Exercise (Instructions): Cross-Origin Resource Sharing

Objectives and Outcomes

In this exercise you will learn about CORS and how you can configure your Express application to support CORS using the cors Node module. At the end of this exercise you will be able to:

- Install and configure the cors Node module
- Configure your Express application to use the cors module to support CORS on various endpoints

Installing cors Module

- To install the cors module, type the following at the prompt:

```
npm install cors@2.8.4 --save
```

Configuring the Server for CORS

- In the routes folder, add a new file named *cors.js* and add the following code to it:

```
const express = require('express');
const cors = require('cors');
const app = express();

const whitelist = ['http://localhost:3000', 'https://localhost:3443'];
var corsOptionsDelegate = (req, callback) => {
    var corsOptions;
    console.log(req.header('Origin'));
    if(whitelist.indexOf(req.header('Origin')) !== -1) {
        corsOptions = { origin: true };
    }
    else {
        corsOptions = { origin: false };
    }
    callback(null, corsOptions);
};

exports.cors = cors();
exports.corsWithOptions = cors(corsOptionsDelegate);
```

- Then, open *dishRouter.js* and update it as follows:

```
. . .

const cors = require('./cors');

. . .

dishRouter.route('/')
.options(cors.corsWithOptions, (req, res) => { res.sendStatus(200); })
.get(cors.cors, (req,res,next) => {

. . .

.post(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
  (req, res, next) => {

. . .

.put(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
  (req, res, next) => {


. . .

.delete(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
  (req, res, next) => {


. . .

dishRouter.route('/:dishId')
.options(cors.corsWithOptions, (req, res) => { res.sendStatus(200); })
.get(cors.cors, (req,res,next) => {

. . .

.post(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
  (req, res, next) => {

. . .

.put(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
  (req, res, next) => {


. . .

.delete(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
  (req, res, next) => {


. . .

dishRouter.route('/:dishId/comments')
.options(cors.corsWithOptions, (req, res) => { res.sendStatus(200); })
.get(cors.cors, (req,res,next) => {

. . .

.post(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
  (req, res, next) => {

. . .

.put(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
  (req, res, next) => {


. . .

.delete(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
  (req, res, next) => {


. . .

dishRouter.route('/:dishId/comments/:commentId')
.options(cors.corsWithOptions, (req, res) => { res.sendStatus(200); })
.get(cors.cors, (req,res,next) => {

. . .

.post(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
  (req, res, next) => {

. . .

```

```
77  .put(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
      (req, res, next) => {
78
79
80  . . .
81
82  .delete(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin,
      (req, res, next) => {
83
84
```

- Do similar updates to promoRouter.js, leaderRouter.js, uploadRouter.js and users.js.
- Save all the changes and test your server.
- Do a Git commit with the message "Cors".

Conclusions

In this exercise you learnt to configure your Express server to support CORS with the cors Node module.