

Home > Programming Blog > Node.js

Node Express Passport Facebook Twitter Google GitHub Login

by Didin J. on Aug 30, 2017



Step by step tutorial of Node, Express, Mongoose, Passport.js, Facebook, Twitter, Google, and GitHub login or authentication.

This tutorial shows you how to create multiple social media (Facebook, Twitter, Google and GitHub) authentication (login) using Node, Express, Mongoose and `Passport.js`. This authentication is using OAuth provider which has the different name and authentication keys between Facebook, Twitter, Google and GitHub. If you like to use form based authentication, we already publish authentication or login tutorial using [Node.js](#), [Express.js](#), [Passport.js](#) and [Mongoose.js](#).



Le Firewall des TPE et PME
Facile et abordable



ESSAYEZ MAINTENANT ►

The following tools and library are required for this tutorial:

- [Node.js](#)
- [Express.js](#) Application Generator

- [Passport.js](#), passport-facebook, passport-twitter, passport-google, passport-github Library
- Terminal or Node Command Line
- Text Editor or IDE ([Atom](#), [Netbeans](#), etc)

The first thing before starting the tutorial is making sure that `Node.js` and it's command line installed and working properly. After `Node.js` installed, open terminal or Node command line then type this command.

```
node -v
```

You will see the version like this.

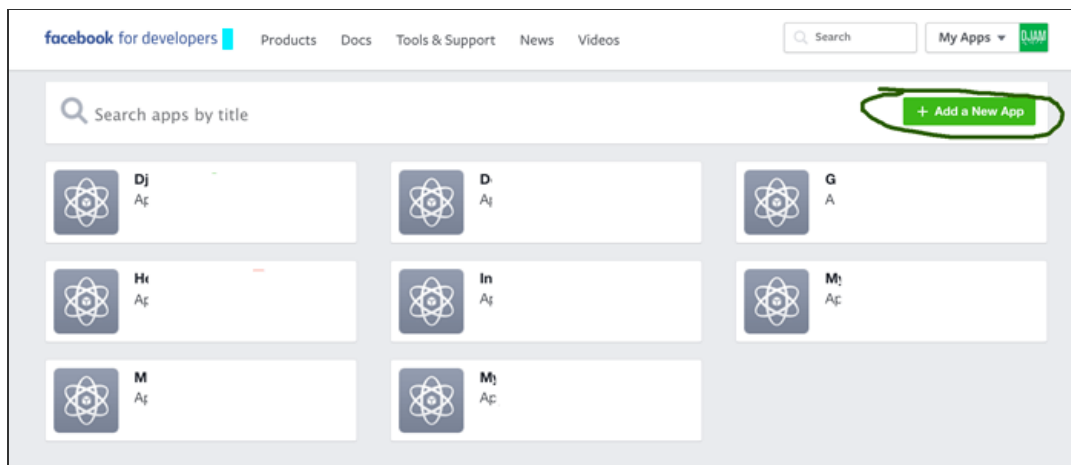
```
v6.9.4
```

Now, install Express generator for generating new Express application by this command.

```
sudo npm install express-generator -g
```

