ACIT 3855 – Lab 2 – