```
var express = require('express');
var router = express.Router();

/* */
router.get('/', function(req, res, next) {
  res.send('respond with a resource');
});


/**
 * topics that will be tested:
 *      - writing nodeJS routes
 *      - callbacks
 *      - promises
 *      - Ajax
 *      - socket.io
 *
 */



/**
 * Exercise 1
 * - example of: writing nodeJS routes. The route takes some parameters and combines them. We
had a similar
 *              exercise in the lab with two numbers being added. The following is another
example
 * Create a nodejs POST route that takes four strings as input and returns the concatenation
of the four strings
 * e.g. input -> {val1: 'one', val2: 'two', val3: 'three', val4: 'four'}
 * output -> 'one - two - three - four'
 * e.g. your answer should look like:
 * router.post('/concatenate', function (req, res) {...}
 */

/**
 * solution:
 */
router.post('/concatenate', function (req, res, next) {
  const val1 = req.body.val1;
  const val2 = req.body.val2;
  const val3 = req.body.val3;
  const val4 = req.body.val4;
  res.setHeader('Content-Type', 'application/json');
  res.json(val1 + ' - ' + val2 + ' - ' + val3 + ' - ' + val4);
});


// +------------------------------------------------ +
/**
 * Exercise 2
 * - example of: Ajax exercise
 *     - write the JQuery Ajax function that communicates with the following server route
 *     your solution should take the form of e.g.
 *     function sendAjaxQuery(url, data) { ...}
 *
 *     the Ajax call should
 *         - print on the console the values returned if the call is successful
 *         - create an alert showing the error otherwise
 */

router.post('/user_data', function(req, res, next) {
    let userData = req.body;
    const currentYear = (new Date()).getFullYear();
    const dob= parseInt(userData.year);
    if (userData == null) {
        res.setHeader('Content-Type', 'application/json');
        res.status(403).json({error: 403, reason: 'no user data provided'});
    } else if (!isNumeric(userData.year) || dob>currentYear) {
        res.setHeader('Content-Type', 'application/json');
        res.status(403).json({error: 403, reason: 'Year is invalid'});
    } else if (!userData.firstname) {
            res.setHeader('Content-Type', 'application/json');
            res.status(403).json({error: 403, reason: 'First name is invalid'});
    } else if (!userData.lastname) {
        res.setHeader('Content-Type', 'application/json');
        res.status(403).json({error: 403, reason: 'Last name is invalid'});
    }
    else {
        userData.age = currentYear - dob;
        res.setHeader('Content-Type', 'application/json');
        res.json(userData);
    }
});

/**
 * solution:
 *
 */
function sendAjaxQuery(first, last, year) {
    $.ajax({
        url: '/user_data' ,
        data: JSON.stringify({firstname: first, lastname: last, year: year}),
        contentType: 'application/json',
        dataType: 'json',
        type: 'POST',
        success: function (dataR) {
            console.log(JSON.stringify(dataR));
        },
        error: function (response) {
            // the error structure we passed is in the field responseText
            // it is a string, even if we returned as JSON
            // if you want o unpack it you must do:
            // const dataR= JSON.parse(response.responseText)
            alert (response.responseText);
        }
    });
}

/**
 * +-----------------------------------------------------+
 */

/**
 * Exercise 3
 * - example of: Ajax exercise
 * Write the route the following Ajax call points to.