1. Get Facebook APP ID and Secret

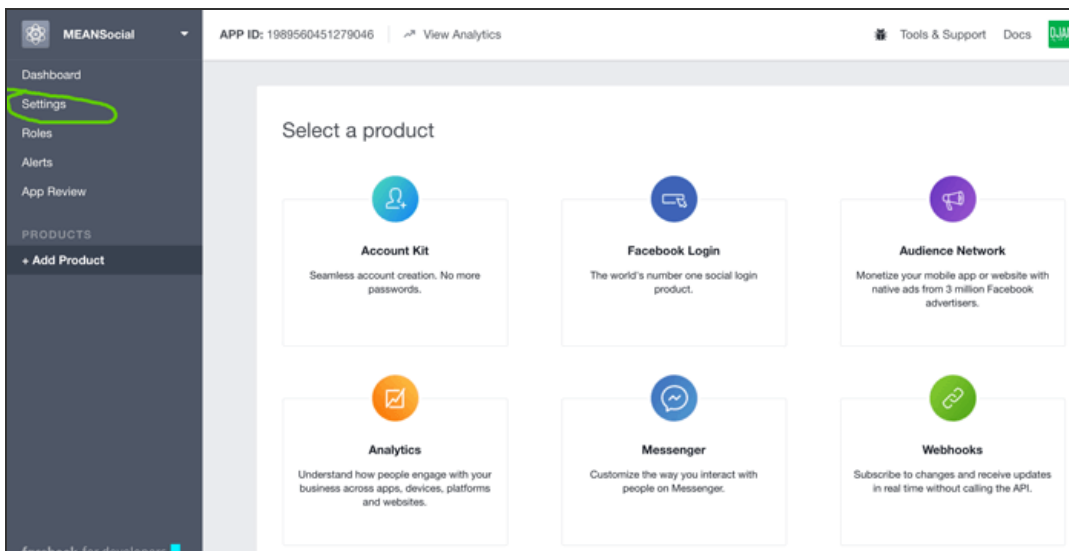
Facebook using APPID and App Secret that use for OAuth provider requirements for authentication (login). Open your browser then go to [Facebook Developer Dashboard](#). Log in using your Facebook developer account or your standard user account.



Click Add a New App button then fill Display Name (ours using `MEANSocial`) and email address.

The image shows the 'Create a New App ID' form on the Facebook Developer Dashboard. The form has a title 'Create a New App ID' and a subtitle 'Get started integrating Facebook into your app or website'. There are two input fields: 'Display Name' with the value 'MEANSocial' and 'Contact Email' with the value 'didin@jamware.com'. At the bottom, there's a checkbox for 'By proceeding, you agree to the Facebook Platform Policies' and two buttons: 'Cancel' and 'Create App ID'.

Click Create App ID button, enter captcha then click Submit button. It will redirect to Facebook App dashboard.



Click Settings menu on the left menu. Click Add Platform then choose the website. Fill site URL with the callback URL for OAuth authentication callback URL.

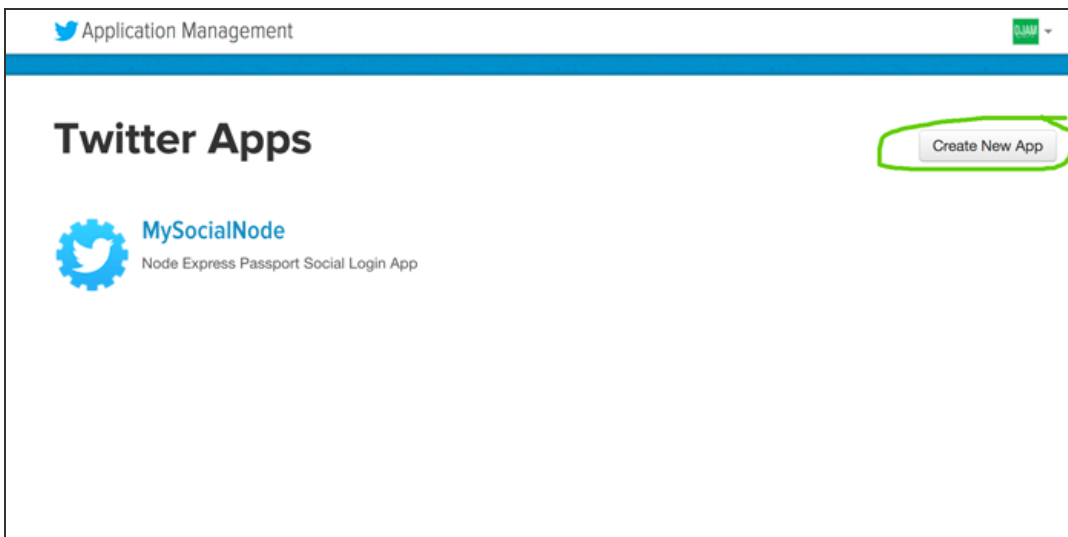
 The image shows the 'Add Platform' configuration form in MEANSocial. It contains the following fields and sections:

