My docker-compose stack is fully working.

# New endpoints in this assignment

1. Add a specified student

Endpoint:

/add\_student

Code:

一張含有 文字 的圖片

自動產生的描述

In this operation, the student data should be represented in JSON format. The API will first check the availability of the database. If it is not available, the API will return an error message with HTTP response code of 503. Then, the API will check whether student id and student name are included in the JSON data or not. If not, the API will return an error message with HTTP response code of 400. Next, the API will check the student id whether it exists in the database. If it already exists. A conflict has occurred. The API will return an error message with HTTP response code of 409. Finally, the API will insert the student record into the database and return a success message with HTTP response code of 201.

The sample instruction and result are as follows.

Instruction:

curl -X POST -v http://localhost:15000/add\_student -H 'Content-Type: application/json' -d '{"student\_id":"55555", "name":"Ian", "dept\_name":"Comp. Sci.", "gpa":3.0}'

Results:

If the insertion operation is successful, the API will return the following JSON with HTTP response code of 201.

一張含有 文字 的圖片

自動產生的描述

If the student id or student name are not included in the JSON data, the insertion operation will fail, and the API will return the following JSON with HTTP response code of 400.

一張含有 文字 的圖片

自動產生的描述

If this student already exists in the database, the insertion operation will fail, and the API will return the following JSON with HTTP response code of 409.

一張含有 文字 的圖片

自動產生的描述

If the database server is not connected, the API will return the following JSON with HTTP response code of 503.

一張含有 文字 的圖片

自動產生的描述

1. Remove a specified student

Endpoint:

/delete\_student/<student\_id>

Code:

一張含有 文字 的圖片

自動產生的描述

In this operation, users can delete a student by the student id. The API will first check the availability of the database. If it is not available, the API will return an error message with HTTP response code of 503. Then, the API will check whether the student id exists in the database. If it is not found, The API will return an error message with HTTP response code of 404. Finally, the API will delete the student record in the database and return a success message with HTTP response code of 200.

The sample instruction and result are as follows.

Instruction:

curl -X DELETE -v http://localhost:15000/delete\_student/55555

Results:

If the deletion operation is successful, the API will return the following JSON with HTTP response code of 200.

一張含有 文字 的圖片

自動產生的描述

If this student is not found in the database, the API will return the following JSON with HTTP response code of 404.

一張含有 文字 的圖片

自動產生的描述

If the database server is not connected, the API will return the following JSON with HTTP response code of 503.

一張含有 文字 的圖片

自動產生的描述

1. Add a course taken by a specified student

Endpoint:

/add\_course

Code:

一張含有 文字 的圖片

自動產生的描述

In this operation, the course data should represent in JSON format. The API will first check the availability of the database. If it is not available, the API will return an error message with HTTP response code of 503. Then, the API will check whether student id and course id are included in the JSON data or not. If not, the API will return an error message with HTTP response code of 400. Next, the API will check whether the student id exists in the database. If it is not found, The API will return an error message with HTTP response code of 404. Also, the API will check whether the student took this course already. If yes, the API will return an error message with HTTP response code of 409. Finally, the API will insert the course record into the database and return a success message with HTTP response code of 201.

The sample instruction and result are as follows.

Instruction:

curl -X POST -v http://localhost:15000/add\_course -H 'Content-Type: application/json' -d '{"student\_id":"33333", "course\_id":"COMP2345", "credits":"3"}'

Results:

If the insertion operation is successful, the API will return the following JSON with HTTP response code of 201.

一張含有 文字 的圖片

自動產生的描述

If the student id or course id are not included in the JSON data, the insertion operation will fail, and the API will return the following JSON with HTTP response code of 400.

一張含有 文字 的圖片

自動產生的描述

If this student is not found in the database, the API will return the following JSON with HTTP response code of 404.

一張含有 文字 的圖片

自動產生的描述

If this course with this student id exists in the database, the API will return the following JSON with HTTP response code of 409.

一張含有 文字 的圖片

自動產生的描述

If the database server is not connected, the API will return the following JSON with HTTP response code of 503.

一張含有 文字 的圖片

自動產生的描述

1. Remove a specified course taken by a specified student

Endpoint:

/delete\_course/<course\_id>/<student\_id>

Code:

一張含有 文字 的圖片

自動產生的描述

In this operation, users can delete a specified course taken by a specified student by the student id and course id. The API will first check the availability of the database. If it is not available, the API will return an error message with HTTP response code of 503. Then, the API will check whether the student id exists in the database. If it is not found, the API will return an error message with HTTP response code of 404. The API will also check whether this course with this student exists in the database. If it is not found, the API will return an error message with HTTP response code of 404. Finally, the API will delete this specified course record with this specified student id in the database and return a success message with HTTP response code of 200.

