Async Await + Error Handling

Introduction

We will be reducing the complexity of the code that we have built so far in terms of database queries. We will also be creating notifications for various tasks visible to users { Flash message }.

Converting to Async Await

Asynchronous - When the task is running on the CPU at the same time the main thread of the node.js server is executing something else. Asynchronous code doesn't have to wait – the program can continue to run. We can do this to keep the site or app responsive, reducing waiting time for the user

- We will make the { **home_controller.js** } page clutter-free using either **promises** or **async-await**.
- Async await tells the server that the function contains some asynchronous statements and we need to wait with each async statement. Once it gets executed then and only then moves on to the next statement.

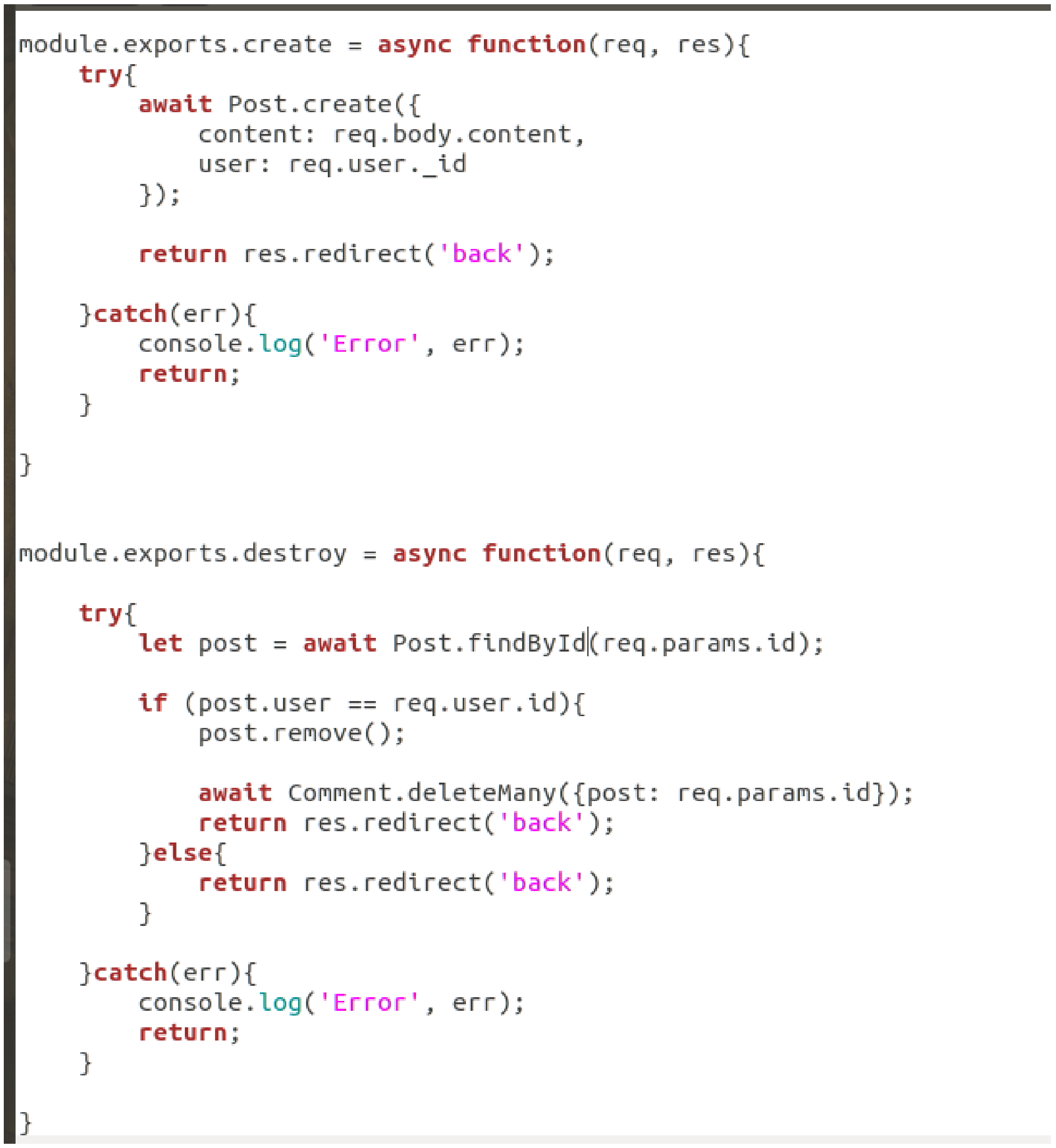
{ **home_controller.js** }

```
const Post = require('../models/post');
const User = require('../models/user');



module.exports.home = async function(req, res){

    try{
         // populate the user of each post
        let posts = await Post.find({})
        .populate('user')
        .populate({
            path: 'comments',
            populate: {
                path: 'user'
            }
        });

        let users = await User.find({});

        return res.render('home', {
            title: "Codeial | Home",
            posts:  posts,
            all_users: users
        });

    }catch(err){
        console.log('Error', err);
        return;
    }

}
```

