In this article, I will guide you through each steps necessary to implement password forgot and password in the Node Js backend.

When a user forget his or her password, they will type in their email address and submit the form requesting the link to be emailed to their email address.

Once they click on that link, they will be taken to a page where they will be able to enter new password. When they enter their new password, we need to apply some validation. Their **new password should not be empty**, s**hould be minimum 6 characters long** and **must include a number** That is how they will be able to reset password.

**Add resetPassword property to the user model**

**nodeapi/models/user.js**

resetPasswordLink: {

data: String,

default: ""

}

**Create password reset validator method.**

**nodeapi/validator/index.js**

exports.passwordResetValidator = (req, res, next) => {

// check for password

req.check("newPassword", "Password is required").notEmpty();

req.check("newPassword")

.isLength({ min: 6 })

.withMessage("Password must be at least 6 chars long")

.matches(/\d/)

.withMessage("must contain a number")

.withMessage("Password must contain a number");

// check for errors

const errors = req.validationErrors();

// if error show the first one as they happen

if (errors) {

const firstError = errors.map(error => error.msg)[0];

return res.status(400).json({ error: firstError });

}

// proceed to next middleware or ...

next();

};

**Create routes for forgot-password and reset-password**

**nodeapi/routes/auth.js**

// import from controllers/auth

// these methods you will create next in controllers/auth

const {

signup,

signin,

signout,

forgotPassword,

resetPassword

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

// import password reset validator

const { userSignupValidator, passwordResetValidator } = require("../validator");

// password forgot and reset routes

router.put("/forgot-password", forgotPassword);

router.put("/reset-password", passwordResetValidator, resetPassword);

Next you will be creating forgotPassword and resetPassword methods in controllers/auth.js

* In forgotPassword method, you will check if the user with that email exists. If not then you will send the error message.
* If that email exists, then you will generate a token and set user model's resetPasswordLink property with the value of token
* Then email that token to the user's email
* Upon clicking on that link, user will be taken to the react frontend app, where we will capture that token which will be available in the route params
* Based on that token we grab from route params (in React app), we will make request to resetPassword (see the resetPassword method below)

**nodeapi/controllers/auth.js**

// add forgotPassword and resetPassword methods

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

if (!req.body) return res.status(400).json({ message: "No request body" });

if (!req.body.email)

return res.status(400).json({ message: "No Email in request body" });

console.log("forgot password finding user with that email");

const { email } = req.body;

console.log("signin req.body", email);

// find the user based on email

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

// if err or no user

if (err || !user)

return res.status("401").json({

error: "User with that email does not exist!"

});

// generate a token with user id and secret

const token = jwt.sign(

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

process.env.JWT\_SECRET

);

// email data

const emailData = {

from: "noreply@node-react.com",

to: email,

subject: "Password Reset Instructions",

text: `Please use the following link to reset your password: ${

process.env.CLIENT\_URL

}/reset-password/${token}`,

html: `<p>Please use the following link to reset your password:</p> <p>${

process.env.CLIENT\_URL

}/reset-password/${token}</p>`

};

return user.updateOne({ resetPasswordLink: token }, (err, success) => {

if (err) {

return res.json({ message: err });

} else {

sendEmail(emailData);

return res.status(200).json({

message: `Email has been sent to ${email}. Follow the instructions to reset your password.`

});

}

});

});

};

// to allow user to reset password

// first you will find the user in the database with user's resetPasswordLink

// user model's resetPasswordLink's value must match the token

// if the user's resetPasswordLink(token) matches the incoming req.body.resetPasswordLink(token)

// then we got the right user

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

const { resetPasswordLink, newPassword } = req.body;

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

// if err or no user

if (err || !user)

return res.status("401").json({

error: "Invalid Link!"