The sample instruction and result are as follows.

Instruction:

curl -X DELETE -v http://localhost:15000/delete\_course/COMP1234/22222

Results:

If the deletion operation is successful, the API will return the following JSON with HTTP response code of 200.

一張含有 文字 的圖片

自動產生的描述

If this student is not found in the database, the API will return the following JSON with HTTP response code of 404.

一張含有 文字 的圖片

自動產生的描述

If this course record with this student is not found in the database, the API will return the following JSON with HTTP response code of 404.

一張含有 文字 的圖片

自動產生的描述

If the database server is not connected, the API will return the following JSON with HTTP response code of 503.

一張含有 文字 的圖片

自動產生的描述

# Automated unit tests

In this assignment, 19 unit test cases, that are used for testing the various endpoints of the REST API, are designed as follows.

1. Test endpoint /students

一張含有 文字, 橙色, 靠近 的圖片

自動產生的描述

The expected result is HTTP response code of 200.

1. Test endpoint/students/<student\_id>

This will get the first student’s id from requesting endpoint /students. Then, it will use this id to get a specified student record.

The expected result is HTTP response code of 200.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint/students/<student\_id> with incorrect student id

The expected result is HTTP response code of 404 and JSON message of {'error': 'not found'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /takes

The expected result is HTTP response code of 200.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /takes/<student\_id>

It will get the student id from requesting endpoint /takes who took at least one course. Then it will get the course record for the specified student using this student id.

The expected result is HTTP response code of 200.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /takes/<student\_id> with incorrect student id

The expected result is HTTP response code of 404 and JSON message of { error: 'not found'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /add\_student

This will delete the student record, which is expected to add to the database, to avoid conflict. Then, it will try to create the student record.

The expected result is HTTP response code of 201 and JSON message of { 'message': 'The insertion operation is successful.'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /add\_student with student id which student already exists

This will delete the student record, which is expected to add to the database, to avoid conflict. Then, it will try to create the student record using an existing student id.

The expected result is HTTP response code of 409 and JSON message of {'error': 'This student already exists'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /add\_student with incomplete data

This will delete the student record, which is expected to add to the database, to avoid conflict. Then, it will try to create the student record using incomplete data.

The expected result is HTTP response code of 400 and JSON message of {'error': 'Missing data. The student id and student name must be included.'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /delete\_student/<student\_id>

This will create the student record, which is expected to be deleted in the database. Then, it will delete this student record.

The expected result is HTTP response code of 200 and JSON message of {'message': 'The deletion operation is successful.'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /delete\_student/<student\_id> with nonexistent student id

This will delete a student record using a nonexistent student id.

The expected result is HTTP response code of 404 and JSON message of {'error': 'This student is not found'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /add\_course

This will delete the course record, which is expected to add to the database, to avoid conflict. Then, it will try to create the course record.

The expected result is HTTP response code of 201 and JSON message of {'message': 'The insertion operation is successful.'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /add\_course with incomplete data

This will delete the course record, which is expected to add to the database, to avoid conflict. Then, it will try to create the course record using incomplete data.

The expected result is HTTP response code of 400 and JSON message of {'error': 'Missing data. The student id, course id and credits must be included.'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /add\_course with nonexistent student id

This will delete the course record, which is expected to add to the database, to avoid conflict. Then, it will try to create the course record using nonexistent student id.

The expected result is HTTP response code of 404 and JSON message of {'error': 'This student is not found'}

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /add\_course with existing course

This will delete the course record, which is expected to add to the database, to avoid conflict. Then, it will create the course record two times.

The expected result is HTTP response code of 409 and JSON message of {'error': 'This student already took this course.'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /delete\_course /<course\_id>/<student\_id>

This will create the course record, which is expected to delete in the database. Then, it will delete the course record.

The expected result is HTTP response code of 200 and JSON message of {'message': 'The deletion operation is successful.'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint /delete\_course /<course\_id>/<student\_id> with nonexistent student id

It will delete the course record using a nonexistent student id.

The expected result is HTTP response code of 404 and JSON message of {'error': 'This student is not found'}.

一張含有 文字, 螢幕, 螢幕擷取畫面, 靠近 的圖片

自動產生的描述

1. Test endpoint /delete\_course /<course\_id>/<student\_id> by deleting a nonexistent course record

It will delete the course record two times.