Converting More Code to Async Await

- It is not always necessary to change every part of the async code to async-await. Although it is a good practice to follow the same convention everywhere.
- We will convert **{ posts_controller.js }** and **{ comments_controller.js }** files into Async Await.

{ posts_controller.js }



```
module.exports.create = async function(req, res){
    try{
        await Post.create({
            content: req.body.content,
            user: req.user._id
        });

        return res.redirect('back');

    }catch(err){
        console.log('Error', err);
        return;
    }

}


module.exports.destroy = async function(req, res){

    try{
        let post = await Post.findById(req.params.id);

        if (post.user == req.user.id){
            post.remove();

            await Comment.deleteMany({post: req.params.id});
            return res.redirect('back');
        }else{
            return res.redirect('back');
        }

    }catch(err){
        console.log('Error', err);
        return;
    }

}
```

{ comments_controller.js }

```
module.exports.create = async function(req, res){
    try{
        let post = await Post.findById(req.body.post);
        if (post){
            let comment = await Comment.create({
                content: req.body.content,
                post: req.body.post,
                user: req.user._id
            });
            post.comments.push(comment);
            post.save();

            res.redirect('/');
        }
    }catch(err){
        console.log('Error', err);
        return;
    }
}
module.exports.destroy = async function(req, res){

    try{
        let comment = await Comment.findById(req.params.id);
        if (comment.user == req.user.id){
            let postId = comment.post;
            comment.remove();
            let post = Post.findByIdAndUpdate(postId, { $pull: {comments: req.params.id}});

            return res.redirect('back');
        }else{
            return res.redirect('back');
        }
    }catch(err){
        console.log('Error', err);
        return;
    }
}
```

Flash Messages:: Introduction

Flash messages are the ones that get displayed on your website. We require that the users should be given a response in terms of notification whenever they are performing actions on the website, and this is where Flash messages come in.

Creating Flash Messages

- Flash messages are stored in the session cookies and they are cleared on the next request. Hence, whenever we sign in, the flash message is sent into the session cookie, and whenever we refresh that flash message is erased.
- We will be using the **{ Connect Flash library }** for creating flash messages.
- To install the **Connect Flash library** use the command **{ npm install connect-flash }** in the terminal.
- We have to require the library inside the { **index.js** } file, which is the entry point for the application.
- Using the app. use method, we will enable the connect-flash package. This comes right after the use of sessions inside the **{ index.js }** file.
- Inside the **{ users_controller.js }** file we will create two flash messages — one for sign-in and one for sign-out.

{ users_controller.js }

```
// sign in and create a session for the user
module.exports.createSession = function(req, res){
    req.flash('success', 'Logged in Successfully');
    return res.redirect('/');
}

module.exports.destroySession = function(req, res){
    req.logout();
    req.flash('success', 'You have logged out!');
    return res.redirect('/');
}
```

- We need to send the flash message to the response. For that, we will create a new file **{ middleware.js }** inside the config folder.

