**Final assignment: Combining Node.js and AJAX Version**

**Submission**

The final assignment represents 20% of the final score. The assignment was introduced on the 4th of March 2024. Please submit the following (as one zip file) through web courses before Tuesday, April 30th, 2024, at 23:00. If you have any issues, please write to **luis.miralles@tudublin.ie**

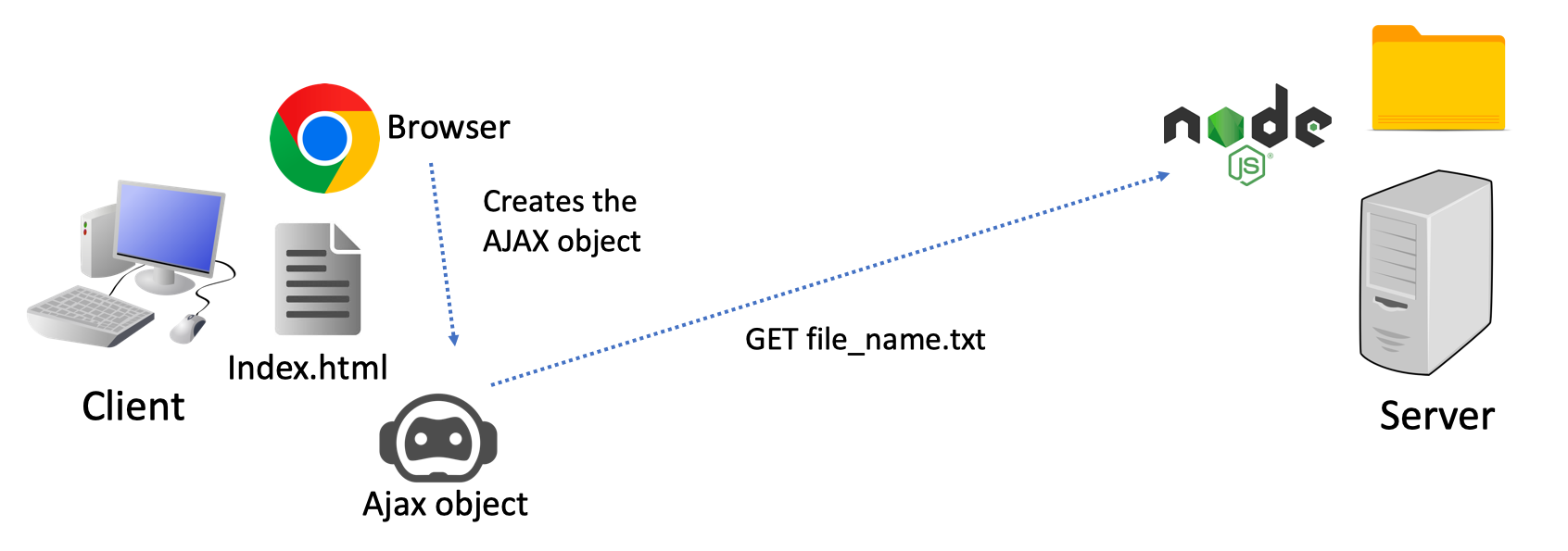
You will need to submit:

* Source code.
* 2 to 5-minute video recording of the functionalities of your code.
* Instructions on how to run your code.
* Please, save everything in a zip file with your name.

\**Late submissions will be penalized by an absolute mark of 5% for every day (or part thereof) that you are late (1 day is 5%, 2 days is 10%, 3 days is 15%, and so on). Submissions will not be accepted after Friday, May 10th.*