The expected result is HTTP response code of 404 and JSON message of {'error': 'This course record with this student id is not found.'}.

一張含有 文字 的圖片

自動產生的描述

1. Test endpoint without database server

It will stop the MongoDB container. Next, this will delete the student record, which is expected to add to the database, to avoid conflict. Then, it will try to create the student record. After the request process, it will restart the MongoDB container.

The expected result is HTTP response code of 503 and JSON message of {'error': 'The DB service is unavailable'}.

一張含有 文字 的圖片

自動產生的描述

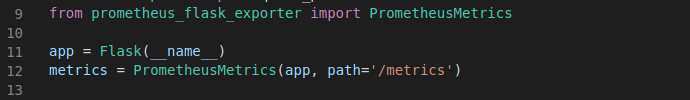
The unit test result:

一張含有 文字 的圖片

自動產生的描述

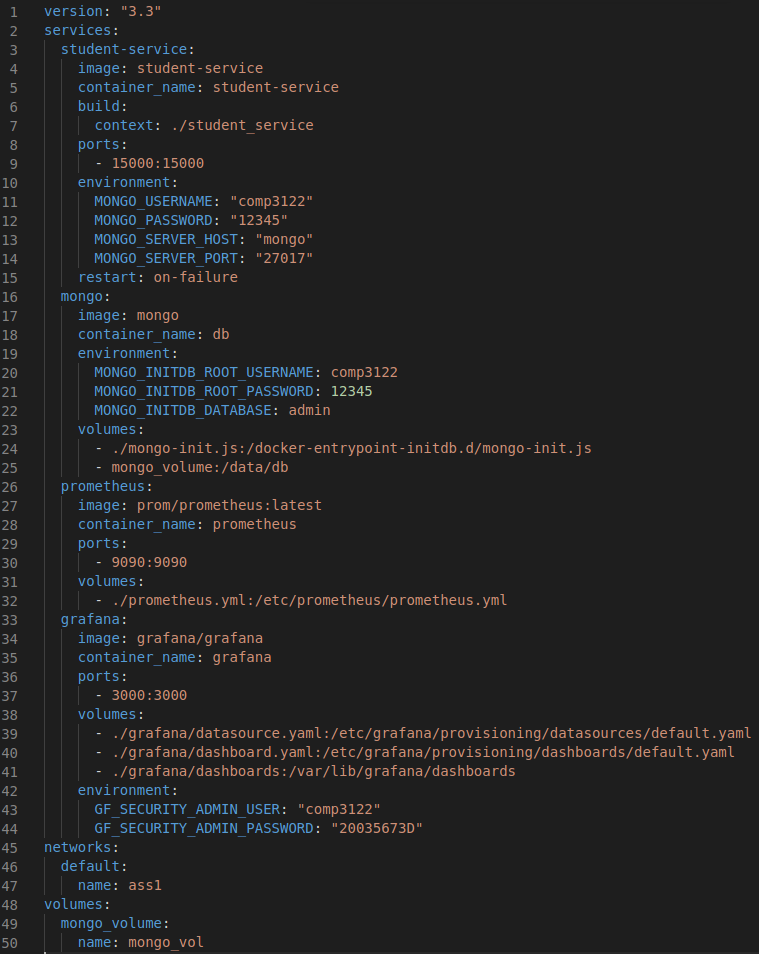
# Configuration files

Prometheus\_flask\_exporter in app.py



This exporter is used to export the application metrics for the various endpoint through the path /metrics.

Docker-compose.yaml



This docker-compose file set up four services including student-service, MongoDB, Prometheus and Grafana, a network named ass1 and a volume for MongoDB.

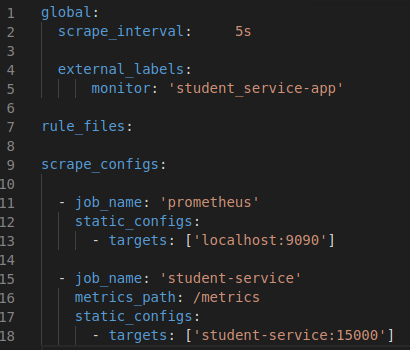
Usage of line 3 to line 15 setup the student service where line 8 and line 9 expose the service at the host port 15000 and the environment values are for accessing the MongoDB.

Usage of line 16 to line 25 setup the MongoDB where the environment values are for initializing and setting the database’s account, and line 24 is for copying the volume and a js file that store the initial record to the target directory.

