npm install express mongoose passport passport-jwt jsonwebtoken body-parser bcryptjs validator –save

**Node.js Gravatar library**

$ npm install gravatar

var gravatar = require('gravatar');

gravatar.url(email);

gravatar.url(email, options);

var url = gravatar.url('emerleite@gmail.com', {s: '200', r: 'pg', d: 'mm'});

size, rating, default

**const** bcrypt = require("bcryptjs");

bcrypt.genSalt(10, (error, salt) **=>** {

bcrypt.hash(newUser.password, salt, (error, hash) **=>** {

if (error) throw error;

newUser.password = hash;

newUser.save((err, user) **=>** {

if (err) console.log(err);

else {

res.json(user);

}

});

});

Generate salt and hash the password

Login using bcrypt

router.post("/login", (req, res) **=>** {

**const** email = req.body.email;

**const** password = req.body.password;

User.findOne({ email }, (err, user) **=>** {

if (err) {

console.log(err);

} else if (!user) {

return res.status(404).json({ email: "User not found !" });

} else {

bcrypt.compare(password, user.password).then(isMatch **=>** {

if (isMatch) {

res.json({ msg: "Success" });

} else {

return res.status(400).json({ password: "Password incorrect" });

}

});

}

});

});

On successful login generate a token and pass it to access protected routes .

Use jsonwebtoken for that

**const** jwt = require("jsonwebtoken");

jwt.sign()

payload , what we want to include in the token, i.e. Some user information because later we will decode it and know what user it is. We also need to send some secret or key.

And expiration if we want it to expire in a certain amount of time

**const** jwt = require("jsonwebtoken");

**const** keys = require("../config/keys");

router.post("/login", (req, res) **=>** {

**const** email = req.body.email;

**const** password = req.body.password;

User.findOne({ email }, (err, user) **=>** {

if (err) {

console.log(err);

} else if (!user) {

return res.status(404).json({ email: "User not found !" });

} else {

bcrypt.compare(password, user.password).then(isMatch **=>** {

if (isMatch) {

*//res.json({ msg: "Success" });*

*//User matched*

**const** payload = {

*//Create jwt payload*

is: user.id,

userName: user.userName,

firstName: user.firstName,

lastName: user.lastName,

avatar: user.avatar

};

jwt.sign(payload, keys.secret, { expiresIn: 3600 }, (err, token) **=>** {

res.json({

success: true,

token: "Bearer " + token

});

});

} else {

return res.status(400).json({ password: "Password incorrect" });

}

});

}

});

});

We will get

{

"success": true,

"token": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpcyI6IjVjYjk3NmNiNGU0NWIzMWZkNDAyNWRhYiIsImF2YXRhciI6Ii8vd3d3LmdyYXZhdGFyLmNvbS9hdmF0YXIvZGIzZGZkMTgxNTgwMzQ3NTY4ZGI2MmIwNTRiM2Y1YzY\_cz0yMDAmcj1wZyZkPW1tIiwiaWF0IjoxNTU1NjY5Nzk4LCJleHAiOjE1NTU2NzMzOTh9.yPGKBUBXCy9Gp8lkC2KhCRFqbiE2I-R0nJ7XEMNKbDo"

}

We will send this token in the header to get access to the protected routes

Bearer is a certain type of protocol that we append

So, after successful login we get a payload which contains user information

We need passport to be able to verify that token or make any routes private

Using middleware

Goto app.js

**const** passport = require("passport");

We will use the jwt stratergy

*//Passport middleware*

app.use(passport.initialize());

*//Passport Config*

require("./config/passport")(passport);

create a passport.js file in config folder and there create the stratergy

passport-jwt

A [Passport](http://passportjs.org/) strategy for authenticating with a [JSON Web Token](http://jwt.io/).

**const** JwtStrategy = require("passport-jwt").Strategy;

**const** ExtractJwt = require("passport-jwt").ExtractJwt; *//enable to extract payload*

**const** mongoose = require("mongoose");

**const** User = mongoose.model("Users");

**const** keys = require("../config/keys");

**var** opts = {};

opts.jwtFromRequest = ExtractJwt.fromAuthHeaderAsBearerToken(); *//specify we want to use bearer token*

opts.secretOrKey = keys.secretOrKey; *//add secret or key*

module.exports = passport **=>** {

passport.use(

new JwtStrategy(opts, (jwt\_payload, done) **=>** {

*//create new stratergy and pass in options and a callback function with payload*

User.findById({ \_id: jwt\_payload.id }, (err, user) **=>** {

if (err) {

*//Find the user and validate it*

return done(err, false);

}

if (user) {

return done(null, user);

} else {

return done(null, false);

}

});

})

);

};

Simply add middleware to private routes

*//@route GET api/users/current*

*//@desc Returning current user*

*//@access Private*

router.get(

"/current",

passport.authenticate("jwt", { session: false }),

(req, res) **=>** {

res.json({

id: req.user.id,

userName: req.user.userName,

firstName: req.user.firstName,

lastName: req.user.lastName,

avatar: req.user.avatar

});

}

);

Passport-jwt documentation

Write validations.. Ex: tell email is in wrong format

Use validator package

Goto documentation validator github. One issue is that it has to be a string