- App ID:** 1989560451279046
- App Secret:** A masked field with a 'Show' button.
- Display Name:** MEANSocial
- Namespace:** An empty text field.
- App Domains:** An empty text field.
- Contact Email:** didin@damware.com
- Privacy Policy URL:** Privacy policy for Login dialog and App Details
- Terms of Service URL:** Terms of Service for Login dialog and App Details
- App Icon (1024 x 1024):** A placeholder image with a plus sign and the text '1024 x 1024'.
- Category:** A dropdown menu labeled 'Choose a Category'.
- Website Section:** A section header with a 'Quick Start' button and a close icon.
- Site URL:** http://127.0.0.1:1337auth/facebook/callback, with a small red and blue icon to the right.

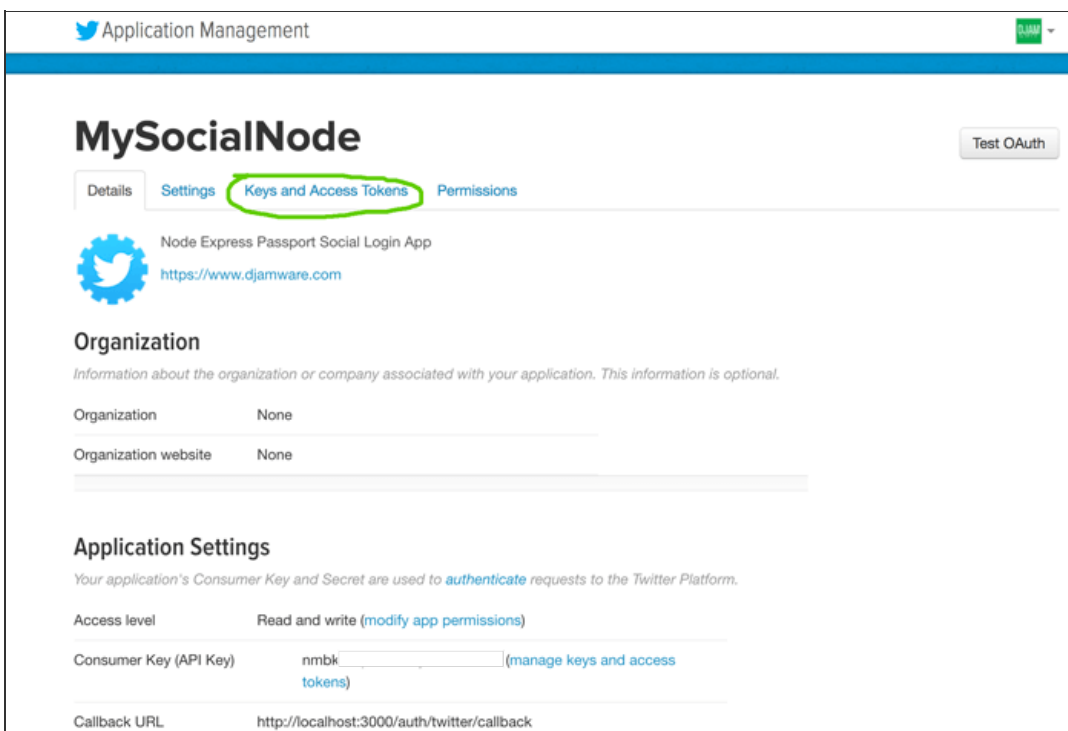
Don't forget to write down the App ID and App Secret to your notepad.

2. Get Twitter Consumer Key (API Key) and Consumer Secret (API Secret)

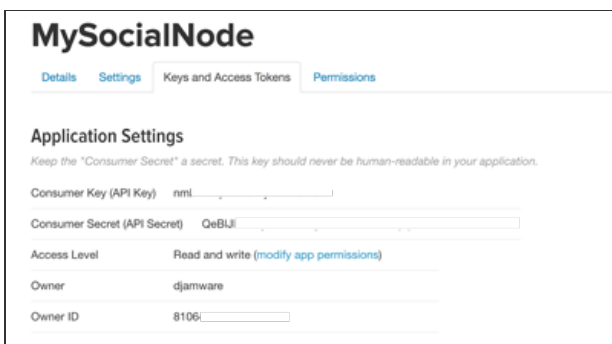
To get Twitter Consumer Key (API Key) and Consumer Secret (API Secret), go to [Twitter Apps console](#). Log in using your existing Twitter account.



