({                         //configure the sessions
         cookieName: 'session',
         secret: 'cmpe273_test_string',
         duration: 30 * 60 * 1000,
         activeDuration: 5 * 60 * 1000,
       }));
```

Exercise

- Create a simple shopping cart application
- View items and their cost, and a select item button. Show a cart on the right side, which will be empty at the start
- Add “Add to Cart” button, which sends data to node.js and calculates the total, adds the items to sessions and shows the cart items in a section on the right side of the same page
- When you open the same page in different tab, the cart should be visible with the items in the session

References

- client-sessions documentation:

  https://github.com/mozilla/node-client-sessions

- Default sessions Node.js (client-sessions):

  https://stormpath.com/blog/everything-you-ever-wanted-to-know-about-node-dot-js-sessions/

- External sessions – Redis/MongoDB:

  http://blog.modulus.io/nodejs-and-express-sessions

- MySQL sessions in Node.js:

  https://www.npmjs.com/package/express-mysql-session

Passportjs

- Passportjs is capable of performing authentication and storing sessions
- Stores the sessions on external store – in MySQL database
- Every time a request is sent, after the session is created, we can check whether the session exists
- If session doesn’t exist, redirect user to the login page
- Independent of server which receives the request, as the session is stored on the database

Passportjs - Configuration

```
app.use(session({
                secret: 'cmpe273_testing',
                resave: true,// forces the session to store every time, even when no session data has
been                                                            modified with the request
                saveUninitialized: true,//Forces a new session to be saved to the memory
                duration: 30 * 60 * 1000,//how long the session will stay valid in ms
                activeDuration: 5 * 60 * 1000 // if expiresIn < activeDuration, the session will be
extended                                                                    by activeDuration milliseconds
} )); // session secret
app.use(passport.initialize());
app.use(passport.session()); //persistent login sessions
```

Passportjs - Configuration

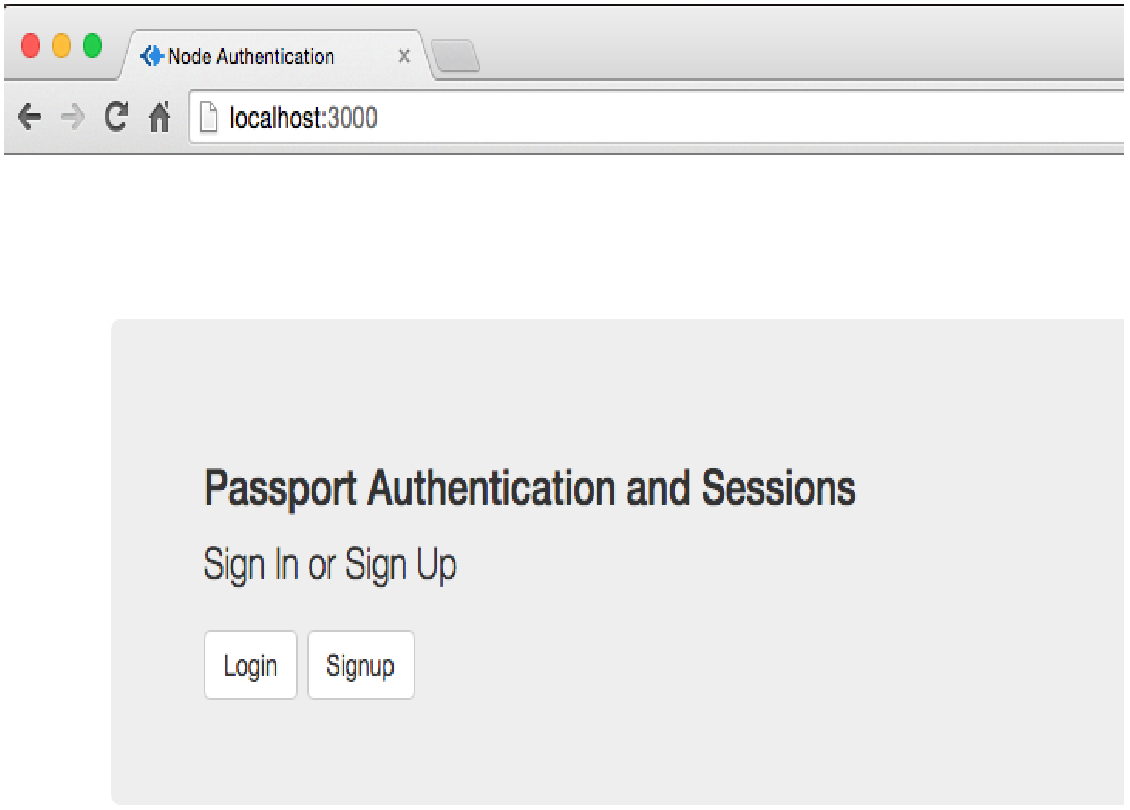
- Passport authentication is done using passport.authenticate function
- 3 results mapped to the function
  - successRedirect – goes to this page if the authentication succeeds
  - failureRedirect – goes to this page if the authentication fails
  - failureFlash – allows messages to be displayed if failure occurs