Usage of line 26 to line 32 setup the Prometheus where the port is 9090 and line 32 is for copying the configuration file prometheus.yml to target directory.

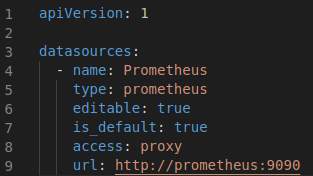
Usage of line 33 to line 44 setup the Grafana where the port is 3000 and the environment values are for initializing and setting the user account, and the volumes part copy the configuration file and dashboards to the target directory.

Prometheus.yml



This file set up the configuration of the Prometheus where line 2 set the interval to 5s and line 9 to line 18 set up two services named Prometheus and student-service.

datasource.yaml



This file set up the configuration of data sources of the Grafana that will make the Grafana configure the data sources automatically.

dashboard.yaml

一張含有 文字 的圖片

自動產生的描述

This file set up the configuration of the dashboard of the Grafana that will make the Grafana know where has the dashboards files, and use the mydashboard.json in the folder to preconfigured.

# Grafana dashboard panels

In the dashboard, there are 7 panels as follows.

一張含有 文字, 監視器, 室內 的圖片

自動產生的描述

The data will display last 5 minutes. And the refresh period is 5 second.



up service

一張含有 文字, 螢幕擷取畫面, 監視器, 黑色 的圖片

自動產生的描述

The “up service” panel indicates the number of services is available currently. This use PromQL “sum(up{})”. And the visualization method is Stat.

Failed HTTP Requests

一張含有 文字, 螢幕擷取畫面, 監視器 的圖片

自動產生的描述

This panel use the PromQL ” sum(flask\_http\_request\_total{status!~"2.."})” to show the number of failed HTTP requests. This PromQL sum the number of HTTP request which its status code does not begin at number 2. And the visualization method is Stat.

Successful HTTP Requests

一張含有 文字, 監視器, 螢幕擷取畫面 的圖片

自動產生的描述

This panel use the PromQL ” sum(flask\_http\_request\_total{status=~"2.."})” to show the number of successful HTTP requests. This PromQL sum the number of HTTP request which its status code begins at number 2. And the visualization method is Stat.

Total HTTP Request by Status Code[1m interval]

一張含有 文字, 監視器, 螢幕擷取畫面, 螢幕 的圖片

自動產生的描述

This panel use the PromQL ”increase(flask\_http\_request\_total[1m])” to display the total number of HTTP requests which is grouped by status code while the interval is 1 minute. Also, I set the legend as {{method}}:{{status}} for reference. And I use the transform to calculate the number of HTTP requests from different methods for different status codes and organize fields to hide some raw fields and rename some fields. And the visualization method is time series.

一張含有 文字, 監視器, 黑色, 螢幕 的圖片

自動產生的描述

一張含有 文字, 螢幕, 黑色, 黑暗 的圖片

自動產生的描述

一張含有 文字, 監視器, 電子用品, 螢幕 的圖片

自動產生的描述



HTTP Status Code

一張含有 文字, 監視器, 螢幕擷取畫面, 黑色 的圖片

自動產生的描述

This panel use the PromQL ”flask\_http\_request\_total” to display the instant total number of HTTP requests which is grouped by status code. The instant function is enabled. Also, I use the transform to calculate the number of HTTP requests and group the fields, which have the same status code and come from different methods together, and organize fields to hide and rename some fields. And the visualization method is table.

Successful HTTP Requests

一張含有 文字, 螢幕擷取畫面, 監視器, 黑色 的圖片

自動產生的描述

This panel use the PromQL ” flask\_http\_request\_duration\_seconds\_count{status=~"2.."}” to display the instant total number of successful HTTP requests from different method, status and path. The PromQL requires the data to have the status code beginning at 2. The instant function is enabled. Also, I use the transform organize fields to hide and rename some fields. And the visualization method is table.

一張含有 文字, 螢幕擷取畫面, 螢幕 的圖片

自動產生的描述

Failed HTTP Requests

一張含有 文字, 監視器, 螢幕擷取畫面, 黑色 的圖片

自動產生的描述

This panel use the PromQL ” flask\_http\_request\_duration\_seconds\_count{status!~"2.."}” to display the instant total number of failed HTTP requests from different method, status and path. The PromQL requires the beginning of the status code of the data is not 2.The instant function is enabled. Also, I use the transform organize fields to hide and rename some fields. And the visualization method is table.

