API Assessment

Background

Teachers need a system where they can perform administrative functions for their students. Teachers and students are identified by their email addresses.

Your Task

Your task is to:

1. Develop a set of API endpoints, listed under *User Stories* below, for teachers to perform administrative functions for their classes.
    - Your code must be hosted on Github, or any other similar service, in a publicly-accessible repository.
    - You may assume that login and access control have already been handled.
2. *(Optional)* Deploy your API to any publicly accessible hosting environment.

Requirements/Expectations

1. Your code repository should contain a `README.md` that includes the following:
    - Link(s) to the hosted API (if applicable)
    - Instructions for running local instance of your API server; we need to minimally be able to launch and test your solution locally
2. Please use **Golang** for the backend code if possible (can fallback to Node.js if needed).
3. Please use PostgreSQL or MySQL as the database.
4. Please include unit tests.
5. If you are selected for a face-to-face interview, you should be prepared to:
    - Walk through your code to interviewers
    - Explain any design decisions you've made
    - Modify the API endpoints, or implement more endpoints

Important!

- We will assess your submission holistically (i.e. not just in terms of functionality), including factors such as:
    - Readability and code cleanliness
    - Secure coding practices
    - Code structure/design, e.g. modularity, testability
- Your API will be subjected to automated test tools, so please adhere closely to the given specs.
    - (Optional) You can provide a Postman collection for the APIs that you've implemented, *but* we can (and likely will) still use our own tools as well to test your API.

User stories

1. As a teacher, I want to register one or more students to a specified teacher.

A teacher can register multiple students. A student can also be registered to multiple teachers.

- Endpoint: `POST /api/register`
- Headers: `Content-Type: application/json`
- Success response status: HTTP 204
- Request body example:

```
{
  "teacher": "teacherken@gmail.com"
  "students":
    [
      "studentjon@gmail.com",
      "studenthon@gmail.com"
    ]
}
```

2. As a teacher, I want to retrieve a list of students common to a given list of teachers (i.e. retrieve students who are registered to ALL of the given teachers).

- Endpoint: `GET /api/commonstudents`
- Success response status: HTTP 200
- Request example 1: `GET /api/commonstudents?teacher=teacherken%40gmail.com`

- Success response body 1:

```
{
  "students" :
    [
      "commonstudent1@gmail.com",
      "commonstudent2@gmail.com",
      "student_only_under_teacher_ken@gmail.com"
    ]
}
```

- Request example 2: `GET /api/commonstudents?teacher=teacherken%40gmail.com&teacher=teacherjoe%40gmail.com`
- Success response body 2:

```
{
  "students" :
    [
      "commonstudent1@gmail.com",
      "commonstudent2@gmail.com"
    ]
}
```

3. As a teacher, I want to suspend a specified student.

- Endpoint: `POST /api/suspend`
- Headers: `Content-Type: application/json`
- Success response status: HTTP 204
- Request body example:

```
{
  "student" : "studentmary@gmail.com"
}
```

4. As a teacher, I want to retrieve a list of students who can receive a given notification.

A notification consists of:

- the teacher who is sending the notification, and
- the text of the notification itself.

To receive notifications from e.g. 'teacherken@gmail.com', a student:

- MUST NOT be suspended,
- AND MUST fulfill *AT LEAST ONE* of the following:
    i. is registered with “teacherken@gmail.com"
    ii. has been @mentioned in the notification

The list of students retrieved should not contain any duplicates/repetitions.

- Endpoint: `POST /api/retrievefornotifications`
- Headers: `Content-Type: application/json`
- Success response status: HTTP 200
- Request body example 1:

```
{
  "teacher":  "teacherken@gmail.com",
  "notification": "Hello students! @studentagnes@gmail.com
@studentmiche@gmail.com"
}
```

- Success response body 1:

```
{
  "recipients":
    [
      "studentbob@gmail.com",
      "studentagnes@gmail.com",
      "studentmiche@gmail.com"
    ]
}
```

In the example above, studentagnes@gmail.com and studentmiche@gmail.com can receive the notification from teacherken@gmail.com, regardless whether they are registered to him, because they are @mentioned in the notification text. studentbob@gmail.com however, has to be registered to teacherken@gmail.com.

- Request body example 2:

```
{
  "teacher":  "teacherken@gmail.com",
  "notification": "Hey everybody"
}
```

- Success response body 2:

```
{
  "recipients":
    [
      "studentbob@gmail.com"
    ]
}
```

Error Responses

For all the above API endpoints, error responses should:

- have an appropriate HTTP response code
- have a JSON response body containing a meaningful error message:

```
{ "message": "Some meaningful error message" }
```