Click Create new app button then fill all required fields. Fill callback URL with your MEAN application URL, ours using "http://127.0.0.1:3000/auth/twitter/callback". Check developer agreement then clicks Create Your Twitter Application button. It will redirect to your new application dashboard.



Click on Keys and Access Tokens tab.



Now, write down the Consumer Key (API Key) and Consumer Secret (API Secret) to your notepad.

3. Get Google Client ID and Client Secret

To get the Google Client ID and Client Secret, go to [Google Developer Console](#). Login using your Gmail account. On the selected project drop down click "+" button to create the new project then give it a name.

New Project

? You have 5 projects remaining in your quota. [Learn more.](#)

Project name **?**

Your project ID will be mysocialnode **?** [Edit](#)

Organization **?**
 djamware.com

? You have logged in under a managed account. Your [domain administrator](#) may be able to access, change or suspend any projects created using this account. If you do not want your domain administrator to access your projects, please log out and create a project under an unmanaged Google Account. For more information, please review Google's [Privacy Policy](#).

[Create](#) [Cancel](#)

Click Create button then it will redirect to Project API dashboard.

Google APIs My Social Node **?**

APIs & services **Dashboard** [ENABLE APIS AND SERVICES](#)

Dashboard

Enabled APIs and services

? You don't have permission to view services enabled in this project

Library

Credentials

Click Credentials menu from left menu then click OAuth consent screen tabs to create product name, just fill product name and email then click save button.

Credentials

Credentials **OAuth consent screen** **Domain verification**

Email address **?**

Product name shown to users **?**

Homepage URL (Optional)

Product logo URL (Optional) **?**

? This is how your logo will look to end users
 Max size: 128x128 px

Privacy policy URL
 Optional until you deploy your app

Terms of service URL (Optional)

[Save](#) [Cancel](#)

Back to Credentials tab then click Create Credentials and choose OAuth Client ID. Choose Web Application then fill the Authorized redirect URI with MEAN application URL like previously then click create.

←

Create client ID

Application type

☒ Web application
 ☐ Android [Learn more](#)
☐ Chrome App [Learn more](#)
☐ iOS [Learn more](#)
☐ PlayStation 4
 ☐ Other

Name

Web client 2

Restrictions

Enter JavaScript origins, redirect URIs, or both

Authorized JavaScript origins

For use with requests from a browser. This is the origin URI of the client application. It can't contain a wildcard (http://*.example.com) or a path (http://example.com/subdir). If you're using a nonstandard port, you must include it in the origin URI.

http://www.example.com

Authorized redirect URIs

For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorization code for access. Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.

http://127.0.0.1:3000/auth/google/callbacks

Create

Cancel

Now, you get Client ID and Secret. Write it down on your notepad.

←

Client ID for Web application

DOWNLOAD JSON

RESET SECRET

DELETE

Client ID

591307

Client secret

BagEl

Creation date

Aug 6, 2017, 9:19:25 PM

Name

My Node Social Client

Restrictions

Enter JavaScript origins, redirect URIs, or both

Authorized JavaScript origins

For use with requests from a browser. This is the origin URI of the client application. It can't contain a wildcard (http://*.example.com) or a path (http://example.com/subdir). If you're using a nonstandard port, you must include it in the origin URI.

http://www.example.com

Authorized redirect URIs

For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorization code for access. Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.

http://127.0.0.1:3000/auth/google/callback

×

http://www.example.com/oauth2callback

Save

Cancel

4. Get GitHub Client ID and Client Secret

To get GitHub client ID and client secret, go to [Github developers](#). Log in using your GitHub account.