```
//process the signup form
app.post('/signup', passport.authenticate(’signup', {
		successRedirect : '/profile', // redirect to the secure profile section
		failureRedirect : '/signup', // redirect back to the signup page if there is an error
		failureFlash : true // allow flash messages
}));
```

Example

- Login and Sign up application
- Use Passport module to authenticate and store sessions on MySQL

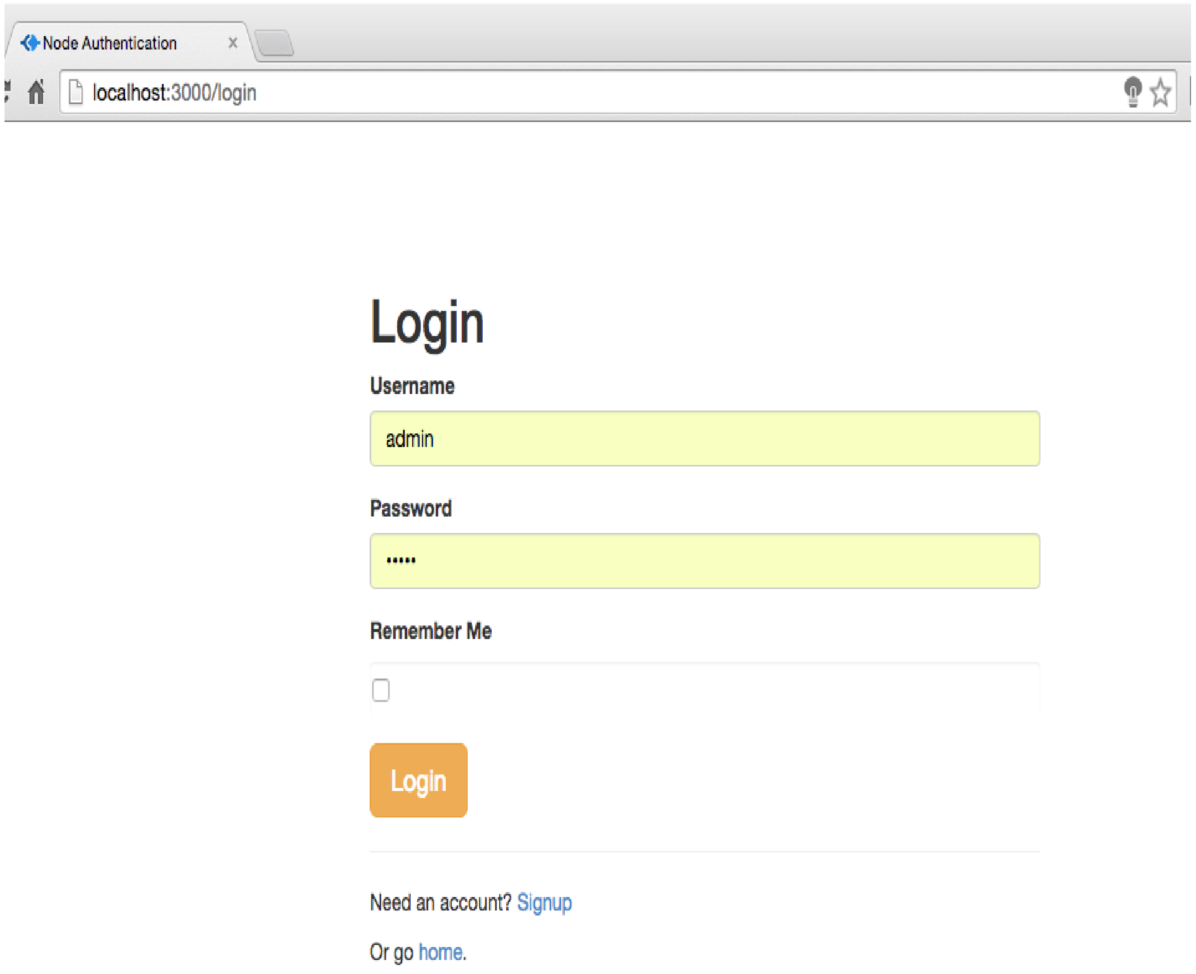
Example – User Interface



Passport Authentication and Sessions

Sign In or Sign Up

Example – User Interface



Node Authentication

localhost:3000/login

Login

Username

admin

Password

•••••

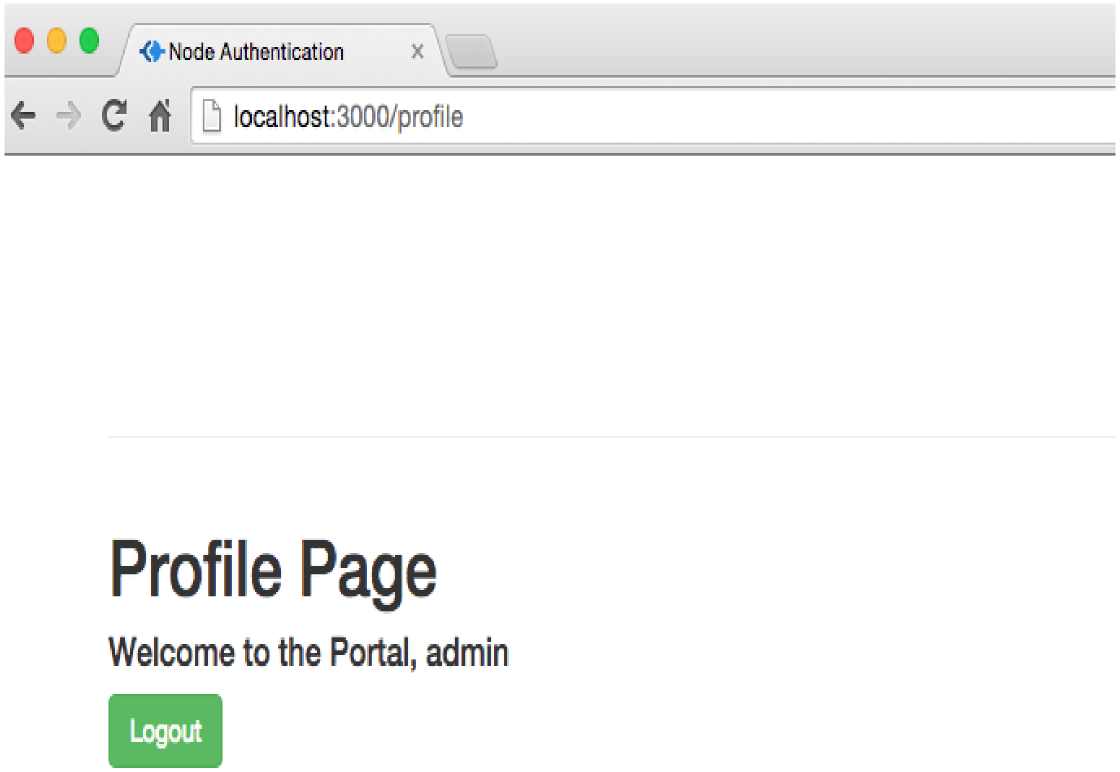
Remember Me

Login

Need an account? Signup

Or go home.

Example – User Interface



Profile Page

Welcome to the Portal, admin

Example – index.ejs