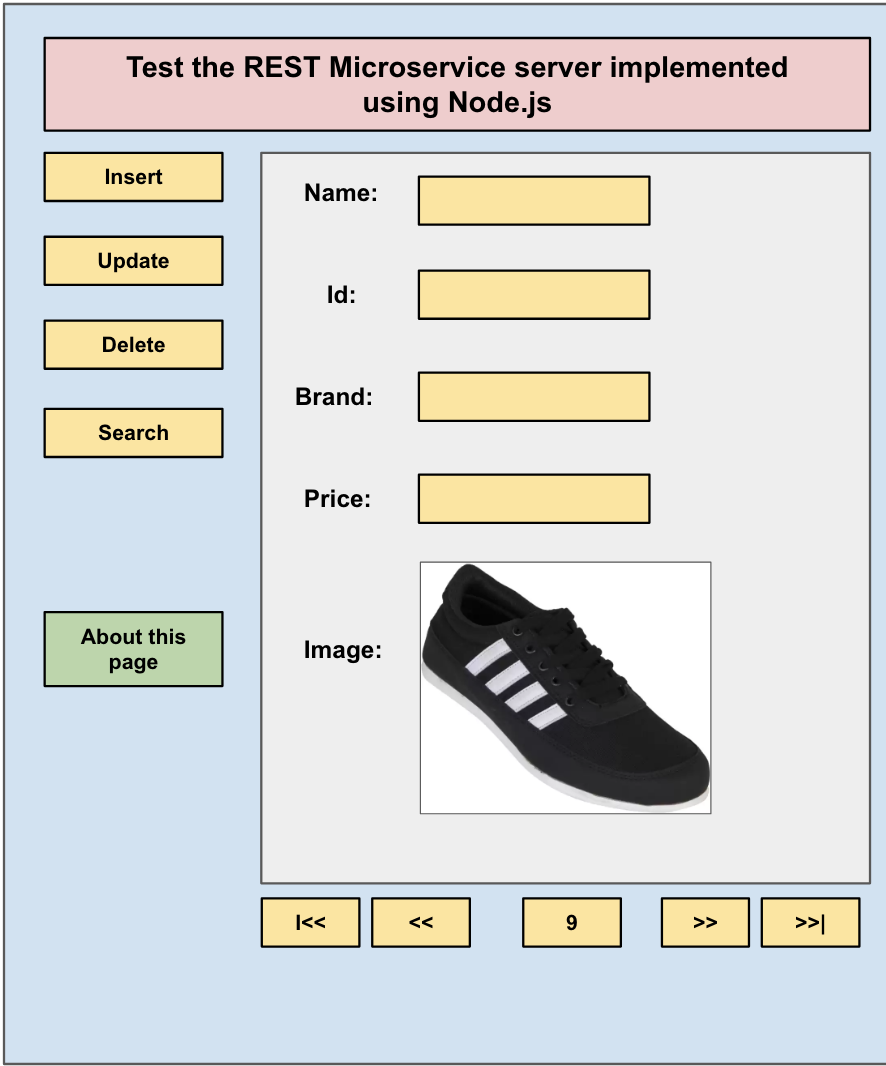
**Description of the assignment**

The idea of the assignment is to create microservices using Node.js and to use a client written in JavaScript and AJAX to test the microservices. You must build a scheme like the one shown in Figure 1. As you can see, the client is testing the REST services on the server using the Index.html page. The index.html page can have buttons to test the different options that can be done with a catalogue that has thousands of different items. The user needs to be able to do the following operations: Add items, remove items, search for items, and update items. You are allowed to use any library/framework that you consider such as Axios, React, Angular, etc.



**Figure 1: The server provides a file that the client will use for the different REST services of the server.**

To test the server services, you must create your index.html page. Figure 2 shows an example from which you can get some ideas. But the idea of the assignment is that you create something more creative and interesting.



**Figure 2: Example of the index.html page to test the different functionalities of the server.**

You can download the catalogue of products in the form of a JSON object from the following link [open-data-set/products.json at master](https://github.com/BestBuyAPIs/open-data-set/blob/master/products.json). However, maybe to start building your application you can use a smaller version of the list (let’s say 10 items) so it will take less to debug. And when you have it running you can use the full list of products. The development of a User Interface (UI) intuitive, usable, and well-designed is an important factor of the assignment.

|  |
| --- |
| A tip to make the assignment easier is to Install the MongoDB simulator using the command: npm install mongodb-memory-server mongodb. To see how this works, you can download the provided zip file "MongoExample.zip" in the Lab tab and follow the included instructions to execute the code. Then, you can execute the code and observe the output. Take note of any challenges or issues encountered during the process. |

In the following URL, you can find an example of a server implementing services using REST, but before you delve into the tutorial you need to know that you will get an error. So you better download directly the code from Git Hub that you can find inside the same website and execute it.