 * Make sure to check for errors in the input
 */
function sendAjaxQuery_1(first, last, year) {
    $.ajax({
        url: '/person_values' ,
        data: JSON.stringify({firstname: first, lastname: last, year: year}),
        contentType: 'application/json',
        dataType: 'json',
        type: 'POST',
        success: function (dataR) {
            // no need to JSON parse the result, as we are using
            // dataType:json, so JQuery knows it and unpacks the
            // object for us before returning it
            // in order to have the object printed by alert
            // we need to JSON.stringify the object
            console.log(JSON.stringify(dataR));
        },
        error: function (response) {
            // the error structure we passed is in the field responseText
            // it is a string, even if we returned as JSON
            // if you want o unpack it you must do:
            // const dataR= JSON.parse(response.responseText)
            alert (response.responseText);
        }
    });
}

/**
 * solution
 */
router.post('/user_data', function(req, res, next) {
    let userData = req.body;
    const currentYear = (new Date()).getFullYear();
    const dob= parseInt(userData.year);

    if (userData == null) {
        res.setHeader('Content-Type', 'application/json');
        res.status(403).json({error: 403, reason: 'no user data provided'});
    } else if (!isNumeric(userData.year) || dob>currentYear) {
        res.setHeader('Content-Type', 'application/json');
        res.status(403).json({error: 403, reason: 'Year is invalid'});
    } else if (!userData.firstname) {
        res.setHeader('Content-Type', 'application/json');
        res.status(403).json({error: 403, reason: 'First name is invalid'});
    } else if (!userData.lastname) {
        res.setHeader('Content-Type', 'application/json');
        res.status(403).json({error: 403, reason: 'Last name is invalid'});
    }
    userData.age = currentYear - dob;
    res.setHeader('Content-Type', 'application/json');
    res.json(userData);
});

// -- note:
// * not checking for errors -10% each
// * not returning an error code (whatever returned is fine as long as it is not 200-299)
means -10% each
// * potential last line could be either res.json (userData)  or
res.send(JSON.stringify(userData))
// * not returning JSON or stringified data means -30%

/**
 * +-----------------------------------------------------+
 */

/**
 * Exercise 4
 * Suppose you have a client side of a socket.io programme that does the following:
 */
function clientSide4(socket, room, userId) {
    socket.emit('hello', room, userId);
    // ....
    socket.on('new_member', function(room, userId){
        console.log('User '+userId+'  just joined the room '+room);
    })
}

/**
 * Exercise 5
 * write the server side of socket.io that responds to a client message:
```

```
 *              socket.emit('hello', room, userId);
 * the server should:
 *     - insert the requester into the requested room
 *     - send everyone in the room (including the requester) a message 'new_member'
 * your solution should take the form of, e.g.
 * function initSocket (io) {
     io.sockets.on('connection', function (socket) {
     // here write the code to join the room and send the message 'new_member' to everyone)
     }
 */
/**
 * ------SOLUTION ----------
 */
function initSocket_4 (io) {
    io.sockets.on('connection', function (socket) {
        try {
            socket.on('hello', function (room, userId) {
                          socket.join(room);
                          socket.broadcast.to(room).emit('new_member', room, userId);
            });
            //...
        } catch (error) {
            console.log(error);
        }
    });
}

/**
 * Exercise 6
 * topic tested: creating a constellation of servers
 *  write a node.js server route sending data to the following route.
 * Your solution should take the form of:
 * router.post('/connect, function(req, res){
 *              // here connect to the route  http://localhost:3001/connect_to_other
 *              }
 *  the route must return the json data to the caller  both in case of error and and in case
of success
 */
router.post('/connect_to_other', function (req, res, next) {
    const userData = req.body;
    const age= userData.age;
    const name= userData.name;
    const surname= userData.surname;
    if (!userData || !age || !name || !surname)
        res.status(504).send('wrong data in input');
    else {
        res.setHeader('Content-Type', 'application/json');
        res.json(userData);
    }
});

/**
 * Solution:
 */

router.route('/exercise4')
    .get (function(req, res) {
        res.render('index', {title: 'Express'});
    })

    .post(function  (req, res) {
        const userData = req.body;

        fetch('http://localhost:3000/add', {
            method: 'post',
            body: JSON.stringify(userData),
            headers: {'Content-Type': 'application/json'},
        })
            .then (res => res.json())
            .then(json =>
                res.render('index', {title: " results is: "+json.result}))
            .catch(err =>
                res.render('index', {title: err}))
    });

// Note: -20% for not checking the error

/**
 * Exercise 7
 * topic tested: understanding callbacks
  * look at the sequence of callbacks. What is the output on the console in case ?
 * A: X, Q, Z
 * B: X, Y, Z
 * C: X, Z, Y
 * D: X, Y, Q
  */
 router.post('/whatever', function (req, res, next) {
     console.log("X");
     asyncOperationWithCallback(body.value, function (err) {
         if (err) {
             console.log("Y");
             res.render('index', {title: err})
         } else {
             console.log("Q");
             res.render('index', {title: "What?"})
         }
     });
     console.log("Z");
 });
/**
 * solution: C: X, Z, Y
 */

module.exports = router;
```