

# Final assignment: Art application



## Submission

The final assignment constitutes 20% of the overall score. It was introduced on 6 March 2025. Kindly submit the following as a single zip file via web courses before Wednesday, 30 April 2025, at 23:00. Should you encounter any issues, please get in touch with [luis.miralles@tudublin.ie](mailto:luis.miralles@tudublin.ie)

You will need to submit:

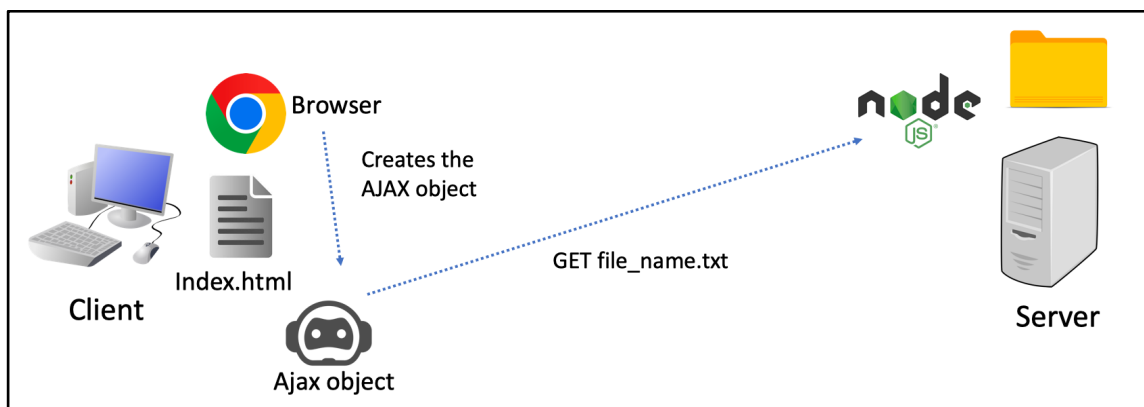
- Source code.
- 2 to 5-minute video recording of the functionalities of your code.
- Instructions on how to run your code.
- Please save everything in a zip file with your name on it.

*\*Late submissions will incur a penalty of 5% for each day (or part thereof) they are late (1 day is 5%, 2 days is 10%, 3 days is 15%, and so forth). Submissions will not be accepted after **Friday, 9th May**.*

## 1.- Description of the assignment

The assignment involves creating microservices using Node.js and employing a client written in JavaScript and AJAX to test these microservices. You should construct a scheme like that shown in Figure 1. As illustrated, the client tests the REST services on the server using the index.html page. This page can feature buttons to test the various options in a catalogue containing thousands of items. The user must be able to perform the following operations: add items, remove items, search for items, and update items.

**Note: You are allowed to use any library/framework that you consider instead of AJAX directly, such as Axios, React, Angular, etc.**



**Figure 1: The server supplies a file that the client will utilise for the various REST services provided by the server.**

To test the server services, you need to create your index.html page. Figure 2 presents an example from which you can draw some inspiration. However, the aim of the assignment is for you to develop something more imaginative and engaging.

Test the REST Microservice server implemented using Node.js

Insert

Update

Delete

Search

About this page


Name:

Id:

Author:

Year:

...:

Image: 

<< < 9 > >>

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

You can download the catalogue of artworks as a JSON object from the following link:

<https://github.com/MuseumofModernArt/collection/blob/main/Artworks.json>

Also in brightspace

However, to start building your application, you can use a smaller version of the list (let's say 10 items), so it will take less time to debug. When it is running, you can use the complete list of artworks. Developing an intuitive, usable, and well-designed user interface (UI) is essential to 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. However, before you delve into the tutorial, you must know you will encounter an

error. Therefore, it is advisable to download the code directly from GitHub, which is available on the same website, and execute it.

- Rest implementation tutorial: [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](#)
- To easily test the application (without using authentication tokens), comment out line 49 in the index.html file where it states “//app.use(checkJwt);”. To execute the code, install Node.js on your computer. Then, open a terminal and navigate to the downloaded GitHub code “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”; 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”

## 2.- Implementation

The steps you need to follow are:

From the server's perspective:

1. Store the catalogue file in JSON format in an object.
2. Save that object in a MongoDB dataset.
3. Implement methods to handle requests via HTTP from various clients.
4. Start the server to listen to user requests.

From the client's perspective:

1. Load the website on the server by accessing [localhost:8080/index.html](#)
2. Enable the user to test the different REST services on the server using the interface.

The basic functionality of the application is:

1. Create the index.html and include the documentation file.
2. Include functionality on the server that allows the user to **add** an item, navigate through the items, **Update** an item, or **Delete** an item.
3. Include a [MongoDB](#) dataset where all the app items and information are stored.
4. Include the [README](#) file to execute the app.

To achieve a better score, the student should implement new ideas or functionalities to enhance the app's professionalism and appeal. Examples of potential improvements include enabling users to search for new items, listing items by price or category, and allowing users to create profiles with their orders or records. Additionally, maintain a record of the app's primary users and the number of art works they have purchased, among other details on.

## 3.- Deliverables

The documents you need to complete for the final assignment are:

- A 2 to 5-minute video tutorial explaining the app and all its functionalities.
- An index.html file to test the services.
- All the scripts required to run the Node.js server.

- A README file with instructions on how to execute the application.
- An HTML subpage that is retrieved from the server when the user clicks on the button labelled "About this page," which provides explanations on the following points:
  - A general description of how the application functions (You may refer to the PowerPoint on Node.js and AJAX).
  - A description of the technologies involved in your project.
  - A description of some weaknesses of your application.
  - A description of some alternatives you could use to implement your application.

#### **4.- Grading**

The app will be graded according to the following scheme:

- a. FAIL – (Below 40) - Not functioning at all; incomplete.
- b. PASS – (Between 40 and 60) – Basic functionality only, presented rudimentarily.
- c. GOOD – (Between 60 and 80) – The app addresses all the essentials, and the student has added new functionalities to enhance its appearance.
- d. VERY GOOD – (Above 80) – The student has gone the extra mile, making a considerable effort to develop an app with numerous impressive functionalities and a striking interface, aiming to emulate a professional standard app.

#### **Interesting links**

- JavaScript Tutorial: <https://www.w3schools.com/js/>
- Node.js Tutorial: [https://www.w3schools.com/nodejs/nodejs\\_mysql.asp](https://www.w3schools.com/nodejs/nodejs_mysql.asp)
- Ajax Tutorial: [https://www.w3schools.com/xml/ajax\\_intro.asp](https://www.w3schools.com/xml/ajax_intro.asp)
- MongoDB and Node.js Tutorial: CRUD Operations and Examples in Node.js MongoDB.