* Rest implementation tutorial: [Node.js and Express Tutorial: Building and Securing RESTful APIs](https://auth0.com/blog/node-js-and-express-tutorial-building-and-securing-restful-apis/)
* You can test this code easily if you download the code from the repository: [GitHub - auth0-blog/auth0-express](https://github.com/auth0-blog/auth0-express)
* To test the application easily (without using authentication tokens). Comment the line 49 in the index.html file where it says “//app.use(checkJwt);”. To execute the code, install Node.js on your computer. Then open a terminal and inside the downloaded GitHub code type “node src”. You will get an error saying you need to install package CORS (“Error: Cannot find module 'cors'”), so install the cors package using the following instruction: “*npm i cors”*. Run again *“node src”,* and if it went well, you should get the following message: **“Listening on port 3001”**.
* Open a terminal write in the terminal and type: “curl http://localhost:3001”

**Implementation**

The steps you need to run are:

* From the server perspective:
* Store the catalogue file in JSON format into an object.
* Store that object in a MongoDB dataset.
* Implement the methods to listen to requests via HTTP requests from different clients.
* Start the server to listen to the users’ requests.
* From the client’s perspective:
* Load the website in the server by accessing **localhost:8080/index.html**
* Allow the user to test the different REST services on the server by using the interface.

The basic functionality of the application is:

1. Create the index.html and include the documentation file.
2. Include the functionality on the server to allow the user the following operations: **Adding** an item, navigating through the items, **Update** an item, or **Delete** an item.
3. Include a MongoDB dataset where all the items and information of the app are stored.
4. Include the README file to execute the app.

To get a **better score,** the student should implement some new ideas or functionalities to make the app look more professional and interesting. Some examples of improvements are enabling the user to search for new items. List items by price, or by category, allowing the users to create their profile with some orders or records. Keep a list of the main users of the app how many products they have bought and so on.

**Deliverables**

The documents you should complete for the final assignment are:

* 2 to 5-minute Video Tutorial explaining the app and all its functionalities.
* Index.html to test the services.
* All the scripts for running the Node.js server
* README file with instructions on how to execute the application
* HTML subpage that is retrieved from the server when the user clicks on the button called “About this page” with explanations on the following points:
  + General description of how the application works (You can look at the PowerPoint of Node.js and AJAX)
  + Description of the technologies involved in your project.
  + Description of some weaknesses of your application
  + Description of some alternatives you could use to implement your application

**Grading**

The app will be graded following this scheme:

a. **FAIL** – (Below 40) - Not working at all, incomplete

b. **PASS** – (Between 40 and 60)- Basic functionality only, in a basic way.

c. **GOOD** – (Between 60 and 80) – The app covers all the basics and the student added new functionalities to make it look better.

d. **VERY GOOD** – (More than 80) – The student went the extra mile and made a great effort to make an app with many impressive functionalities and a very impressive interface trying to emulate a professional app.

**Interesting links**

* JavaScript tutorial: <https://www.w3schools.com/js/>
* Node.js Tutorial: <https://www.w3schools.com/nodejs/nodejs_mysql.asp>
* Ajax tutorial: <https://www.w3schools.com/xml/ajax_intro.asp>
* MongoDB and Node.js tutorial: [MongoDB and Node.js Tutorial - CRUD Operations](https://www.mongodb.com/developer/quickstart/node-crud-tutorial/) and [Node.js MongoDB Tutorial with Examples](https://www.guru99.com/node-js-mongodb.html)