Prerequisites

- An already existing app.
- JWT authentication.

Tools

Node and it's package managers

- Postman
- Bcrypts
- JWT
- Nodemailer-express-handlebars
- Nodemailer

Steps

1. A route for the forgot password.
2. Configure nodemailer to send email templates
3. Create email template
4. Method to send password confirmation
5. A route to confirm password reset.
6. Method to generate and token send to the user's email.

1. A route for the forgot password

In the route folder, open the `todoListRoutes.js` file.
Add the snippet below to the file:

```
  app.route('/auth/forgot_password')
    .get(userHandlers.render_forgot_password_template)
    .post(userHandlers.forgot_password);
```

The get route, load the template with which the user needs to interact with and request for reset password.

A post is used to send the username or email address that you want to reset.

The 'forgot_password' handler will listen to post request on '/auth/forgot_password'

2. Configure nodemailer to send email templates

In other words, to send emails with nodemailer, we need to configure it.
For the purpose of this article, we will be using nodemailer Handlebars.js templating engine for convenience.

```
var  hbs = require('nodemailer-express-handlebars'),
  email = process.env.MAILER_EMAIL_ID || 'auth_email_address@gmail.com',
  pass = process.env.MAILER_PASSWORD || 'auth_email_pass'
  nodemailer = require('nodemailer');

var smtpTransport = nodemailer.createTransport({
  service: process.env.MAILER_SERVICE_PROVIDER || 'Gmail',
  auth: {
    user: email,
    pass: pass
  }
});

var handlebarsOptions = {
  viewEngine: 'handlebars',
  viewPath: path.resolve('./api/templates/'),
  extName: '.html'
};

smtpTransport.use('compile', hbs(handlebarsOptions));
```

As you can see, I used the `createTransport` method in Nodemailer to add the service type and authentication params. Check Nodemailer for more.

After this, the template engine and the template location were added to the Nodemailer options.

This helps the Nodemailer to know which engine to use and where to look for the templates.

Note: If you are not using Gmail, change the service option in the nodemailer configuration to your service proovider's name.

3 Email templates

Create a folder called "templates" inside the api folder.
In this template folder, create two HTML files called "forgot-password-email.html" and "reset-password-email.html" respectively.

In the "forgot-password-email", add the code snippet below into it:

```
<!DOCTYPE html>
<html>

<head>
    <title>Forget Password Email</title>
</head>

<body>
    <div>
        <h3>Dear {{name}},</h3>
        <p>You requested for a password reset, kindly use this <a href="{{url}}">link</a> to reset your password</p>
        <br>
        <p>Cheers!</p>
    </div>

</body>

</html>
```

To the "reset-password-email.html" file, the code snippet below.

```
<!DOCTYPE html>
<html>

<head>
    <title>Password Reset</title>
</head>

<body>
    <div>
        <h3>Dear {{name}},</h3>
        <p>Your password has been successful reset, you can now login with your new password.</p>
        <br>
        <div>
            Cheers!
        </div>
    </div>

</body>

</html>
```

4. Method to send password confirmation (forgot_password)

This method will search for the existence of the username/email in the database. If the user exists, a reset password link with a token will be sent to the user's email.

The Async waterfall helps to make sure that each of the functions are performed one after the other (i.e in series often referred to as synchronous).

The first function in the waterfall searches the database for the user existence, if it exist, a token is generated and updates the user object in the database. After this, an email is sent to the user with the token for him/her to use to reset the password.

This serves as a means of confirming that you own the username/email entered.

```
exports.forgot_password = function(req, res) {
  async.waterfall([
    function(done) {
      User.findOne({
        email: req.body.email
      }).exec(function(err, user) {
        if (user) {
          done(err, user);
        } else {
          done('User not found.');
        }
      });
    },
    function(user, done) {
      // create the random token
      crypto.randomBytes(20, function(err, buffer) {
        var token = buffer.toString('hex');
        done(err, user, token);
      });
    },
    function(user, token, done) {
      User.findByIdAndUpdate({ _id: user._id }, { reset_password_token: token, reset_password_expires: Date.now() + 86400000
}, { upsert: true, new: true }).exec(function(err, new_user) {
        done(err, token, new_user);
      });
    },
    function(token, user, done) {
      var data = {
        to: user.email,
        from: email,
        template: 'forgot-password-email',
        subject: 'Password help has arrived!',
        context: {
          url: 'http://localhost:3000/auth/reset_password?token=' + token,
          name: user.fullName.split(' ')[0]
        }
      };

      smtpTransport.sendMail(data, function(err) {
        if (!err) {
          return res.json({ message: 'Kindly check your email for further instructions' });
        } else {
          return done(err);
        }
      });
    }
  ], function(err) {
    return res.status(422).json({ message: err });
  });
};
```

5. A route to confirm password and reset reset.

```
  app.route('/auth/reset_password')
    .get(userHandlers.render_reset_password_template)
    .post(userHandlers.reset_password);
```

When the user who requested the password request clicks the link in the email sent to him/her, the route, '/auth/reset_password' route renders the template with which the user can enter and confirm the new password.

Behind this, is an AJAX request to the back-end on the post route to save the new password.

6. Method to save the new password.

To save the password, there are basic things that need to be done first.

