**Background**

School administrators and teachers need a system where they can perform administrative functions on their students.

Some assumptions that you can make regarding the system:

1. ***Login*** and ***access control*** have already been handled.
2. Teachers and students are identified by their email addresses.
3. Each teacher can teach multiple subjects across multiple classes.
4. A class can consists of 2 types of students, internal and external.
5. External students’ details are retrieved from an external system.
6. The external system will only update external student details at 12:00 am daily.
7. A class size can go up to around 500 students in total.

**Your Task**

1. Use one of two base codes (JavaScript or TypeScript) provided.
2. Extend the base code with a set of API endpoints, listed under *User Stories* below.
3. Your code must be hosted on GitHub, or any other similar publicly accessible code repository.
4. You should overwrite the default README.md to includes the following:
   1. The NodeJS version that you are using.
   2. Instructions for running the local instance of your API server as we need to be able to launch and test your solution locally.
5. You may add any new libraries needed, without replacing the following libraries:
   1. **Framework**: ExpressJS
   2. **ORM**: Sequelize
   3. **Database:**  MySql 8.0
   4. **HTTP Client:** Axios
   5. **Multipart Parser:** Multer
   6. **CSV Parser:** csv-parser
   7. **Logger:** WinstonJS
   8. **Test Runner:** Jest
6. Use the **async/await** syntax instead of Promise chaining (.then(), .catch()).
7. Include unit tests.
8. Your API will be subjected to automated test tools, so **please adhere closely to the given specifications** (according to ***User Stories*** section)
9. Send us the URL of the code repository containing the completed assignment. Ensure the URL does not have the word “Ufinity" in it.
10. If you are selected for a face-to-face interview, you should be prepared to:
    1. Walk through your code to interviewers.
    2. Explain any design decisions you have made.
    3. Modify the API endpoints or implement more endpoints.

**Assessment Criteria**

1. Readability
2. Maintainability
3. Code cleanliness
4. Code structure/design, e.g. modularity, testability
5. Database design, e.g. normalized, correct keys and indices
6. Balance between performance and readability
7. Meaningful error responses
8. Appropriate logs
9. Appropriate comments
10. Appropriate typing, which facilitate code completion and type checking, if typescript is used

**Queries**

If you have any queries, contact the Ufinity person who is currently liaising with you.

**User Stories**

1. **As an Administrator, I want to upload Teachers, Students, Classes information in a CSV (comma-separated value) file, so that I can use the system for administrative purposes.**

Description

This API should be able to:

* create new record(s)
* update existing record(s)
* delete existing record(s)

The CSV file shall include the following information:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S/N** | **Name** | **Description** | **Mandatory** | **Type** |
| 1 | teacherEmail | The unique identifier of Teacher | Yes | string |
| 2 | teacherName | The display name of Teacher | Yes | string |
| 3 | studentEmail | The unique identifier of Student | Yes | string |
| 4 | studentName | The display name of Student | Yes | string |
| 5 | classCode | The unique identifier of Class | Yes | string |
| 6 | className | The display name of a Class | Yes | string |
| 7 | subjectCode | The unique identifier of Subject | Yes | string |
| 8 | subjectName | The display name of Subject | Yes | string |
| 9 | toDelete | Will be 1 if the Teacher is no longer teaching this Student | Yes | 0 or 1 |

Assumptions

* If teacherEmail is the same for multiple rows, always take the teacherName of the latest record. This applies to Students, Classes and Subjects too.
* 1 teacher can teach multiple classes.
* 1 student can attend multiple classes.
* 1 class can be taught different teachers, provided they are all teaching different subject.

Request

**Method:** POST **Endpoint:** /api/upload  
**Headers:** Content-Type: multipart/form-data

Response  
**Success HTTP Code:** 204  
**Error HTTP Code:** 400 or 500

1. **As a Teacher, I am able to retrieve the Students list of my class in a paginated manner so that I can perform administrative action on the students individually.**

Description

A class can be made up of 2 types of Students, *Internal* and *External*. Internal Students are those uploaded using API 1. External Students are retrieved from an External System. External Students data is considered confidential, and therefore cannot be stored in the local database **except for their *id*** **from the External System**.

The External System is **already included in the base code**, and is accessible via [http://localhost:8080/students?class=<CLASSCODE>](http://localhost:8080/students?class=%3cCLASS%3c)&offset=<OFFSET>&limit=<LIMIT>.

The response body of this provided API will be in the following JSON format:

{

“count”: number,

“students”: [

{

“id”: number,

“name”: string,

“email”: string

}

]

}

The API should return only the required list of students, based on the pageSize and page number, sorted in alphanumerical order.

Assumptions

* External students’ information are to be fetched on demand via *localhost:8080/students?class=<CLASS>&offset=<OFFSET>&limit=<LIMIT>*.
* Your API only needs to support infinite scrolling i.e. skipping of pages is not required.

Request **Method:** GET **Endpoint:** /api/class/<classCode>/students*, where classCode is the required Class*

*Code*  
**Query:**

|  |  |  |
| --- | --- | --- |
| **S/N** | **Query Param** | **Description** |
| **1** | offset | How many records to ignore before returning the first record. Equivalent to offset in MySQL |
| **2** | limit | How many students to retrieve |

Response  
**Success HTTP Code:** 200  
**Error HTTP Code:** 400 or 500  
**Body:** Same as the **External System API’s**(see API 2) response body.  
 **More properties can be returned if needed.**

1. **As an Administrator, I should be able to update the Class Name, so that the new Class Name will be reflected accordingly in the system.**

Description

This API will allow the user to change Class Name based on the Class Code.

Request **Method:** PUT **Endpoint:** /api/class/<classCode>,  *where classCode is the required Class Code*  
**Body:**

{

“className”: string

}

Response  
**Success HTTP Code:** 204  
**Error HTTP Code:** 400 or 500

1. **As an Administrator, I should be able to generate a report on a Teacher’s workload, so that I use it for planning.**

Description

This API will retrieve the following data for each Teacher:

* 1. Subject Code
  2. Subject Name
  3. How many classes is the Teacher teaching for the Subject (in a)

Request **Method:** GET **Endpoint:** /api/reports/workload

Response  
**Success HTTP Code:** 200  
**Error HTTP Code:** 500  
**Body:**

{

“dummy teacher name 1”: [

{

“subjectCode”: string,

“subjectName”: string,

“numberOfClasses”: number,

}

],

“dummy teacher name 2”: [

{

“subjectCode”: string,

“subjectName”: string,

“numberOfClasses”: number,

}

]

}