{ middleware.js }

```
module.exports.setFlash = function(req, res, next){
    res.locals.flash = {
        'success': req.flash('success'),
        'error': req.flash('error')
    }

    next();
}
```

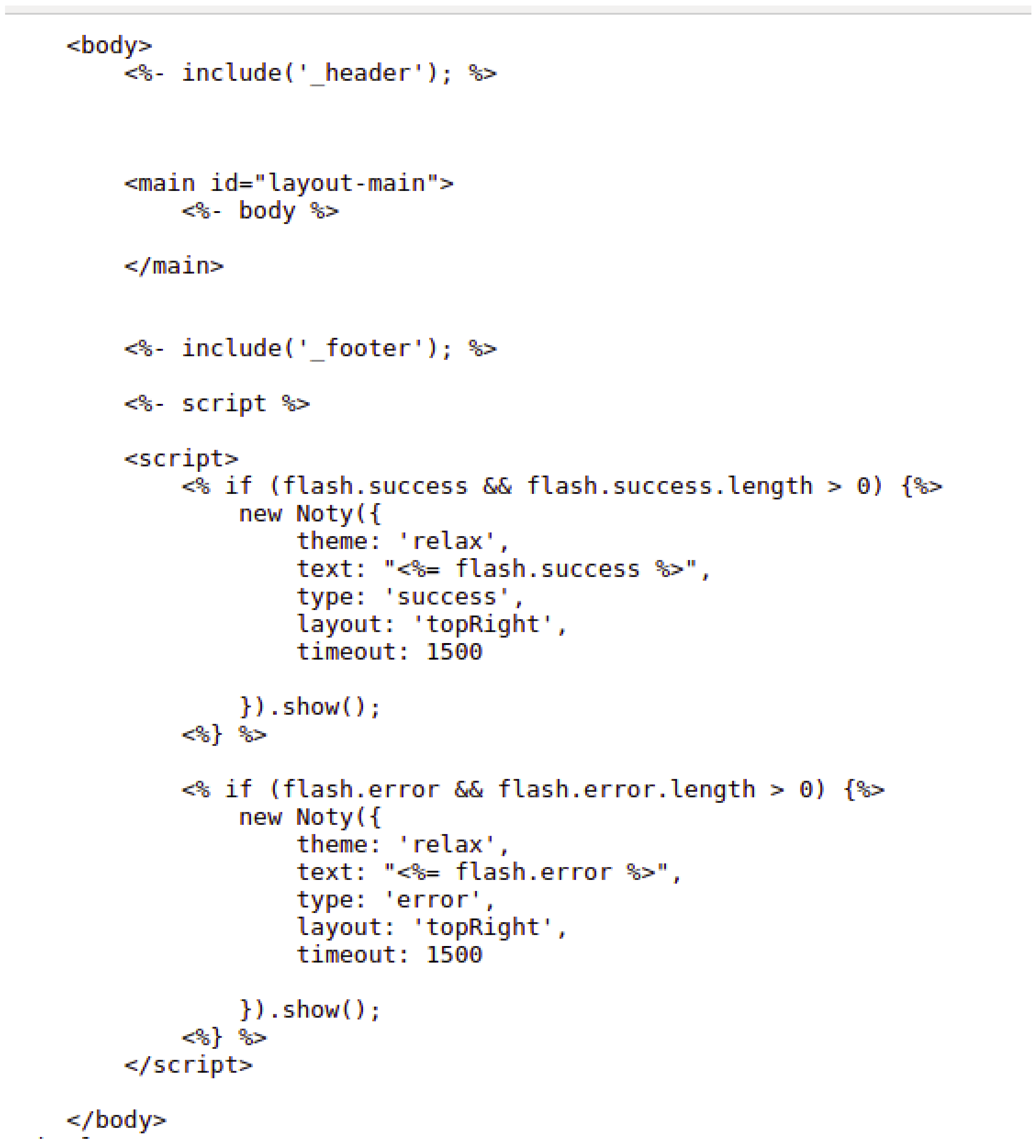
- ***<u>EXTRA</u>*: For further read, you can refer to the link provided below.**

  [***https://www.npmjs.com/package/connect-flash***](https://www.npmjs.com/package/connect-flash)

