**Security**

This project will build upon the RESTful API that has access to the music database and display information accordingly. Add authentication to this project using the [passport-local strategy](https://github.com/jaredhanson/passport-local). To do this, create a form that handles account creation. When an account is created, a user should be saved to the session. Check for this user when accessing your existing routes, a user should only be able to access information if they are authenticated.

Instructor Grey [12:07]

Hey @Lisa B-E

Per our conversation, here is the breakdown for the Security Hands-On, with source code

https://github.com/isaacg11/passport-local-auth

1. Install all needed modules: express, bodyParser, sqlite3, etc. (see package.json for full details)

2. Setup `bodyParser` to accept multi-part data ("forms") and parse it into JSON format

3. Connect to your Sqlite Db via Sequalize

4. In Sqlectron, create new "User" table with user properties (see index.js for full details)

5. Create "User" schema

6. Initialize Passport

7. Setup the serialization/deserialization middleware to save the user info to the server session

8. Setup the authentication strategy that will verify a username exists and the password is correct

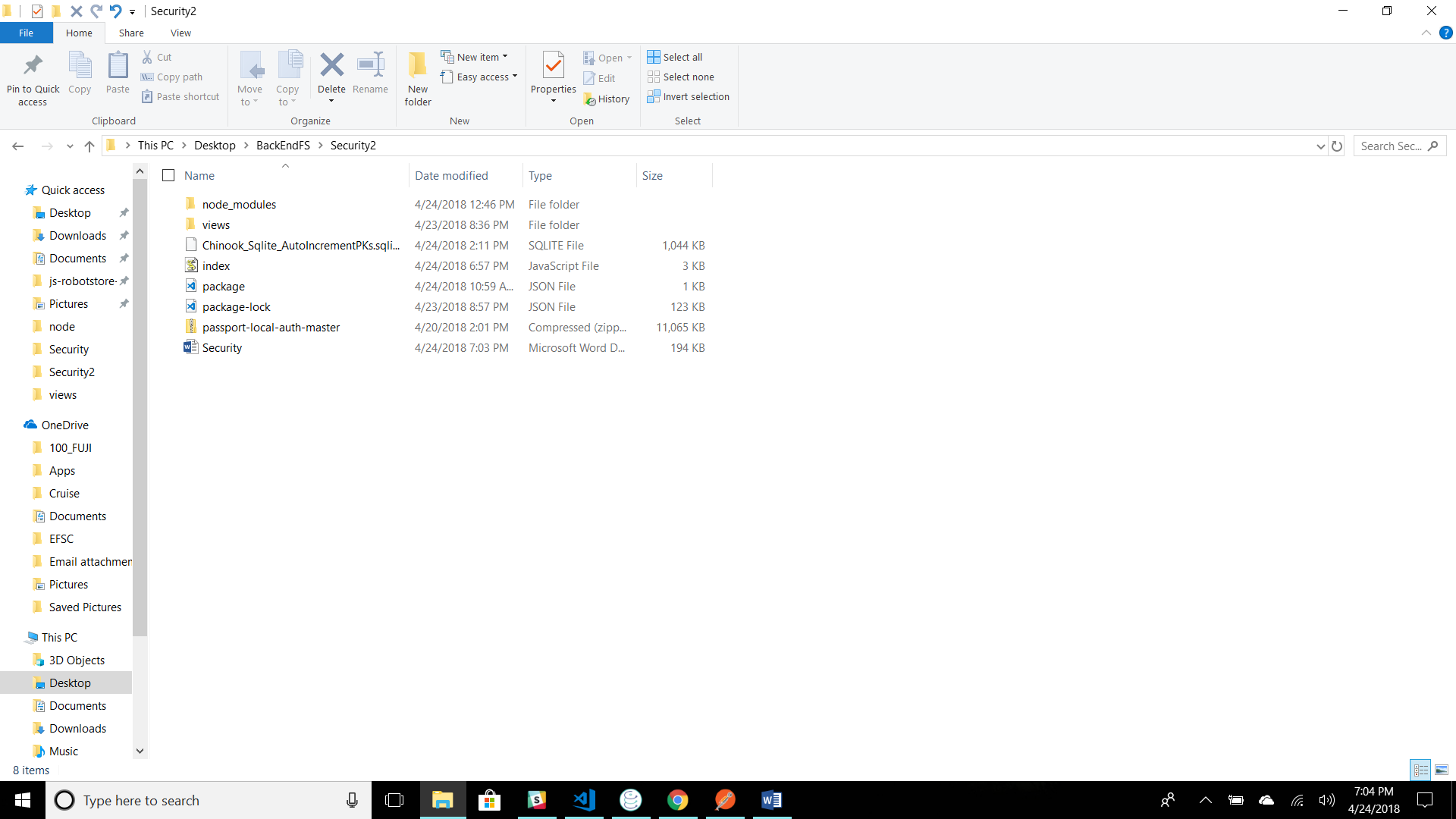
9. Setup `/signup` endpoint which will save a new User to the database, use Crypto to encrypt the password and store the salt & hash.