});

const updatedFields = {

password: newPassword,

resetPasswordLink: ""

};

user = \_.extend(user, updatedFields);

user.updated = Date.now();

user.save((err, result) => {

if (err) {

return res.status(400).json({

error: err

});

}

res.json({

message: `Great! Now you can login with your new password.`

});

});

});

};

You must have noticed that we have used process.env.CLIENT\_URL and sendEmail method in the above methods in controllers/auth. First of all lets bring them from .env and helpers/index.js (below I have shown you how to create those files/folders and update .env)

Add this code at the beggining of controllers/auth.js. It includes lodash, sendEmail method from helpers and dotenv to load .env variables.

**nodeapi/controllers/auth.js**

const \_ = require("lodash");

const { sendEmail } = require("../helpers");

// load env

const dotenv = require("dotenv");

dotenv.config();

Now update .env file

**.env**

CLIENT\_URL=http://localhost:3000

For sendEmail(), we need to create a new folder in the root directory and call it helpers. Inside helpers folder you will create index.js

**nodeapi/helpers/index.js**

const nodeMailer = require("nodemailer");

const defaultEmailData = { from: "noreply@node-react.com" };

exports.sendEmail = emailData => {

const transporter = nodeMailer.createTransport({

host: "smtp.gmail.com",

port: 587,

secure: false,

requireTLS: true,

auth: {

user: "youremail@gmail.com",

pass: "kshzlmomlthllktq"

}

});

return (

transporter

.sendMail(emailData)

.then(info => console.log(`Message sent: ${info.response}`))

.catch(err => console.log(`Problem sending email: ${err}`))

);

};

The code above needs some work. First you need to install nodemailer. Run this command:

npm install nodemailer --save

Read more about nodemailer in https://nodemailer.com/about/

Then I have used gmail here to send email. Your regular gmail password will not work here. You need to generate gmail app password.

**Follow this article to Create a Gmail Application-Specific Password for POP/IMAP**

<https://www.lifewire.com/get-a-password-to-access-gmail-by-pop-imap-2-1171882>

<https://myaccount.google.com/u/1/security> (click on APP PASSWORD / SECURITY on left sidebar menu)

Finding your way through google's api can be difficult in times but just be patient and go through the steps mentioned in the above article.

Then look at the existing code in helpers/index.js and update with your email and password

auth: {

user: "youremail@gmail.com",

pass: "fdgkliuewykjhfdslajl"

}

That's it! You have implemented forgot password and reset functionality. Lets give it a try using postman. In the next lesson, you will learn to implement it in the React frontend.

**Testing using Postman**

Make sure you have node api server running: npm run dev

Open postman and make a PUT request to the url

http://localhost:8080/forgot-password

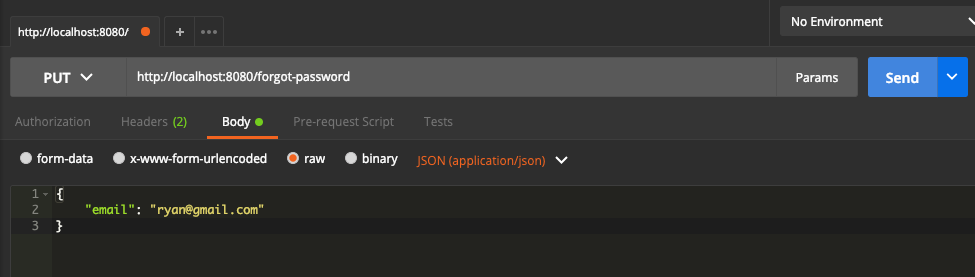
And the email address of the user, who want to request password reset email:

{

"email": "ryan@gmail.com"