```
<!doctype html>
<html>
<head>
		<title>Node Authentication</title>
		<link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0.2/css/bootstrap.min.css”>
</head>
<body>
    <div class="container">
        <div>
            <h3>Passport Authentication and Sessions</h3>

            <p>Sign In or Sign Up</p>

            <a href="/login" class="btn btn-default">Login</a>
            <a href="/signup" class="btn btn-default">Signup</a>
        </div>
    </div>
</body>
</html>
```

Example – login.ejs

```
<title>Node Authentication</title>
<link rel="stylesheet"
            href="//netdna.bootstrapcdn.com/bootstrap/3.0.2/css/bootstrap.min.css”>
</head>
<body>
<div class="container">
<div class="col-sm-6">
      <h1><span class=""></span> Login</h1>
      <form action="/login" method="post">
            <div class="form-group”><label>Username</label> <input type="text" class="form-control”
      name="username">
            </div>
            <div class="form-group”><label>Password</label> <input type="password" class="form-control”
      name="password">
            </div>
            <div class="form-group"><label>Remember Me</label> <input type="checkbox” class="form-control"
      name="remember" value="yes">
            </div>
            <button type="submit" class="btn btn-success">Login</button>
      </form>
</div>
</div>
```

Example – profile.ejs

```
<body>
        <div class="container">

                <div class="page-header text-center"></div>

                <div class="row">
                        <div class="col-md-12">
                                <h1>Profile Page</h1>

                        </div>
                        <div class="col-md-12">
                                <form action="logout" method="post">
                                        <h4>Welcome to the Portal, <%=
user.username %></h4>
                                        <a href="/logout" class="btn btn-
success">Logout</a>
                                </form>
                        </div>
                </div>


        </div>
</body>
```

Example – Database

Database.js

```
// config/database.js
module.exports = {
    'connection': {
        'host': 'localhost',
        'user': 'root',
        'password': ''
    },
            'database': 'sessions',
    'users_table': 'users'
};
```

Database Schema

*Database needs to be created before running the application

- Database : **sessions**
- Table name : **users**
- Columns:
  - **id** – primary key int not null auto_increment
  - **username** – varchar(20) not null
  - **password** – varchar(30) not null

Example – passport.js (Part 1)

```
// load all the things we need
var LocalStrategy   = require('passport-
local').Strategy;

// load up the user model
var mysql = require('mysql');
var bcrypt = require('bcrypt-nodejs');
var dbconfig = require('./database');
var connection =
mysql.createConnection(dbconfig.connection);

connection.query('USE ' + dbconfig.database);
```

Example – passport.js (Part 2)

```
module.exports = function(passport) {

    // =========================================================================
    // passport session setup ==================================================
    // =========================================================================
    // required for persistent login sessions
    // passport needs ability to serialize and deserialize users out of session

    // used to serialize the user for the session, checking the session is live
    passport.serializeUser(function(user, done) {
        done(null, user.id);
    });

    // used to deserialize the user and destory the session
    passport.deserializeUser(function(id, done) {
        connection.query("SELECT * FROM users WHERE id = ? ",[id], function(err, rows){
            done(err, rows[0]);
        });
    });
```

Example – passport.js (Part 3)