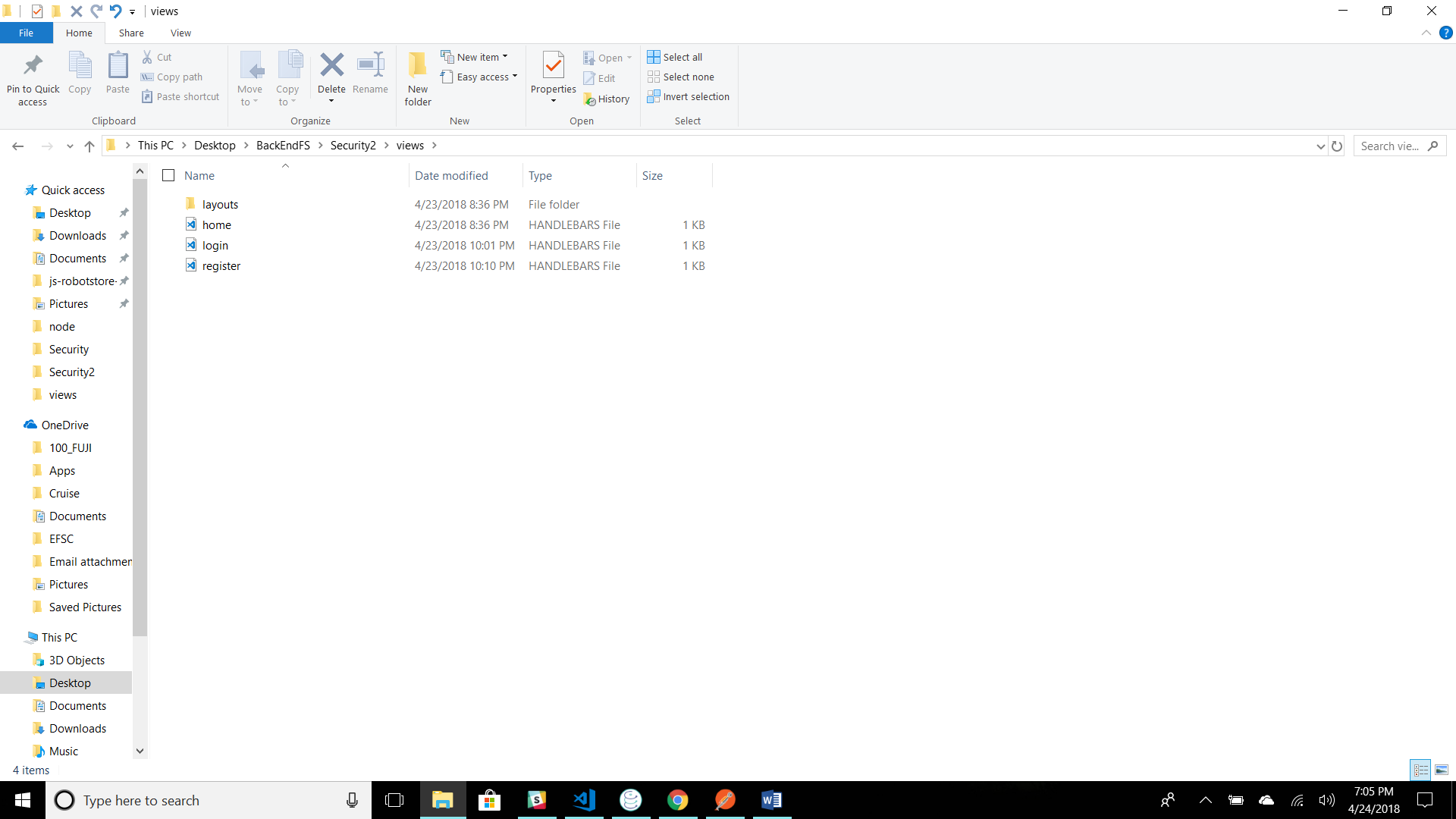
10. Setup `/login` endpoint which will authenticate the User