Search GitHub

Pull requests

Issues

Marketplace

Gist

Personal settings

Profile

Account

Emails

Notifications

Billing

SSH and GPG keys

Security

No OAuth applications

OAuth applications are used to access the GitHub API. [Read the docs](#) to find out more.

Register a new application

Click Register New OAuth Application button. Fill all required fields, use your MEAN application URL for Authorization callback URL then click Register Application button.

Register a new OAuth application

Application name

My Social Node

Something users will recognize and trust

Homepage URL

http://127.0.0.1:3000

The full URL to your application homepage

Application description

My Social Node Web Application

This is displayed to all users of your application

Authorization callback URL

http://127.0.0.1:3000/auth/github/callback


Your application's callback URL. Read our [OAuth documentation](#) for more information.

Register application

Cancel

Now, you get the Client ID and Client Secret. Write down to your notepad.

My Social Node

 **didinj** owns this application.

Transfer ownership

You can list your application in the [GitHub Marketplace](#) so that other users can discover it.

List this application in the Marketplace

0 users

Client ID

e7t

Client Secret

bb07

Revoke all user tokens

Reset client secret

5. Generate Express Application

Now, we will create Node, Express and Mongoose Web Application. Open terminal od Node command line then types this command.

```
express --view=pug node-passport-social
```

That command creates new Express web application with using `pug` view template, this different than the previous tutorial that uses `ejs` or `jade`. Here's the output from that command.

```
create : node-passport-social
create : node-passport-social/package.json
create : node-passport-social/app.js
create : node-passport-social/public
create : node-passport-social/routes
create : node-passport-social/routes/index.js
create : node-passport-social/routes/users.js
create : node-passport-social/views
create : node-passport-social/views/index.pug
create : node-passport-social/views/layout.pug
create : node-passport-social/views/error.pug
create : node-passport-social/bin
create : node-passport-social/bin/www
create : node-passport-social/public/javascripts
create : node-passport-social/public/images
create : node-passport-social/public/stylesheets
create : node-passport-social/public/stylesheets/style.css
```

install dependencies:

```
$ cd node-passport-social && npm install
```

run the app:

```
$ DEBUG=node-passport-social:* npm start
```

Now, do as terminal output command. Type or paste this command.

```
cd node-passport-social && npm install
```

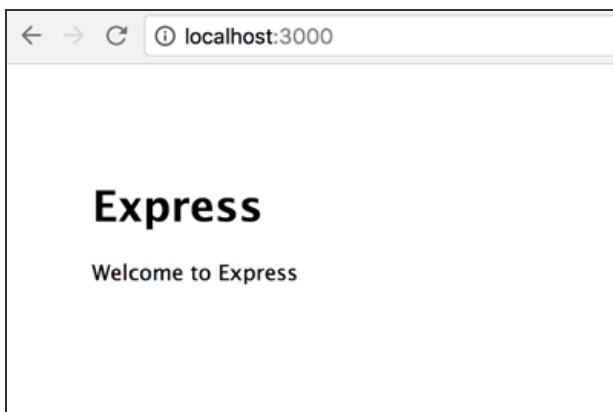
Next, check if the Express web application working by type this command.

```
nodemon
```

or

```
npm start
```

Open the browser then go to `http://localhost:3000`, you will see this page if everything is working properly.



6. Add Mongoose as MongoDB and Node ORM

Now, we will add Mongoose module and dependencies. Stop the app by press `CTRL+C` key. Type this command to install Mongoose the modules and dependencies.



```
npm install mongoose --save
```

Also, install this dependency to make easy query find or create data.

```
npm install find-or-create --save
```


Open and edit app.js on the root of the project. Add require line for Mongoose.

```
var mongoose = require('mongoose');
```

Create a connection to MongoDB database. Add this lines of codes after the require section.

```
mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost/mean-social', { useMongoClient: true })
  .then(() => console.log('connection successful'))
  .catch((err) => console.error(err));
```

Create User model and models folder by typing this command.

```
mkdir models
touch models/User.js
```

