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

/* GET home page. */
router.get('/', function (req, res) {
    res.render('index', {title: 'Express'});
});



//----------------------------------------------------------

/**
 * Exercise 1
 * create a nodejs POST route that takes two integers as input and returns the sum of their
squares (in Javascript use the  Math.pow function)
 * e.g. input -> 4, 5
 * output -> 41  (that is because 4^2=16, 5^2=25 and 16+25=41)
 * Your answer should look like:
 * router.post('/power', function (req, res) {...});
 */
/**
 * Solution
 */
router.post('/power', function (req, res) {
    const val1 = req.body.val1;
    const val2 = req.body.val2;
    const valx1= Math.pow(parseInt(val1),2);
    const valx2= Math.pow(parseInt(val2),2);
    res.setHeader('Content-Type', 'application/json');
    res.json(valx1+ valx2);
});

//+----+

/**
 * Exercise 2
 * create a nodejs POST route that takes two integers as input and returns the multiplication
of their cubed values
 * (in Javascript use the  Math.pow function)
 * e.g. input -> 4, 5
 * output -> 8,000       (that is because 4^3 =64, 5^3=125 and  64*125=8000)
 * Your answer should look like:
 * router.post('/power', function (req, res) {...});
 */

/**
 * Solution
 */
router.post('/power', function (req, res) {
    const val1 = req.body.val1;
    const val2 = req.body.val2;
    const valx1= Math.pow(parseInt(val1),3);
    const valx2= Math.pow(parseInt(val2),3);
    res.setHeader('Content-Type', 'application/json');
    res.json(valx1 * valx2);
});



//----------------------------------------------------------
/**
 * Exercise 3
 * write an Ajax function sending data to this route
 * e.g.  $.ajax({...});
 */
router.post('/connect_from_ajax', 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
 */

function exercise3 (player, position, age){
    $.ajax({
        url: '/user_data' ,
        data: JSON.stringify({player: player, position: position, age: age}),
        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 4
 * write an Ajax function sending data to this route
 * e.g.  $.ajax({...});
 * the success function must print out the data returned
 * the error function should alert and return any error received
 */
router.post('/connect_from_ajax', function (req, res, next) {
    const userData = req.body;
    const weight = req.body.weight;
    const speed = req.body.speed;
    const height = req.body.height;
    if (!weight || !speed || !height)
        res.status(504).send('wrong data in input');
    else {
        res.setHeader('Content-Type', 'application/json');
        res.json(userData);
    }
});

/**
 * Solution
 *
 */
function exercise4 (weight, speed, height){
    function sendAjaxQuery(weight, speed, height) {
        $.ajax({
            url: '/connect_from_ajax' ,
            data: JSON.stringify({weight: weight, speed: speed, height: height}),
            contentType: 'application/json',
            dataType: 'json',
            type: 'POST',
            success: function (dataR) {
                alert(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)
                console.log (response.responseText);
            }
        });
    }
}

//----------------------------------------------------------
/**
 * Exercise 5
 * print out the correct sequence of letters printed out by console.log in case
 * 1. the fetch returns an error
 * 2. the fetch returns no error
 */
router.post('/callback5', function (req, res, next) {
    console.log("D");
    fetch('http://localhost:3000/other_server', {
        method: 'post',
        body: "{}",
        headers: {'Content-Type': 'application/json'},
    })
        .then(res => res.json())
        .then(json => {
            console.log("A");
            }
        )
        .catch(err => {
            console.log("C");
            res.render('index', {title: err})
        })
        .finally(() => {
            console.log("F");
            res.render('index', {title: "finished"})
        })
    console.log("E");
});

/**
 * solution:
 *    with errors: D, E, C, F
 *    no errors: D, E, A, F
 */



// +-----+
/**
 * Exercise 6
 * print out the correct sequence of letters printed out by console.log in case
 * 1. the fs.writFile calls return  errors
 * 2. the fs.writFile return no errors
 */
router.post('/callback6', function (req, res, next) {
    console.log("A");
    fetch('http://localhost:3000/other_server', {
        method: 'post',
        body: "{}",
        headers: {'Content-Type': 'application/json'},
    })
        .then(res => res.json())
        .then(json => {
                console.log("B");
            }
        )
        .catch(err => {
            console.log("C");
            res.render('index', {title: err})
        })
        .finally(() => {
            console.log("D");
            res.render('index', {title: "finished"})
        })
    console.log("E");
});

/**
 * 1. A, E, C, D
 * 2. A, E, B, D
 */

//----------------------------------------------------------
/**
 * Exercise 7
 * Suppose you have a socket io server with 3 users connected via socket.io
 * user 1 is in room 1
 * user 2 is in room 2
 * user 3 is in room 3
 * suppose user 4 joins room 1. Who will receive the notification that user 4 has joined room
1?
 * A. everyone including user 4
 * B. everyone excluding user 4
 * C. user 1 and user 4
 * D. user 1 only
 */
/**
 * server side
 */
function initSocket (io) {
    io.sockets.on('connection', function (socket) {
        try {
            socket.on('join room', function (room, userId) {
                          socket.join(room);
                          socket.broadcast.to(room).emit('joined', room, userId);
            });
            //...
        } catch (error) {
            console.log(error);
        }
    });
}

/**
 * solution: D as only users 1 and 4 re in room 1 and using socket.broadcast.emit, the
message goes to all the users but the sender
 *
 */
/**
 * client side.
 */
function clientSide1(socket, room, userId) {
    socket.emit('join room', room, userId);
    // ....
    // will print out the message 'User Id userId123 has joined room AAA?
    // A. userId
    socket.on('joined', function(room, userId){
        console.log('User Id '+userId+' has joined room '+room);
    })
}

//+-----+

/**
 * Exercise 8
 * Suppose you have a socket io server with 3 users connected via socket.io
 * user 1 is in room 1
 * user 2 is in room 2
 * user 3 is in room 3
 * suppose user 4 joins room 1. Who will receive the notification that user 4 has joined room
1?
 * A. everyone including user 4
 * B. everyone excluding user 4
 * C. user 1 and user 4
 * D. user 1 only
 */
/**
 * server side
 */
function initSocket (io) {
    io.sockets.on('connection', function (socket) {
        try {
            socket.on('join room', function (room, userId) {
                            socket.join(room);
                            socket.to(room).emit('joined', room, userId);
            });
            //...
        } catch (error) {
            console.log(error);
        }
    });
}

/**
 * solution: C. Using socket.emit means all users in room 1 (i.e. users 1 and 4) receive the
message, including the sender
 */

/**
 * client side.
 */
function clientSide2(socket, room, userId) {
    socket.emit('join room', room, userId);
    // ....
    // will print out the message 'User Id userId123 has joined room AAA?
    // A. userId
    socket.on('joined', function(room, userId){
        console.log('User Id '+userId+' has joined room '+room);
    })
}

module.exports = router;
```