```
passport.use(
        'login',
        new LocalStrategy({
            // by default, local strategy uses username and password
            usernameField : 'username',
            passwordField : 'password',
            passReqToCallback : true // allows us to pass back the entire request to the callback
        },
        function(req, username, password, done) { // callback with username and password from our form
            connection.query("SELECT * FROM users WHERE username = ?",[username], function(err, rows){
                if (err)
                    return done(err);
                if (!rows.length) {
                    return done(null, false, req.flash('loginMessage', 'No user found.')); // req.flash is the way to set
flashdata using connect-flash
                }

                // if the user is found but the password is wrong
                if (!bcrypt.compareSync(password, rows[0].password))
                    return done(null, false, req.flash('loginMessage', 'Oops! Wrong password.')); // create the
loginMessage and save it to session as flashdata

                // all is well, return successful user
                return done(null, rows[0]);
            });
        })
    );
```

Example – app.js (Part 1)

```
//set up ======================================================================
//get all the modules we need
var express  = require('express');
var session  = require('express-session');
var app      = express();
var port     = process.env.PORT || 3000;
var flash    = require('connect-flash');

var passport = require('passport');
//connect to our database
require('./config/passport')(passport); // pass passport for configuration
app.use(express.bodyParser());
app.use(express.cookieParser());
app.set('view engine', 'ejs');
app.use(session({
        secret: 'cmpe273_testing',
        resave: true,// forces the session to store every time, even when no session data has been
modified with the request
        saveUninitialized: true,//Forces a new session to be saved to the memory
        duration: 30 * 60 * 1000,// how long the session will stay valid in ms
        activeDuration: 5 * 60 * 1000 // if expiresIn < activeDuration, the session will be extended by
activeDuration milliseconds

} )); // session secret
app.use(passport.initialize());
app.use(passport.session()); // persistent login sessions
app.use(flash()); // use connect-flash for flash messages stored in session
```

Example – app.js (Part 2)

```
function isLoggedIn(req, res, next) {
		// if user is authenticated in the session, carry on
		if (req.isAuthenticated())
			return next();
		// if they aren't redirect them to the home page
		res.redirect('/');
}

//HOME PAGE (with login links)
app.get('/', function(req, res) {
		res.render('index' , { message: req.flash('loginMessage') }); // load the index.ejs file
});

//show the login form
app.get('/login', function(req, res) {
		res.render('login');
});

//process the login form
app.post('/login', passport.authenticate('login', {
		successRedirect : '/profile', // redirect to the secure profile section
		failureRedirect : '/login', // redirect back to the signup page if there is an error
		failureFlash : true // allow flash messages
}),
function(req, res) {
		req.session.cookie.maxAge = 1000 * 60 * 30;
		res.redirect('/');
});
```

Example – app.js (Part 3)

```
//show the signup form
app.get('/signup', function(req, res) {
                res.render('signup' , { message: req.flash('signupMessage') });
});

//process the signup form
app.post('/signup', passport.authenticate('signup', {
                successRedirect : '/profile', // redirect to the secure profile section
                failureRedirect : '/signup', // redirect back to the signup page if there is an error
                failureFlash : true // allow flash messages
}));

//we will want this protected so you have to be logged in to visit
//we will use route middleware to verify this (the isLoggedIn function)
app.get('/profile', isLoggedIn, function(req, res) {
                res.render('profile', {
                                user : req.user // get the user out of session and pass to template
                });
});

//LOGOUT
app.get('/logout', function(req, res) {
                req.logout();
                res.redirect('/');
});
```

References

- client-sessions documentation:

  https://github.com/mozilla/node-client-sessions

- Default sessions Node.js (client-sessions):

  https://stormpath.com/blog/everything-you-ever-wanted-to-know-about-node-dot-js-sessions/

- External sessions – Redis/MongoDB:

  http://blog.modulus.io/nodejs-and-express-sessions

- Passport-MySQL implemented sample - GitHub:

  https://github.com/manjeshpv/node-express-passport-mysql

- Passportjs Website:

  http://passportjs.org/