Open and edit `User.js` model then add this lines of codes.

```
var mongoose = require('mongoose');

var UserSchema = new mongoose.Schema({
  name: String,
  userid: String,
  updated_at: { type: Date, default: Date.now },
});

UserSchema.statics.findOrCreate = require("find-or-create");

module.exports = mongoose.model('User', UserSchema);
```

7. Add and Configure Passport Facebook Twitter Google and GitHub Module and Dependencies

Install the following Passport, Passport-Facebook, Passport-Twitter, Passport-Google and Passport-GitHub dependencies and modules.

```
npm install passport passport-facebook passport-twitter passport-google-oauth passport-github --save
```

Also, we need to install Express Session dependencies.

```
npm install express-session --save
```

Now, add Passport and Express Session require to `app.js`.

```
var passport = require('passport');
var session = require('express-session');
```

Then add the use of both requires along with another use method.

```
app.use(session({
  secret: 's3cr3t',
  resave: true,
  saveUninitialized: true
}));
app.use(passport.initialize());
app.use(passport.session());
```

Next, create a folder on the root of project folder for holds Passport Facebook, Twitter, Google and GitHub Strategy files.

```
mkdir auth
```

Now, create Passport Facebook strategy file.

```
touch auth/facebook.js
```

Open and edit `auth/facebook.js` then add this lines of codes.

```

var passport = require('passport')
, FacebookStrategy = require('passport-facebook').Strategy;
var User = require('../models/User');

passport.use(new FacebookStrategy({
  clientID: "159030901322260",
  clientSecret: "0d641e47f5d55af221ec80346f3f2d43",
  callbackURL: "http://127.0.0.1:3000/auth/facebook/callback"
}),
function(accessToken, refreshToken, profile, done) {
  User.findOrCreate({name: profile.displayName}, {name: profile.displayName,userid: profile.id}, function(err, user) {
    if (err) { return done(err); }
    done(null, user);
  });
});

module.exports = passport;

```

Create Passport Twitter strategy file.

```
touch auth/twitter.js
```

Open and edit `auth/twitter.js` then add this lines of codes.

```

var passport = require('passport')
, TwitterStrategy = require('passport-twitter').Strategy;
var User = require('../models/User');

passport.serializeUser(function (user, fn) {
  fn(null, user);
});

passport.deserializeUser(function (id, fn) {
  User.findOne({_id: id.doc._id}, function (err, user) {
    fn(err, user);
  });
});

passport.use(new TwitterStrategy({
  consumerKey: "nmbk1uqKB0rbWjBxrPv9iksEf",
  consumerSecret: "QeBIJHanPy232ZbOhYPisfl8hLLUVMujXjul7Sz0Ym4o6m7eGF",
  callbackURL: "http://127.0.0.1:3000/auth/twitter/callback"
}),
function(accessToken, refreshToken, profile, done) {
  User.findOrCreate({name: profile.displayName}, {name: profile.displayName,userid: profile.id}, function(err, user) {
    if (err) {
      console.log(err);
      return done(err);
    }
    done(null, user);
  });
});

module.exports = passport;

```

Create Passport Google strategy file.

```
touch auth/google.js
```

Open and edit `auth/google.js` then add this lines of codes.

```

var passport = require('passport');
var GoogleStrategy = require('passport-google-oauth').OAuth2Strategy;
var User = require('../models/User');

passport.use(new GoogleStrategy({
  clientID: "591307876438-4nm817vks785u467lo22kss40kqno2.apps.googleusercontent.com",
  clientSecret: "BagENe4LxG_PZ_qz2oFX7Aok",
  callbackURL: "http://127.0.0.1:3000/auth/google/callback"
}),
function(accessToken, refreshToken, profile, done) {
  User.findOrCreate({ userid: profile.id }, { name: profile.displayName, userid: profile.id }, function (err, user) {
    return done(err, user);
  });
});

module.exports = passport;

