**10X Academy Management Project**

**Your task is to build an 10x Academy Management Project. This project allows us to create, view, update, and delete a class and students. We will use MongoDB for our database and Node.js and Express.js to build the backend of our web application. You can use any other dependencies if required.**

**Note: Don’t create UI, just write backend logic for this project**

**SUMMARY:**

You will be building a server that can keep track of class and its Students List. Your server must be able to do the following:

* Create a new class with a number of students in that class. A new unique id would be created for each new class.
* You should also be able to create a new student with name and classId fields whereby you should be able to register a student to a class
* List out all classes
* Get information of specific class
* List out all students in a class
* Get one student specific details
* Update student’s information
* Delete a specified class
* Delete a student
* Your application will accept JSON and/or URL parameters and will return JSON data. Your server would be ready to be automatically integrated in a web system.

List of endpoints to be created.

**Test Case 1 - Create a new Class**

POST /v1/myClass

Input:

{class: "Arrays", studentsCount : 100}

Output:

{id: 2} (return a 201 code)

Notes:

The id returned is a unique id for the class that was just created.

**Test Case 2 - Register a new student to class**

POST /v1/myClass/:myClassId/students

Input:

{name: "Rahul", classId : 100}

Output:

{studentId: 89} (return a 201 code)

Notes:

The id returned is a unique id for the student that was just created.

**Test Case 3 - List out all classes**

    GET /v1/myClass

Input:

None

Output:

(return a 200 code)

{

   classes: [

     {id: 1, class: "Class 1", studentCount: 50},

     {id: 2, class: "Class 2", studentCount: 70}

   ]

}

Notes:

This endpoint list all tasks including their id's

**Test Case 4 - Get a specific class**

    GET /v1/myClass/:myClassId

Input :

id (passed through the URL)

Output:

(return a 200 code)

{id: 1, class: "Class 1", studentCount: 50},

On Error:

if id not found:

(return a 404 code)

{

    error: "There is no class at that id"

}

Notes:

This endpoint returns a specific class or returns a 404 not found response

**Test Case 5 - Get all students in a specific class**

    GET /v1/myClass/:myClassId/students

Input :

id (passed through the URL)

Output:

(return a 200 code)

{name: "Rahul", classId : 5, studentId : 45},

{name: "Rama", classId : 5, studentId : 47}

On Error:

if id not found:

(return a 404 code)

{

    error: "There are no students at this class"

}

Notes:

This endpoint returns a list of students or returns a 404 not found response

**Test Case 6 - Get one student details**

    GET /v1/myClass/:myClassId/students/:studentId

Input :

id (passed through the URL)

Output:

(return a 200 code)

{name: "Rama", classId : 5, studentId : 47}

On Error:

if id not found:

(return a 404 code)

{

    error: "There is no student of that id"

}

Notes:

This endpoint returns a specific student or returns a 404 not found response

**Test Case 7 - update student information**

 Put /v1/myClass/:myClassId/students/:studentId

Input:

id (passed through the URL)

Output:

None (return a 204 code)

Notes: This endpoint deletes a specific task. If the task doesn’t exist still send the same response

**Test Case 8 - Delete a specified class**

    Delete /v1/myClass/:myClassId

Input:

id (passed through the URL)

Output:

None (return a 204 code)

On Error:

if id not found:

(return a 404 code)

{

    error: "There is no task at that id"

}

Notes: This endpoint deletes a specific class or returns a 404 not found response

**Test Case 9 - Delete a student**

  Delete /v1/myClass/:myClassId/students/:studentId

Input:

id (passed through the URL)

Output:

None (return a 204 code)

On Error:

if id not found:

(return a 404 code)

{

    error: "There is no task at that id"

}

Notes: This endpoint deletes a specific student or returns a 404 not found response

**Submission**

1. Zip the entire code folder,
2. Submit the Zip File.
3. Submit screenshot of postman (working API).

You can use the document below as a guide to learn [how to zip a folder](https://docs.google.com/document/d/1wLV5j5eNMfY5GQ05R6ApCdqOEkb-1gbIALb3QYEWLJk/edit?usp=sharing) and [how to submit in a feedback form](https://docs.google.com/document/d/1wLV5j5eNMfY5GQ05R6ApCdqOEkb-1gbIALb3QYEWLJk/edit?usp=sharing) : [Steps for submitting in a Feedback Form](https://docs.google.com/document/d/1wLV5j5eNMfY5GQ05R6ApCdqOEkb-1gbIALb3QYEWLJk/edit?usp=sharing) (Click this)