Science Exhibition Management – Spring boot

A school has a service that automates students participation data in their schools 25th Foundation Day – Silver Jubilee Celebration. Each student provides simple data to participate in the exhibition. The exhibition application service helps to understand how many students are participating in the exhibition and acquire details of students based on some parameters.

As step 1, create a service that supports REST APIs for creating and deleting student data. A few more APIs required would be to fetch the student with a particular id and also fetch the details of the student data's using **pagination**. A detailed explanation about the APIs and data is given below.

Each object is a JSON object with the following keys:

1. id - Id of the exhibition. [INTEGER]
2. eventName – name of the event will be final with the value =" TCH 25th Foundation Day - Silver Jubilee Celebrations". [STRING].
3. studentName - name of the student. [STRING]
4. idCardNumber- Unique idCard number of the student. [LONG]
5. topic- Topic of the student. [STRING]
6. guideMember – teacher or guide who is helping the student. [STRING]

EXAMPLE

Request Body


```
{
  "studentName":"studentTwo",
  "idCardNumber":130080,
  "topic":"topicTwo",
  "guideMember":"guideTwo"
}
```

Response Body


```
{
   "id": 2,
   "eventName": "TCH 25th Foundation Day - Silver Jubilee Celebrations",
   "studentName": "studentTwo",
   "idCardNumber": 130080,
   "topic": "topicTwo",
   "guideMember": "guideTwo"
}
```


The following APIs need to be implemented:

1. Adding new student data- **POST** request should be created to add a new student data. The Api endpoint would be /exhibition/add . The request body contains the details of the student. HTTP response should be 201. If some data validations failed and the data is not inserted into the collection, then you should send a response code of 400.

2. Getting all student data– **GET** request to endpoint /exhibition?pageNo={pageNo}&pageSize={pageSize}&sortBy={sortBy} should return the details of all the student data in that particular page with the given page number, page size and sortBy variable. The HTTP response code should be 200. If no student data exists return with this page, return a response code of 400.

Note:

- ❖ pageSize gives the total number of students data that can be displayed on a page. Default value=10.
- ❖ pageNo gives the page number in which the student data's present in the page are displayed. Default value=0.
- ❖ sortBy gives the order in which the student data must be displayed. Default value=id.

3. Updating student data by id – **PATCH** request to endpoint /exhibition/{id} should update the Exhibition's topic, guideMember by id as PathVariable. The HTTP response code should be 200. If no student data exists return with this id, return a response code of 400.

4. Deleting any student data by id - **DELETE** request to endpoint /exhibition/{id}. The route would contain the id of the student data that needs to be deleted. HTTP response should be 200. If id validations failed and the data is not deleted from the collection, then you should send a response code of 400.