```

Create Passport GitHub strategy file.

```
touch auth/github.js
```

Open and edit `auth/github.js` then add this lines of codes.

```

var passport = require('passport')
, GitHubStrategy = require('passport-github').Strategy;
var User = require('../models/User');

passport.use(new GitHubStrategy({
  clientID: "e7b10decd2ed4ef13816",
  clientSecret: "bb073a53914d014f328de98ad9fe5a3cff366912",
  callbackURL: "http://127.0.0.1:3000/auth/github/callback"
}),
function(accessToken, refreshToken, profile, done) {
  User.findOrCreate({userid: profile.id}, {name: profile.displayName,userid: profile.id}, function (err, user) {
    return done(err, user);
  });
});

module.exports = passport;

```

Next, create `auth.js` file inside `routes` folder.

```
touch routes/auth.js
```

Open and edit `routes/auth.js` then declare all required to Facebook, Twitter, Google and GitHub passport.

```

var express = require('express');
var router = express.Router();
var passportFacebook = require('../auth/facebook');
var passportTwitter = require('../auth/twitter');
var passportGoogle = require('../auth/google');
var passportGitHub = require('../auth/github');

```

Add routes for login page.

```

/* LOGIN ROUTER */
router.get('/login', function(req, res, next) {
  res.render('login', { title: 'Please Sign In with:' });
});

```

Add routes for logout from logged in session.

```

/* LOGOUT ROUTER */
router.get('/logout', function(req, res){
  req.logout();
  res.redirect('/');
});

```

Add routes for Facebook, Twitter, Google and Github login.

```

/* FACEBOOK ROUTER */
router.get('/facebook',
  passportFacebook.authenticate('facebook'));

router.get('/facebook/callback',
  passportFacebook.authenticate('facebook', { failureRedirect: '/login' }),
  function(req, res) {
    // Successful authentication, redirect home.
    res.redirect('/');
  });

/* TWITTER ROUTER */
router.get('/twitter',
  passportTwitter.authenticate('twitter'));

router.get('/twitter/callback',
  passportTwitter.authenticate('twitter', { failureRedirect: '/login' }),
  function(req, res) {
    // Successful authentication, redirect home.
    res.redirect('/');
  });

/* GOOGLE ROUTER */
router.get('/google',
  passportGoogle.authenticate('google', { scope: 'https://www.google.com/m8/feeds' }));

router.get('/google/callback',
  passportGoogle.authenticate('google', { failureRedirect: '/login' }),
  function(req, res) {
    res.redirect('/');
  });

/* GITHUB ROUTER */
router.get('/github',
  passportGitHub.authenticate('github', { scope: [ 'user:email' ] }));

router.get('/github/callback',
  passportGitHub.authenticate('github', { failureRedirect: '/login' }),
  function(req, res) {
    // Successful authentication, redirect home.
    res.redirect('/');
  });

module.exports = router;

```

Next, open and edit `routes/users.js` then replace default router with this.

```

/* GET users listing. */
router.get('/', ensureAuthenticated, function(req, res, next) {
  res.render('user', { user: req.user });
});

```

Create `ensureAuthenticated` function before `module.exports`.

```

function ensureAuthenticated(req, res, next) {
  if (req.isAuthenticated()) { return next(); }
  res.redirect('/login')
}

```

Don't forget to register `routes/auth.js` router in `app.js`. Add this lines to `app.js` after another routes variable.



```
var auth = require('./routes/auth');
```

Add `use` method after users `use` method.

```
app.use('/auth', auth);
```

8. Add Login Page for Hold Facebook, Twitter, Google and GitHub Login Button

As you see on router files there's a lot of redirect to login page. For this, we have to create `pug` file inside `views` folder by type this command.

```
touch views/login.pug
```

Open and edit `views/login.pug` then replace all codes with this.

```
extends layout

block content
  .main.container
    .row
      .col-md-6.col-md-offset-3
        h1.display-4.m-b-2= title

    // sign in form
    div.form-group
      a.btn.btn-primary(href='/auth/facebook') Facebook
    div.form-group
      a.btn.btn-info(href='/auth/twitter') Twitter
    div.form-group
      a.btn.btn-danger(href='/auth/google') Google
    div.form-group
      a.btn.black-back(href='/auth/github') Github
```

To make styling easier, we have to use Bootstrap. Type this command to install `pug-bootstrap` module and dependencies.

```
npm install pug-bootstrap --save
```