Introducing Noty

- We will make the notification look fancy, a little more animated using a library known as { **Noty.js** }.
- To include Noty.js in the project we have to include following links -
- Include the CSS link in the { **layout.ejs** }file { <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css" integrity="sha512-NXUhxhkDgZYOMjaIgd89zF2w51Mub53Ru3zCNp5LTlEzMbNNAjTjDbpURYGS5Mop2cU4b7re1nOIucsVlrx9fA==" crossorigin="anonymous" /> }.
- Include the script tag in the { **layout.ejs** } file { <script src="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.min.js" integrity="sha512-lOrm9FgT1LKOJRUXF3tp6QaMorJftUjowOWiDcG5GFZ/q7ukof19V0HKx/GWzXCdt9zYju3/KhBNdCLzK8b90Q==" crossorigin="anonymous"></script> **}.**

{ layout.js }



```
<body>
    <%- include('_header'); %>


    <main id="layout-main">
        <%- body %>

    </main>


    <%- include('_footer'); %>

    <%- script %>

    <script>
        <% if (flash.success && flash.success.length > 0) {%>
            new Noty({
                theme: 'relax',
                text: "<%= flash.success %>",
                type: 'success',
                layout: 'topRight',
                timeout: 1500

            }).show();
        <%} %>

        <% if (flash.error && flash.error.length > 0) {%>
            new Noty({
                theme: 'relax',
                text: "<%= flash.error %>",
                type: 'error',
                layout: 'topRight',
                timeout: 1500

            }).show();
        <%} %>
    </script>

</body>
```

<u>EXTRA</u>:   For further read, you can refer to the link provided below.

