Background

Ufinity School of Technology (UST) is engaging you to design and develop the backend APIs of School Administration System (SAS), which will be used by school administrators and teachers to perform various administrative functions.

Every class in UST can go up to 500 students and is made up of the following groups of student:

1. Students enrolled directly to UST (local student)
2. Students enrolled via UST subsidiary school (external student)

As external students are not enrolled directly to UST, external students' data are confidential and cannot be stored in the SAS local database. They must be retrieved from the subsidiary school's system (named: external) on the fly.

The following can be assumed about the teachers and students:

1. A teacher can teach in multiple classes.
2. A teacher can teach multiple subjects, regardless to the same or different class.
3. 2 different teachers can teach the same subject in the same class.
4. A student can be in multiple classes.

You are tasked to implement the following APIs, with more details found under *User Stories* section:

1. **DataUpload** API that **takes in a csv** to populate the required data.
2. **StudentListing** API that **returns both internal and external students of a class**, in a single paginated list sorted in alphanumerical order.
3. **UpdateClassName** API that allows the user to **update the name of an existing class.**
4. **WorkloadReport** API that **returns the required data in JSON format**, so that it can be used to generate the report.

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* section.
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:
   a. The NodeJS version that you are using.
   b. 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:
   a. **Framework**: ExpressJS
   b. **ORM**: Sequelize
   c. **Database:** MySql 8.0
   d. **HTTP Client:** Axios
   e. **Multipart Parser:** Multer
   f. **CSV Parser:** csv-parser
   g. **Logger:** WinstonJS
   h. **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:
    a. Walk through your code to interviewers.
    b. Explain any design decisions you have made.
    c. 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

   DataUpload API should be able to:

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

   The input 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

   Requirements

   - If teacherEmail is the same for multiple rows, always take the teacherName of the latest record. This applies to Students, Classes and Subjects too.
   - Teachers and students can be uniquely identified by their email address.

   Expected Request

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

   Expected Response

   Success Code: 204
   Error Code: 400 or 500

2. **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

StudentListing API should return the both internal and external students of a class, in a single paginated list sorted in alphanumerical order.

External Student Data

ExternalStudentListing API is accessible via http://localhost:8080/students?class=<CLASSCODE>&offset=<OFFSET>&limit=<LIMIT> when you start up the base code

The response body of **ExternalStudentListing** API will be in the following JSON format:

```
{
  "count": number,
  "students": [
    {
      "id": number,
      "name": string,
      "email": string
    }
  ]
}
```

Requirements

- External students' data are to be fetched on demand via **ExternalStudentListing** API.
- External students' data cannot be stored in the local database.
- **StudentListing** API should return local and external students in a single paginated list.
- **StudentListing** API should return only the required list of students, based on offset and limit.
- **StudentListing** API only needs to support infinite scrolling i.e. skipping of pages is not required.

Expected 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

Expected Response

Success HTTP Code: 200

Error HTTP Code: 400 or 500

Body: Same as the **ExternalStudentListing API's** response body.
Additional properties can be returned if required.

3. **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

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

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

```
{
  "className": string
}
```

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

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

Description

The WorkloadReport API will return the required data in JSON format, so that the data can be used to generate a report in the frontend.

Requirements

- The **WorkloadReport** API should return the required data in JSON format.
- The data required for each teacher are:
  a) Subject Code
  b) Subject Name
  c) How many classes is the Teacher teaching for the Subject (in a)

Expected Request

Method: GET
Endpoint: /api/reports/workload

Expected Response

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

```
{
  "dummy teacher name 1": [
    {
      "subjectCode": string,
      "subjectName": string,
      "numberOfClasses": number,
    },
    {
      "subjectCode": string,
      "subjectName": string,
      "numberOfClasses": number,
    }
  ],
  "dummy teacher name 2": [
    {
      "subjectCode": string,
      "subjectName": string,
      "numberOfClasses": number,
    }
  ]
}
```