In this article, you will learn to implement social login in your Node JS backend. We will be creating a single route and controller method for social login. It will be flexible enough to handle any social login such as login with google, facebook, twitter etc. It all depends on the frontend.

The backend will require only user email to sign that user in.  You will get user email along with other information about that user, when the user tries to Login with Google or Login with Facebook.

Let's begin by creating a route in routes/auth.js

**/nodeapi/routes/auth.js**

// import socialLogin method from controllers/auth

const {

signup,

signin,

signout,

forgotPassword,

resetPassword,

socialLogin

} = require("../controllers/auth");

// then use this route for social login

router.post("/social-login", socialLogin);

**/nodeapi/controllers/auth.js**

This method also works  great, even if the user previously signed in via email and password but later tried to login using social networks. Because this method checks for user email, no matter how user sign in (email/password or social network) the email address will be same. So this method returns user based on email. So our user can login either way!

exports.socialLogin = (req, res) => {

// try signup by finding user with req.email

let user = User.findOne({ email: req.body.email }, (err, user) => {

if (err || !user) {

// create a new user and login

user = new User(req.body);

req.profile = user;

user.save();

// generate a token with user id and secret

const token = jwt.sign(

{ \_id: user.\_id, iss: "NODEAPI" },

process.env.JWT\_SECRET

);

res.cookie("t", token, { expire: new Date() + 9999 });

// return response with user and token to frontend client

const { \_id, name, email } = user;

return res.json({ token, user: { \_id, name, email } });

} else {

// update existing user with new social info and login

req.profile = user;

user = \_.extend(user, req.body);

user.updated = Date.now();

user.save();

// generate a token with user id and secret

const token = jwt.sign(

{ \_id: user.\_id, iss: "NODEAPI" },

process.env.JWT\_SECRET

);

res.cookie("t", token, { expire: new Date() + 9999 });

// return response with user and token to frontend client

const { \_id, name, email } = user;

return res.json({ token, user: { \_id, name, email } });

}

});

};

This is it. Now you can move on to the frontend and implement social login.