Open and edit `views/layout` then add the include for `pug-bootstrap`, so it will look like this.

```
doctype html
html
  include ../node_modules/pug-bootstrap/_bootstrap.pug
  head
    title= title
    link(rel='stylesheet', href='/stylesheets/style.css')
  body
    block content
```

Open and edit `public/stylesheets/style.css` and make it like this.

```
body {
  padding: 50px;
  font: 14px "Lucida Grande", Helvetica, Arial, sans-serif;
}

a {
  color: #00B7FF;
}

.black-back {
  background-color: #000000;
  color: #FFFFFF;
}

.btn {
  width: 100%;
  height: 50px;
  font-size: 24px;
}
```

One thing left, is user page that should open when authentication success. Type this command to create the page.

```
touch views/user.pug
```

Open and edit `user.pug` the file then adds this lines of codes.

```
extends layout

block content
  h1= user.name
  p Welcome to #{user.name}
  a(href="/auth/logout") Logout
```

Now, you can run again the application on the browser then go to restricted page `http://127.0.0.1:3000/users` it should redirect to Social Login page. After successful login with one of Social authentication, then it will return back to user page then display the user name on the page.

Yeah, I know this a little bit complicated. For that, please give us your suggestions, critics or any problem report under comments below for this tutorial. If you need the working example, you find on our [GitHub](#).

Thanks

Follow



The following resources might be useful for you:

- [Master essential business skills to advance your career or grow your business.](#)
- [DATA SCIENTIST: THE SEXIEST JOB OF THE 21ST CENTURY](#)

[← Previous Article](#)

Node, Express, Mongoose and Passport.js REST API Authentication

[Next Article →](#)

Related Articles

- [How to Create REST API Easily using Node.js, Express.js, Mongoose.js and MongoDB](#)
 - [How to Create Node.js, Express.js and MongoDB CRUD Web Application](#)
 - [Node.js, Express.js, Mongoose.js and Passport.js Authentication](#)
 - [Tutorial Building CRUD App from Scratch using MEAN Stack \(Angular 2\)](#)
 - [Getting Started Angular 4 using Angular CLI](#)
 - [Building Chat Application using MEAN Stack \(Angular 4\) and Socket.io](#)
 - [Node, Express, Mongoose and Passport.js REST API Authentication](#)
 - [Building CRUD Web Application using MERN Stack](#)
 - [MEAN Stack \(Angular 5\) CRUD Web Application Example](#)
 - [Mongo Express Vue Node.js \(MEVN Stack\) CRUD Web Application](#)
 - [Node.js and MongoDB Slack Bot Example](#)
 - [Setup Node.js, Nginx and MongoDB on Ubuntu 16.04 for Production](#)
 - [Securing MEAN Stack \(Angular 5\) Web Application using Passport](#)
 - [Securing MERN Stack Web Application using Passport](#)
-





Programming Blog

[Ionic Framework \(34\)](#)

[Javascript \(5\)](#)

[HTML 5 Tutorial \(3\)](#)

[Groovy and Grails \(18\)](#)

[MongoDB \(5\)](#)

[Java \(12\)](#)

[Node.js \(15\)](#)

[CSS 3 \(5\)](#)

All Articles

Popular Articles:

- [MEAN Stack \(Angular 5\) CRUD Web Application Example](#)
- [Ionic 3 Consuming REST API using New Angular 4.3 HttpClient](#)
- [Ionic 3 and Angular 4 Mobile App Example](#)
- [Securing MEAN Stack \(Angular 5\) Web Application using Passport](#)
- [Ionic 3 and Angular 5 Mobile App Example](#)
- [Ionic 3, Angular 4 and SQLite CRUD Offline Mobile App](#)
- [How to Upload File on Ionic 3 using Native File Transfer Plugin](#)
- [Step by Step Tutorial of Ionic 3, Angular 4 and Google Maps Directions Service](#)
- [Step by step tutorial building Ionic 2 REST API Authentication](#)
- [Build Ionic 3, Angular 5 and Firebase Simple Chat App](#)