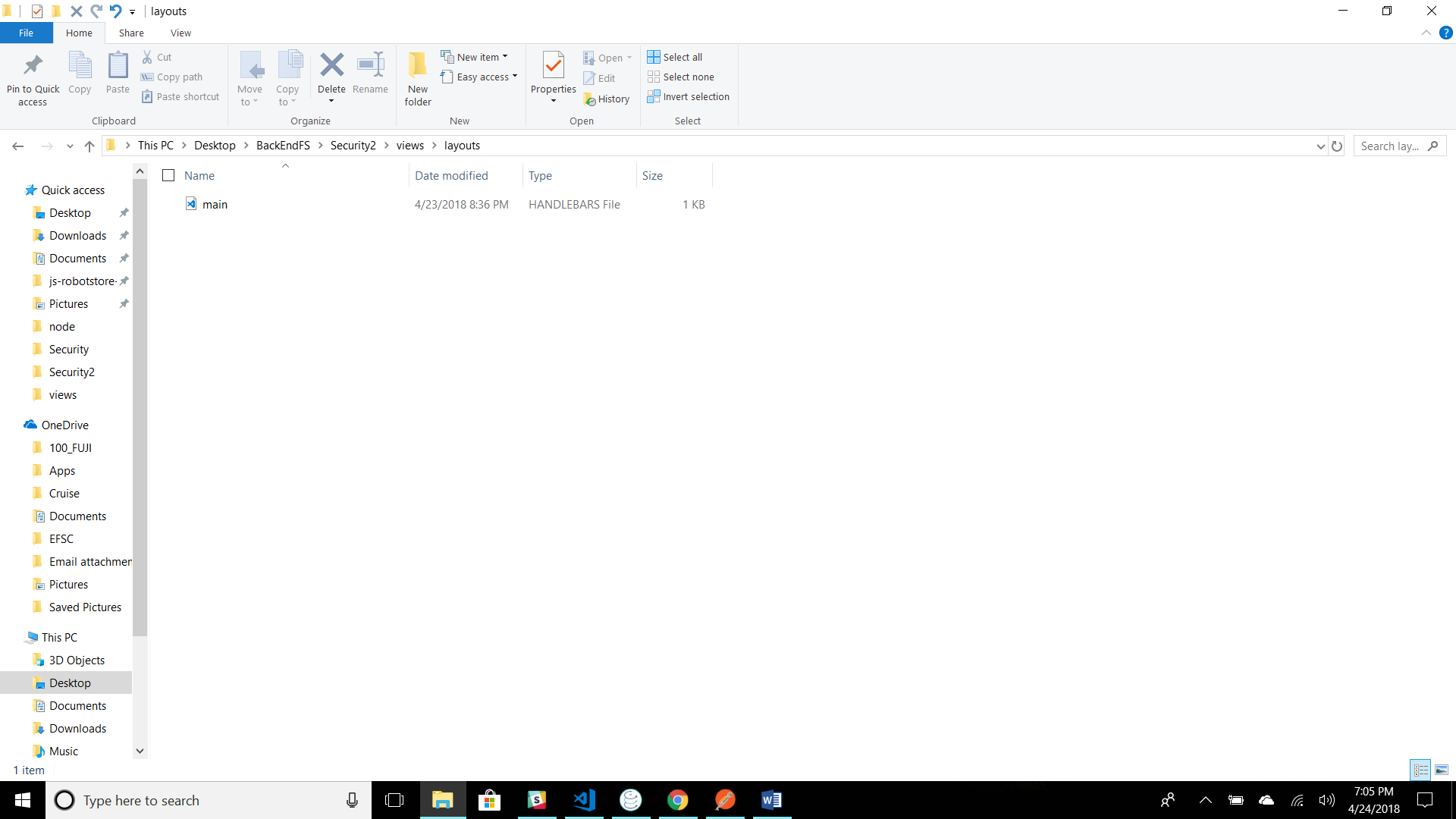
11. Configure GET requests to only allow navigation to a new page if the User is authenticated

