**Lab2 - HTTP server**

1. **Provide Server to List Employees**

* Use a filesystem (Use the **stream** module).
* Both Lab01, Lab02 should use the same JSON file (used as database).
* You should have 4 pages
  + **Page One: Home page localhost:3000/** it’s what the users see when they first land on our website. It'll show our employees list that the user entered from yesterday (just Viewing). (don’t show the id)
  + **Page Two: Astronomy page: localhost:3000/astronomy** 
    - Should display this [image](https://media.cnn.com/api/v1/images/stellar/prod/200505225212-04-fossils-and-climate-change-museum.jpg?q=x_0,y_0,h_1125,w_1999,c_fill/h_720,w_1280) along with a few lines of description.
  + **Page Three: Serbal page: localhost:3000/serbal** 
    - Should display this [image](https://i.imgur.com/DCnpfBy.jpeg) along with a few lines of description.
  + **Page four: 404 page not found (just simple html)**
  + **Add an employee: localhost:3000/employee (using curl or postman)**
* **Style** your page, serve the css file **separately**, from disk.
* Use read **streams**.

**Bonus:**

* Add a new route to download this [image](https://media.cnn.com/api/v1/images/stellar/prod/200505225212-04-fossils-and-climate-change-museum.jpg?q=x_0,y_0,h_1125,w_1999,c_fill/h_720,w_1280). (trigger the download dialog) **localhost:3000/astronomy/download**

**Hint**: look-up for the correct header ;)

* **Page Five: Add an employee, localhost:3000/employee**

**Notes:**

* **Make sure to try the debugging environment**
* **Make sure that you enable restart on change using nodemon**
* **Make sure to split your logic into modules**
* Please make sure that you separate your logic in functions
* Make the names of your functions and variables are expressive of what they are
* Start by defining each functionality and the steps you should make to achieve it
* Please make sure not to pollute the global scope[refrain from using global variables]
* Before writing a function, ask yourself, is there a function in javascript that can do that, if not, write your own.

**Tips:**

* If you can’t do something, search, then ask for help, this is not an exam.
* Please please please, don’t copy someone else’s code.

**Useful Links:**

* [npm | build amazing things](https://www.npmjs.com/)
* Nodejs [Docs](https://nodejs.org/en/docs/)
* Fs module docs [fs](https://nodejs.org/dist/latest-v12.x/docs/api/fs.html) (you will need to use the sync version of the functions)[ends with the word Sync]
* HTTP module to make your http server
* Nodemon for restarting node
* [Headers - Web APIs](https://developer.mozilla.org/en-US/docs/Web/API/Headers)
* [Introduction to Node.js](https://nodejs.dev/)

**Useful Readings:**

* [Build an HTTP Server](https://nodejs.dev/build-an-http-server)
* [An introduction to the npm package manager](https://nodejs.dev/an-introduction-to-the-npm-package-manager)
* [The V8 JavaScript Engine](https://nodejs.dev/the-v8-javascript-engine)
* [How the Web works - Learn web development](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works)
* [What Is a Pure Function in JavaScript?](https://www.freecodecamp.org/news/what-is-a-pure-function-in-javascript-acb887375dfe/)
* [Circular dependency in Node.js and Nest.js](https://dev.to/successgilli/circular-dependency-in-nodejs-and-nestjs-3e1d) (read the Node.js part only)

**Must read before Day3:**

* [**REST API Best Practices – REST Endpoint Design Examples**](https://www.freecodecamp.org/news/rest-api-best-practices-rest-endpoint-design-examples/)