GitHub OAuth2.0 Authentication — using Node JS and PassportJS

This module allows you to use GitHub authentication in your Node.js apps. GitHub authentication may be simply and unobtrusively implemented into any application or framework that supports Connect-style middleware, such as Express, by linking into Passport.

## What is Git Hub

GitHub is a web-based version control repository with around 20 million users worldwide. It is the world's largest source code repository, with 75 million repositories. You might want to implement a GitHub social login to give your store's GitHub users a simpler user experience. This is, indeed, a good concept. 20 million users aren’t a little number: it's a massive number. Customers that buy from you online are more likely to have a GitHub account. As a result, allowing customers to log in using their GitHub accounts is a good way to increase client enrollment. GitHub API details, Client ID, and Client Secret are required for this social login connection.

## What is Passport

Passport is a Node.js authentication middleware. Passport is a highly versatile and modular web application framework that can be seamlessly integrated into any Express-based web application. Authentication with a username and password, as well as Facebook, Twitter, and other social media platforms, is supported by a diverse collection of mechanisms.

## What are strategies in Passport:

Authenticating requests is the responsibility of strategies, which they accomplish by implementing an authentication mechanism. Authentication techniques specify how to encode a credential in a request, such as a password or a claim from an identity provider (IdP). There is 500+ strategies in Passport. You can also read about other strategies at: <https://www.passportjs.org/packages/>

## What is middleware’s:

Middleware sits in the gap between an original request and the final desired path. Middleware features are frequently called in the order in which they are introduced to the stack. Middleware is commonly used to conduct URL-encoded activities, such as JSON request body parsing and cookie parsing for easier cookie management.

### Installation:

npm install passport-github2

## Initial Setup:

1. Create an empty directory with any name you want.
2. Open the folder with any code editor. I would like to prefer VS Code.
3. Now press ctrl + shift + ~ to open terminal.
4. Now type ***npm init -y*** and hit enter. It will create package.json file and install necessary node modules
5. Now type ***npm install express nodemon passport cors dotenv cookie-session npm install passport-github2***

## Registering application

You must first register an application with GitHub before using passport-github2. Please follow following steps to register an application:

1. First, visit the following link [Creating an app in GitHub](https://github.com/settings/applications/new)

Graphical user interface, text, application, email

Description automatically generated

1. Now name your application and fill out all credentials.

Graphical user interface, text, application, email

Description automatically generated

1. Now click on register application and copy client id and client secret id by clicking on generate secret id.

Graphical user interface, text, application

Description automatically generated

1. Now create .env file in root directory and paste following content in it

PORT = 5000

GITHUB\_CLIENT\_ID = your client  id here

GITHUB\_CLIENT\_SECRET = your client secret id here

1. Also in package.json update scripts as:

"scripts": {

    "start": "nodemon server.js"

  },

### Configuring Strategy

The GitHub authentication approach uses a GitHub account and OAuth 2.0 tokens to verify users. A verify callback is required by the strategy, which accepts these credentials and calls completed with a user, as well as options defining a client ID, client secret, and callback URL.

In your src directory, create file server.js and put below code in it.

const express = require("express");

const dotEnv = require("dotenv");

const cors = require("cors");

const app = express();

const passport = require("passport");

const cookieSession = require("cookie-session");

dotEnv.config(); //for reading data from env file

app.use(express.json()); //for fetching data from request body

app.use(cors()); //for cross origin requests

const port = process.env.PORT || 5000;

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

  console.log(res.send("api is running..."));

});

app.listen(port, () => {

  console.log(`port is running on ${port}`);

});

Now create another file passport.js.

### Directory Structure:

Graphical user interface, text, application

Description automatically generated

Put following code in passport.js

### Passport.js:

/\*\*

 \* Configuring Strategy

 \*/

const passport = require("passport");

const GitHubStrategy = require("passport-github2").Strategy;

passport.serializeUser(function (user, done) {

  done(null, user);

});

passport.deserializeUser(function (user, done) {

  done(null, user);

});

passport.use(

  new GitHubStrategy(

    {

      clientID: process.env.GITHUB\_CLIENT\_ID,

      clientSecret: process.env.GITHUB\_CLIENT\_SECRET,

      callbackURL: "http://localhost:5000/github/callback",

    },

    function (accessToken, refreshToken, profile, done) {

      User.findOrCreate({ githubId: profile.id }, function (err, user) {

        return done(err, user);

      });

    }

  )

);

### Authenticating Requests:

//failed auth: route

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

  console.log("you are not authorized");

});

//successful auth: route

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

  console.log(res.send(`Welcome ${user}`));

});

app.get(

  "/auth/github",

  passport.authenticate("github", { scope: ["user:email"] })

);

app.get(

  "/github/callback",

  passport.authenticate("github", { failureRedirect: "/login" }),

  function (req, res) {

    // Successful authentication, redirect home.

    res.redirect("/success");

  }

);

Now create file server.js, if you haven’t and put below code in it.

### Finalizing:

const express = require("express");

const dotEnv = require("dotenv");

const cors = require("cors");

const app = express();

const passport = require("passport");

const cookieSession = require("cookie-session");

dotEnv.config(); //for reading data from env file

app.use(express.json()); //for fetching data from request body

app.use(cors()); //for cross origin requests

const port = process.env.PORT || 5000;

//configuring passport strategy

require("./passport");

//registering middlewares

app.use(

  cookieSession({

    name: "github-auth-session",

    keys: ["key1", "key2"],

  })

);

app.use(passport.initialize());

app.use(passport.session());

//registering routes

//failed auth: route

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

  console.log("you are not authorized");

});

//successful auth: route

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

  console.log(res.send(`Welcome ${user}`));

});

app.get(

  "/auth/github",

  passport.authenticate("github", { scope: ["user:email"] })

);

app.get(

  "/github/callback",

  passport.authenticate("github", { failureRedirect: "/login" }),

  function (req, res) {

    // Successful authentication, redirect home.

    res.redirect("/success");

  }

);

app.listen(port, () => {

  console.log(`port is running on ${port}`);

});

Text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Congratulations 😊, we are now authenticated via google. I hope you really enjoyed this tutorial. Stick with us for such awesome tutorials.