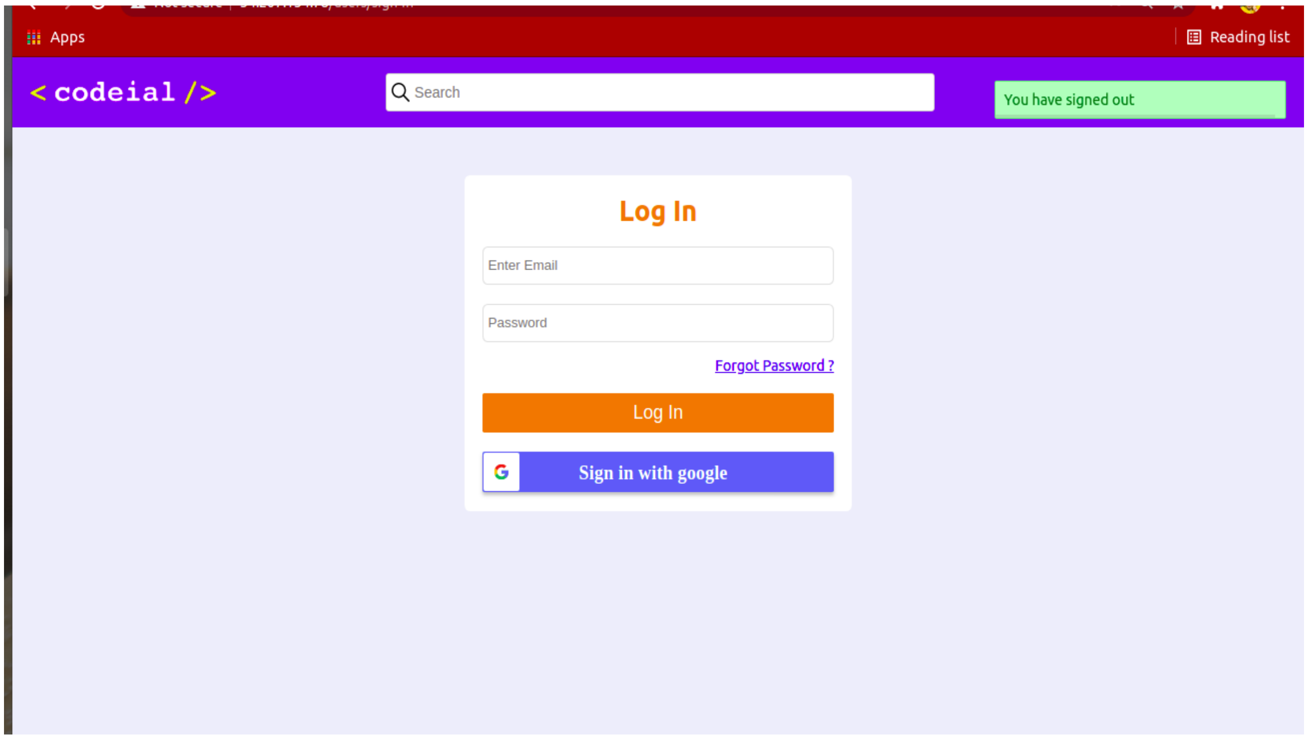
https://ned.im/noty/#/

Adding Flash Messages to more Actions

***<u>EXTRA</u>* - As a mini assignment try showing flash messages whenever the users sign in, sign-out, new sign, new comment creation, and comment deletion message.**

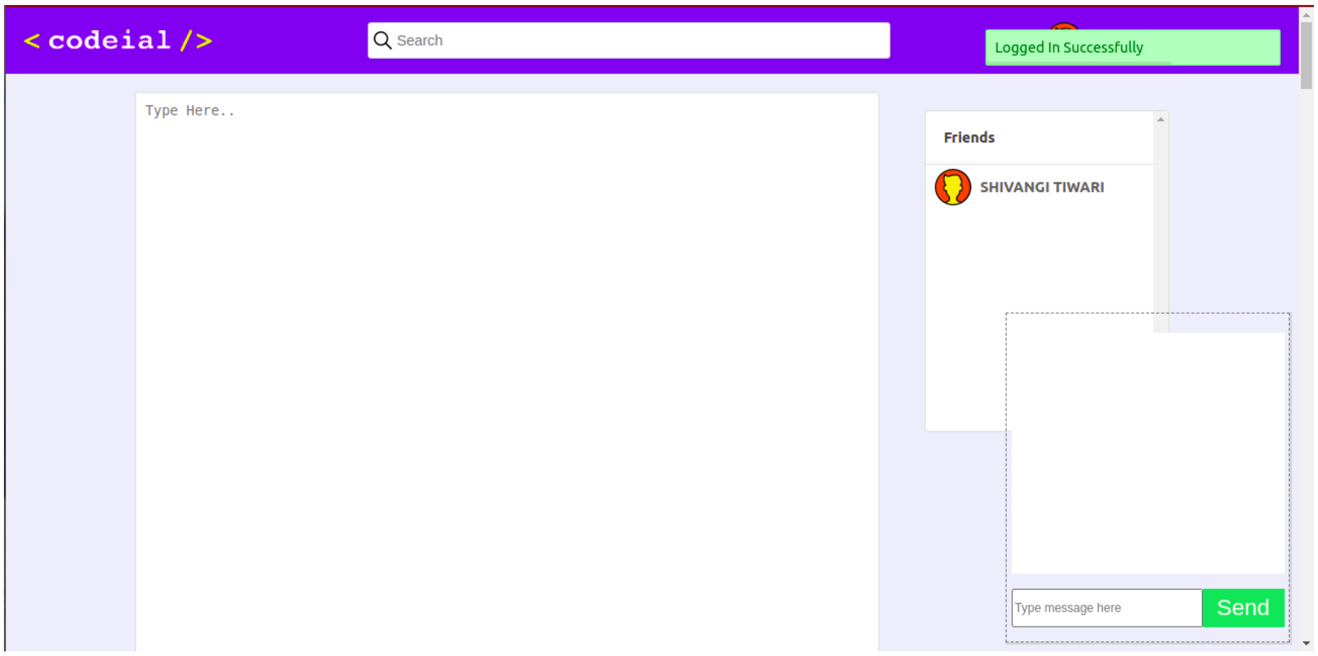
You check the page for your reference.

{ Signed out successfully }



You have signed out

{ Signed in successfully }



Summarizing

- To make the code look clean and modular, we used async-await.
- We have used Noty.js to show notifications in a fancy way.
- Notifications are shown using flash messages.
- We use flash messages which are set up in session cookies.
- We use the Connect Flash library to set up flash messages in session cookies.