## Lab 9

## GET, POST and DELETE REST API endpoints for student JSON data

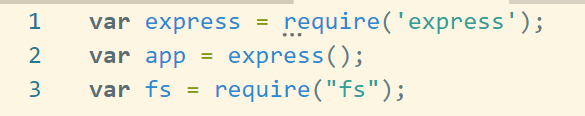
Start node.js.

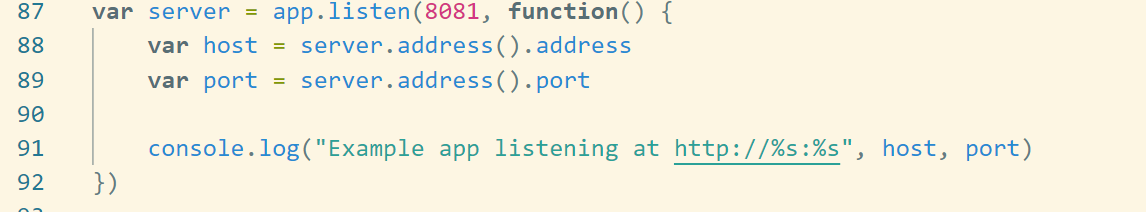
Use VSCode as your editor.

* Save all your answers in a file named **YourFirstNameYourLastName**.doc (or the appropriate file extension). Upload this file to Blackboard.

Make sure your server from Lab 8 is working

Include the file server! You haven’t got the file server in Lab 8.





***1 How can you test if the server is working in Chrome?***

***2 How can you test if the server is working in Insomnia?***

Creating a JSON file with student data

Here is a JSON file consisting of data about 2 users:

[

{"name":"jane",

"password":"password1",

"profession":"teacher"},

{"name":"sinead",

"password":"password2",

"profession":"clerk"}

]

Create a JSON file called students.json consisting of 3 students. You want to include data for the following:

id

fname

lname

course

year

***3 What is the contents of your students.json file?***

Listing the students

The command to send a response string called data to the client is:

res.send(data);

We had called the strings we read from a JSON file jsonString. We are now calling the string data.

***4 Write the code to display the contents of the students.json file on the client and on the console. You can refer to Lab 6 and Lab 8.***

***5 Test out your code in Chrome and also in Insomnia. How did you do this?***

Viewing student details based on an id, id is hardcoded

So as to access particular elements of an array, an array is made equal to the parsed data. The array consists of a number of student objects. The various properties for each object in the array can now be accessed. This example is for userArray. Here the location property for the object located at index 0 is being found, turned into a string and sent as a response to the client.

        const userArray = JSON.parse(data);

        res.send(JSON.stringify(userArray[0].location));

***6 Send a response to the client as follows:***

***The 2nd students last name is xxx***

Viewing all the student lastnames

Using another GET method, create another endpoint called lastnames.

As before, read the students file and in the callback function, make studentArray equal to the parsed contents of the file.

This time, iterate through the array so that you are making a concatenated string of all the lastNames of the students. You can also number the students.

Send the string as a response to the client.

***7 Insert the code here for viewing all the student lastnames.***

Viewing student details based on the id, id is in the query string

The following code allows you to view the details of a user based on the user id. Now do the same for viewing the details of a student based on their id.

You will have to alter the code so it is applicable to view a student instead of a user.

app.get('/:id', **function**(req, res) {

// Insert code here to read the JSON file

// The following code is part of the callback function

// It will send a response consisting of the location of a user

**const** userArray = JSON.parse(data);

      res.send(JSON.stringify(userArray[req.params.id].location));

})

***8 Insert a copy of your code for viewing student details based on their id here.***

Creating a student object

Create a student object:

**var** student = {

property values go here similar to the “name” : “value” pairs Separated by a ,

}

Adding a student

The following is the code for adding a user. Adapt the code so that it will add a student.

Use the GET method for now.

For the callback function:

Declare an array and make it equal to the parsed JSON data.

Make a new object at the end of the array and make it equal to the student object.

Send the stringified array as a response to the client.

You can then modify the following code to write the new data to the file.

        fs.writeFile("./JSONfiles/user.json", JSON.stringify(userArray), err **=>** {

            if (err) console.log("Error writing file:", err);

        })

***9***  ***Insert a copy of your code for adding a student here.***

***10***  ***Copy your code for adding a student. Modify the copy so that it is now using a POST method. Using Insomnia, do a post so as to add a student. How did you do this? What appeared in Insomnia? Take a look at the students.json file. Has it changed?***

Deleting a student

Use the DELETE method.

app.delete('/deleteUser', **function**(req, res) {

        data = JSON.parse(data);

        delete data[2];

***11*** ***Using Insomnia, delete a student and then add a student. Insert a screenshot of your code for deleting a student.***