12. Setup Handlebars with two forms, one for posting to `/signup`, the other for posting to `/login`

Hope this helps, let me know if you have any questions!







// module imports

const express = require('express');

const bodyParser = require('body-parser');

const sqlite3 = require('sqlite3');

const Sequelize = require('sequelize');

const crypto = require('crypto');

const passport = require('passport');

const LocalStrategy = require('passport-local').Strategy;

const app = express();

const router = express.Router();

const handlebars = require("express-handlebars").create({ defaultLayout: 'main' });

app.engine('handlebars', handlebars.engine);

app.set('view engine', 'handlebars');

// body parser config

app.use(bodyParser.json())

app.use(bodyParser.urlencoded({

extended: true

}));

// connect to db

const sequelize = new Sequelize('database', 'YOUR\_NAME', null, {

host: 'localhost',

dialect: 'sqlite',

storage: "./Chinook\_Sqlite\_AutoIncrementPKs.sqlite"

});

// define schema

const User = sequelize.define('User', {

id: {

type: Sequelize.INTEGER,

autoIncrement: true,

primaryKey: true

},

username: Sequelize.STRING,

passwordHash: Sequelize.STRING,

salt: Sequelize.STRING

},

{

freezeTableName: true,

timestamps: false

})

// auth config

app.use(passport.initialize());

passport.serializeUser(function(user, done){

done(null, user);

});

passport.deserializeUser(function(obj, done){

done(null, obj);

});

passport.use(new LocalStrategy(function(username, password, done){

User.findOne({ where: {username: username} }).then(user => {

if(!user){

return done (null, false, {message: "Invalid user"});

} else if(crypto.pbkdf2Sync(password, user.salt, 1000, 64, 'sha1').toString('hex') !== user.passwordHash) {

return done(null, false, {message: 'Incorrect password'});

} else {

return done (null, user);

}

})

}));

// API

app.get('/', (req, res) => {

    res.render('register')

})

app.get('/login', (req, res) => {

    res.render('login')

})

app.post('/login', passport.authenticate('local'), (req, res) => {

res.render('home')

})

app.get('/register', (req, res) => {

    res.render('register')

})

app.post('/register', (req, res) => {

let salt = crypto.randomBytes(16).toString('hex');

let passwordHash = crypto.pbkdf2Sync(req.body.password, salt, 1000, 64, 'sha1').toString('hex');

User.create({

username: req.body.username,

passwordHash: passwordHash,

salt: salt

})

res.render('home')

})

// run server on port 3000

app.listen(3000, () => {

console.log('server running')

})