Check if the token exists in the database and has not expired.
Check to see that the password entered is correctly typed by comparing the two typed passwords

```
exports.reset_password = function(req, res, next) {
  User.findOne({
    reset_password_token: req.body.token,
    reset_password_expires: {
      $gt: Date.now()
    }
  }).exec(function(err, user) {
    if (!err && user) {
      if (req.body.newPassword === req.body.verifyPassword) {
        user.hash_password = bcrypt.hashSync(req.body.newPassword, 10);
        user.reset_password_token = undefined;
        user.reset_password_expires = undefined;
        user.save(function(err) {
          if (err) {
            return res.status(422).send({
              message: err
            });
          } else {
            var data = {
              to: user.email,
              from: email,
              template: 'reset-password-email',
              subject: 'Password Reset Confirmation',
              context: {
                name: user.fullName.split(' ')[0]
              }
            };

            smtpTransport.sendMail(data, function(err) {
              if (!err) {
                return res.json({ message: 'Password reset' });
              } else {
                return done(err);
              }
            });
          }
        });
      } else {
        return res.status(422).send({
          message: 'Passwords do not match'
        });
      }
    } else {
      return res.status(400).send({
        message: 'Password reset token is invalid or has expired.'
      });
    }
  });
};
```

From the code snippet above, if the token exists in the database, has not expired, and the two passwords match, the new password will be saved in the database.

After this, an email is sent to the user on successfully completing his/her password reset.

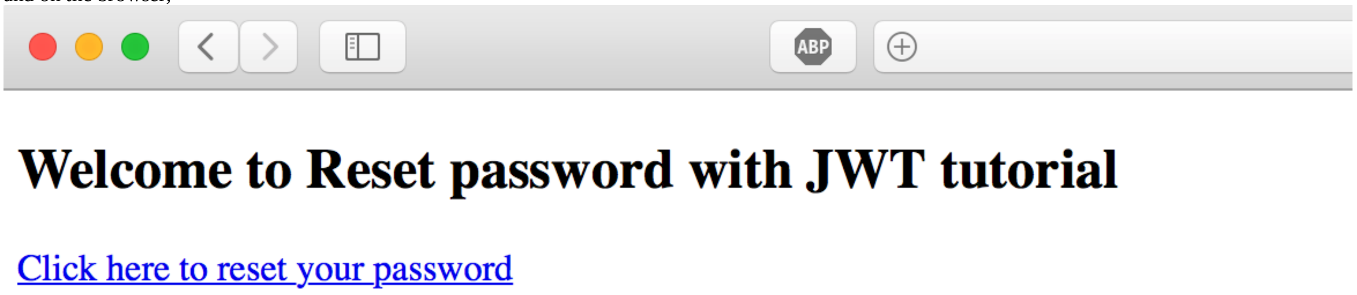
Testing

After cloning and installation of the packages, run "npm run start"

On your browser, go to "localhost:3000

If you've got everything right, you should see

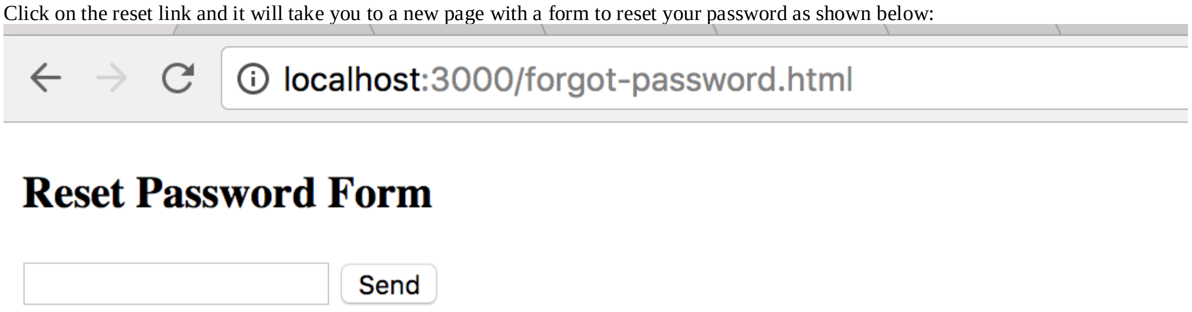
and on the browser,



Welcome to Reset password with JWT tutorial

Click here to reset your password

Click on the reset link and it will take you to a new page with a form to reset your password as shown below:



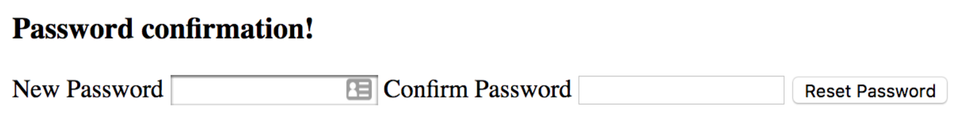
localhost:3000/forgot-password.html

Reset Password Form

Enter the email you want the password to be reset and click send.

If the email is found in the database, an email will be sent to the email address entered.

Note: Don't forget to add email credentials to the nodemail configurations.
Click the link in the email. The link should will take open a new browser tab and you will be prompted to enter your new password with confirmation.



After this, click send. If the token generated has not expired, the email password will be reset to the password and you can login.