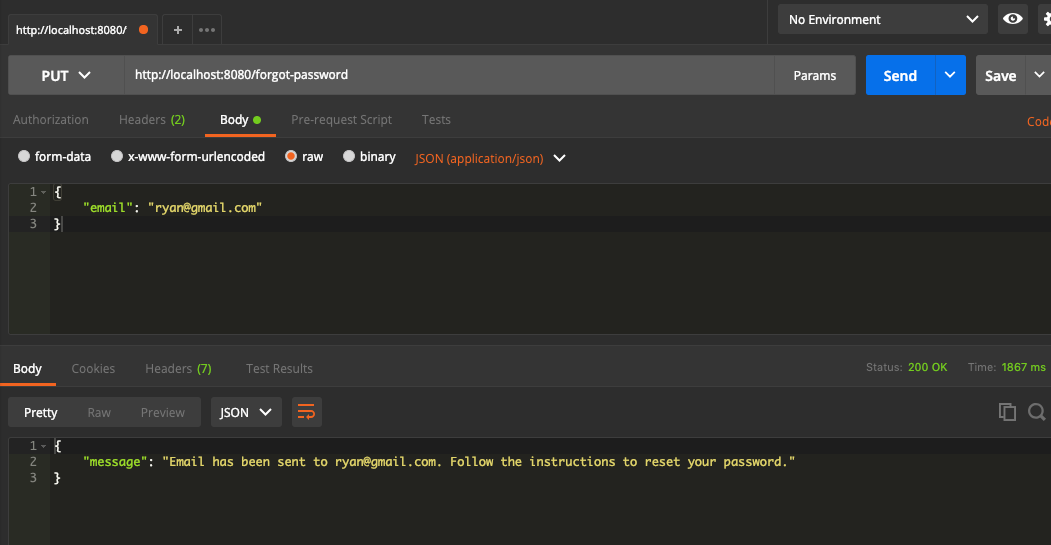
}

Here is the screenshot:



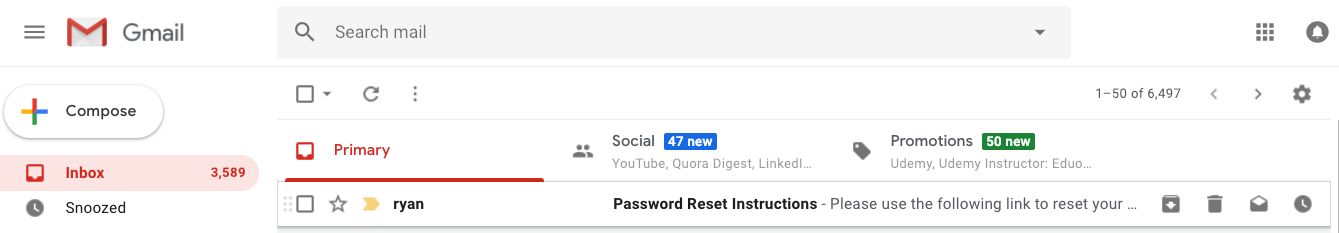
Keep an eye on the terminal console to see if any error occoured. Otherwise you should get the token as response.

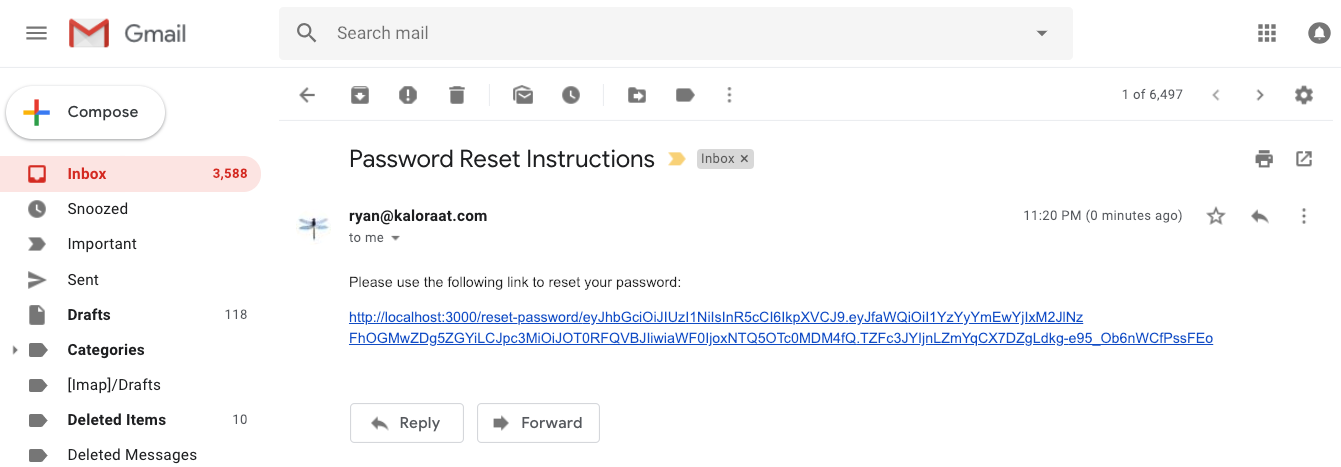
Here is the screenshot:



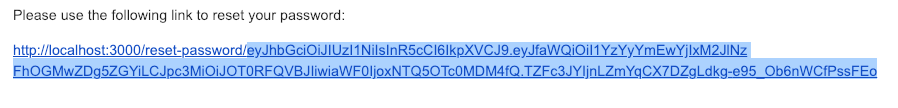
Now check your email. Make sure the user email is valid gmail. If your user is using random gmail for testing purpose such as test1@gmail.com, then you will not get email.

**You should see the following email sent to your user email with clickable password reset link.**





Because at the moment, our react frontend does not have this functionality implemented yet. We cant click on the link because http://localhost:3000/reset-password/tokennnnnnnnnn does not exist. But we can still manually copy the token and try it in postman.



Copy only the token and make another PUT request to backend using the following url and the req body (paste token that you got in the email as value of resetPasswordLink along with the new password).  See the screenshot:

URL: http://localhost:8080/reset-password

body:

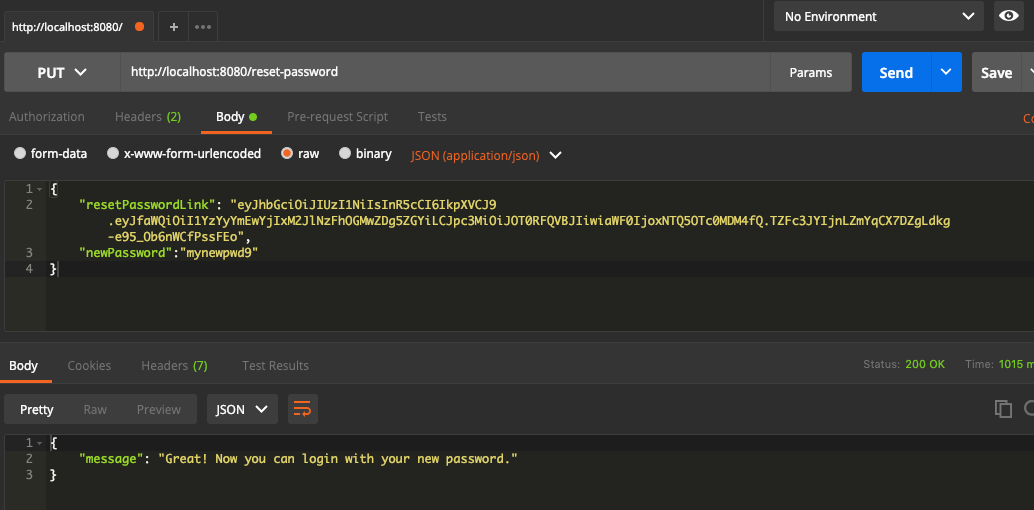
{

"resetPasswordLink": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1YzYyYmEwYjIxM2JlNzFhOGMwZDg5ZGYiLCJpc3MiOiJOT0RFQVBJIiwiaWF0IjoxNTQ5OTc0MDM4fQ.TZFc3JYIjnLZmYqCX7DZgLdkg-e95\_Ob6nWCfPssFEo",

"newPassword":"mynewpwd9"

}

**Here is the screenshot fo successful password reset:**



Now you can try logging that user in with his new password using the React frontend. It works.

In the next lesson, you will learn how to implement it in the react frontend.

Thank You!