LOGIN.HANDLEBARS

<strong><p>Welcome!</strong> <strong><p>Please Log In</p></strong>

<form action="/login" method="post">

<div>

<label>Username:</label>

<input type="text" name="username"/>

</div>

<div>

<label>Password:</label>

<input type="password" name="password"/>

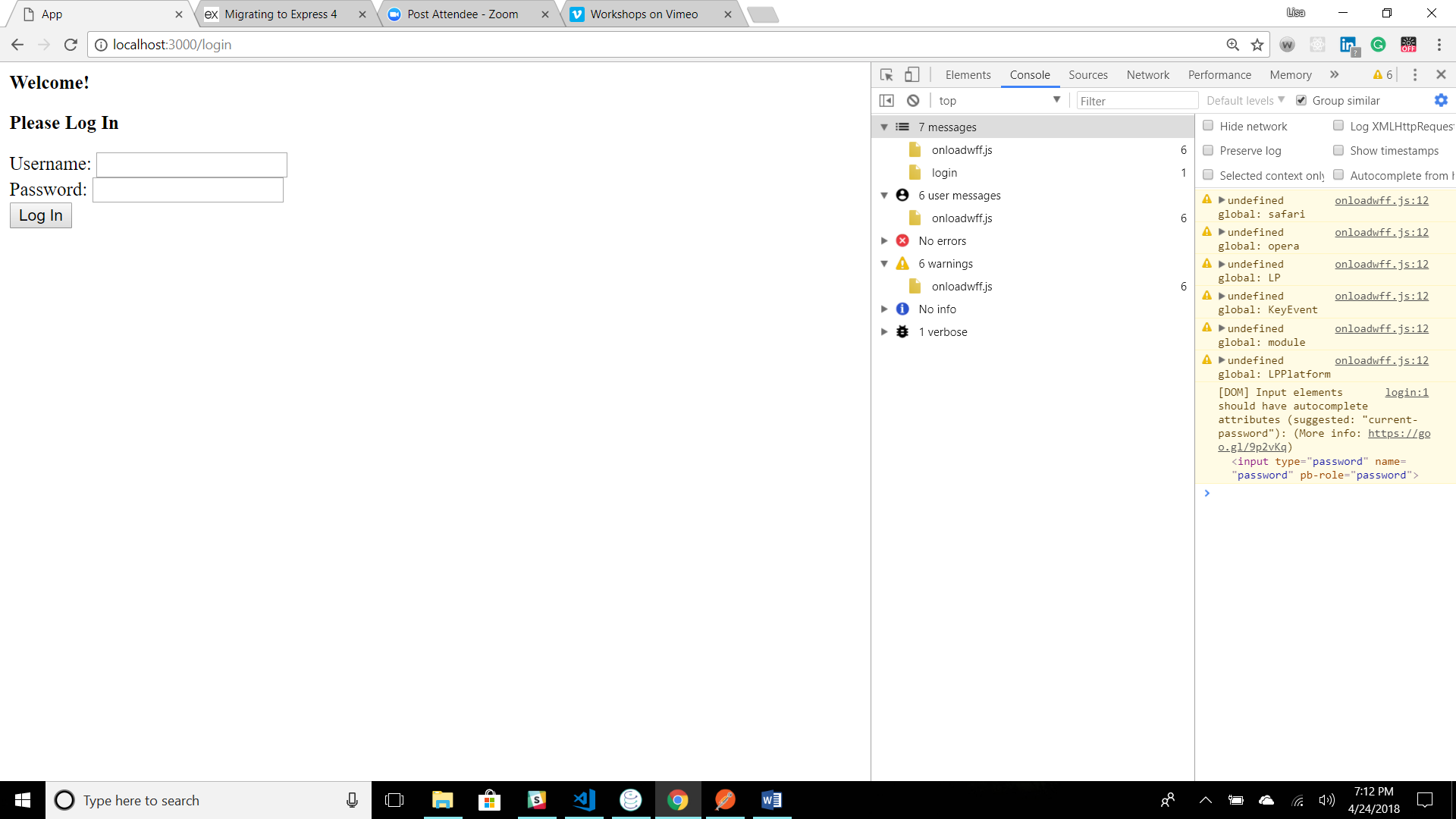
</div>

<div>

<input type="submit" value="Log In"/>

</div>

</form>



Register.handlebars

