GDS SWE Challenge

THE MINI PROJECT

BACKGROUND

Develop a web application with the following HTTP endpoints.

Path	Method	Params	Description

/users	GET	<ul><li>**min** - minimum salary. Optional, defaults to 0.0..</li><li>**max** - maximum salary. Optional, defaults to 4000.0.</li><li>**offset** - first result among set to be returned. Optional, defaults to 0.</li><li>**limit** - number of results to include. Optional, defaults to no limit.</li><li>**sort** - **NAME** or **SALARY**, non-case sensitive. Optional, defaults to no sorting. Sort only in ascending sequence.</li></ul>	Return list of users that match specified criteria and ordering. in JSON form:
			
			`{`
			`  "results": [`
			`    {`
			`      "name": "John",`
			`      "salary": 3000.0`
			`    },`
			`    {`
			`      "name": "John 2",`
			`      "salary": 3500.0`
			`    }`
			`  ]`
			`}`

/upload	POST	<ul><li>Content type: **multipart/form-data**</li><li>Form field name: **file**</li><li>Contents: CSV data. See below.</li></ul>	Return success or failure. 1 if successful and 0 if failure. If failure, HTTP status code should not be HTTP_OK.
			File upload is an all-or-nothing operation. The entire file's changes are only applied after the whole file passes validation. If the file has an error, none of its rows should be updated in the database.
			
			`{`
			`  "success": 1`
			`}`

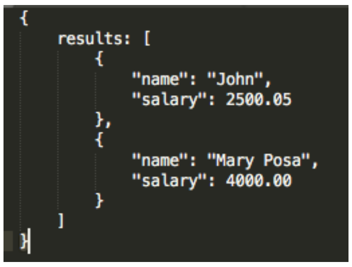
- Both HTTP endpoints return **application/json** content type, and **HTTP_OK** if successful.
- If there is an error processing HTTP request, appropriate HTTP status code should be returned with the result body of the form “**{ “error”: “...” }** to include additional error description.
- CSV file is structured as follows.
    - **Two columns NAME and SALARY.**
        - Name is text.
        - Salary is a floating point number, You do not need to ensure a specific number of decimal points.
        - Salary must be >= 0.0. All rows with salary < 0.0 are ignored.
    - First row of the CSV file is always **ignored** and treated as a header row.
    - If there is a formatting error (ie. salary number cannot be parsed), or incorrect number of columns during input, the CSV file should be rejected. However, rows with salary < 0.0 are skipped, without skipping the entire file.

- User name **must be unique** in the system. If there is already another user with the same name in the database during loading, the record in the database is replaced with the data from the file.
- The database should be **pre-loaded** with seed data on first invocation of the application.
- Bonus: It is not required but desirable to be able to process concurrent upload requests simultaneously.

These are not part of the requirements, but do keep these in mind, and also be prepared to explain your approach on the following:

- What kind of testing would you perform on the application in addition to unit testing?
- How would you scale up the system to handle potentially millions of files?
- CSV files can be very large and take a long time to process. This can be a problem using HTTP POST calls. How would you update the design to handle HTTP POST requests that can potentially take a long time to execute?

<u>Acceptance Criteria 1</u>

- When the application starts up, it should be pre-loaded with testable data.
- Expose /users endpoint that returns users with valid salary (0 <= salary <= 4000)
- Example json response:
    - 

```
{
    results: [
        {
            "name": "John",
            "salary": 2500.05
        },
        {
            "name": "Mary Posa",
            "salary": 4000.00
        }
    ]
}
```

- Additional sub-criterias:
    - 1.1: **min** and **max**
    - 1.2: **order**. Also include illegal order parameters.
    - 1.3: **offset** and **limit**.

<u>Acceptance Criteria 2</u>

- Upload with a properly structured CSV file. You may include any data in the csv file.
- File should include some new data that is not in the database, and some that overwrites the database.
- File should include rows with negative and 0.00 salary.
- **/users** should work as expected after the upload and that there are new results returned as well as previous results that have been overwritten. Negative rows should be ignored in the input and 0.0 should be updated and returned.

Example csv file:

Name, Salary
John,2500.05 Mary Posa, 3000.00

GOVTECH
SINGAPORE

Acceptance Criteria 3

- Upload with an improperly structured CSV file that should contain at least some good rows.
- File should be rejected and none of the good rows should have been applied.

TASK

Use spring boot and java for the project.
https://spring.io/projects/spring-boot

APPROACH

This is a simple assignment, so please feel free to apply coding best practices and technical designs that can demonstrate your understanding.

SUBMISSION

Please upload your code to GitHub and email us the link of the repository. You can also include other documents that you deem necessary for the submission, good luck! ☺