<strong><p>Create New Account</p></strong>

<form action="/register" method="post">

  <label>Username:</label>

  <input type="text" name="username"/><br/>

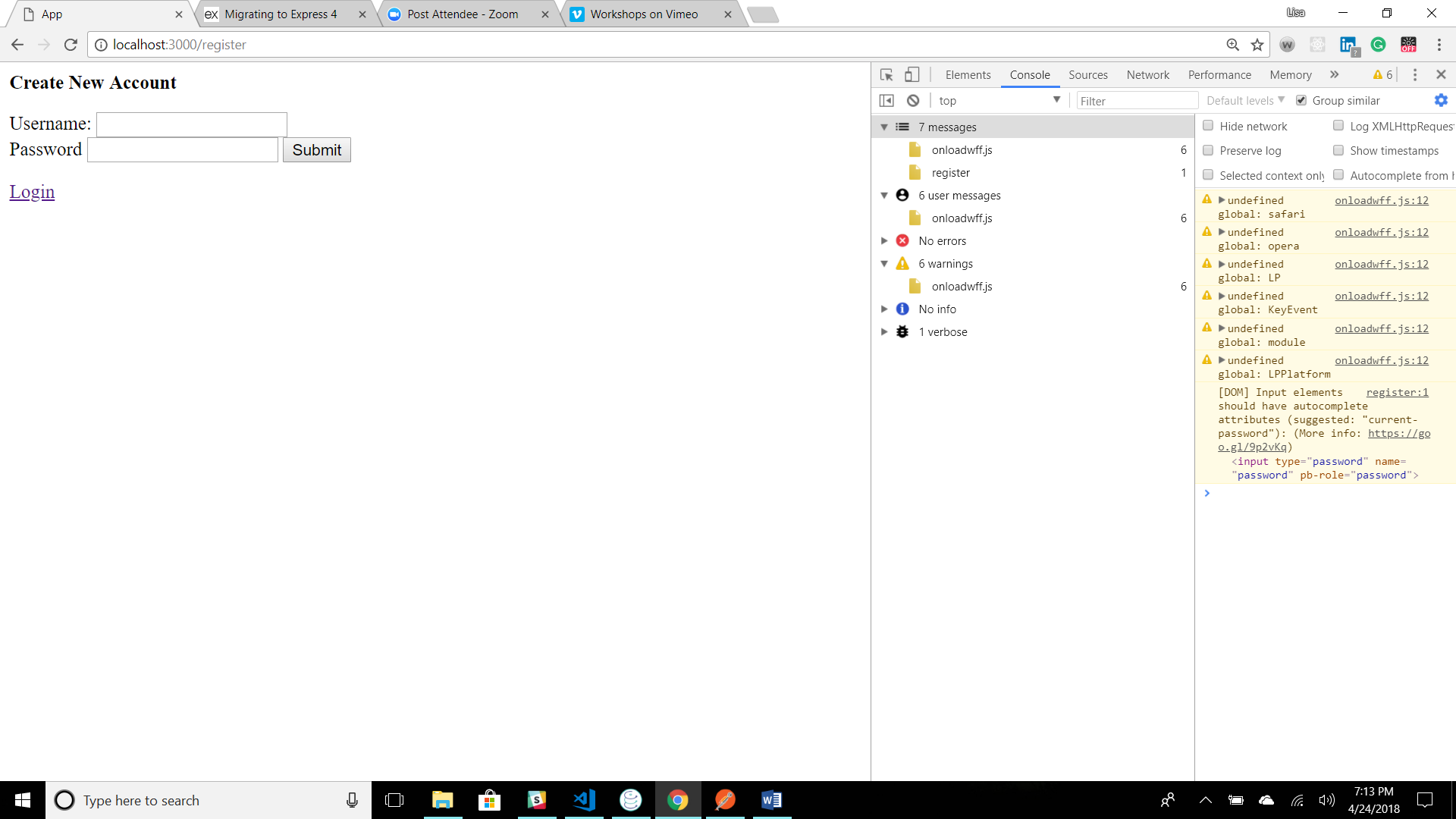
  <label>Password</label>

    <input type="password" name="password">

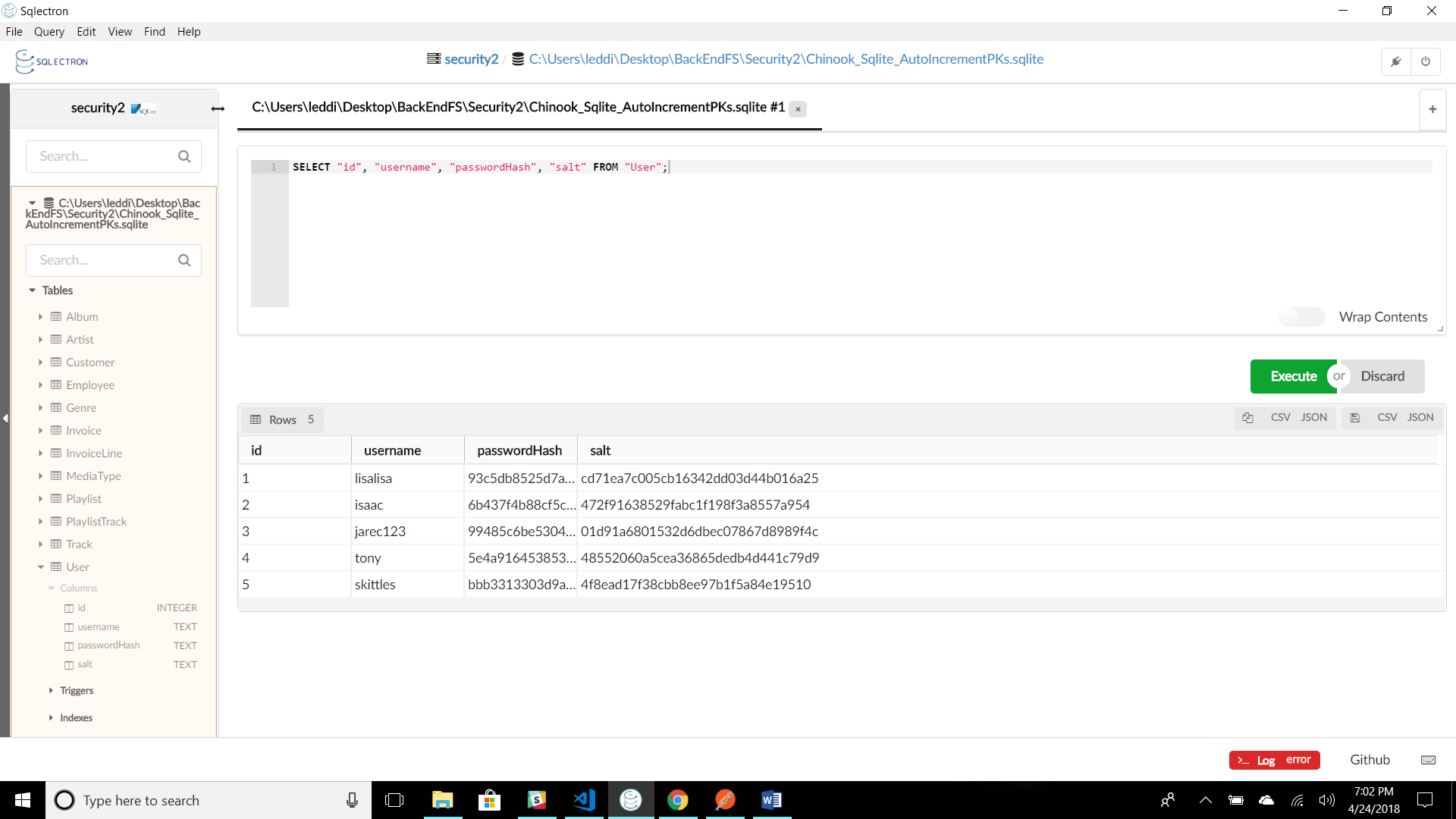
   <input type="submit" value="Submit"/>

  </form>

<a href="\login">Login</a>



Database created



When a user is created correctly through the register page or logged in through the login page, page redirects to Home. If an unauthorized user or incorrect password is given, the screen shows Unauthorized. It seems then that the error message code written in the passport.use function is being completely ignored although the app does tell you when an entry is unauthorized. Is this built into passport-local?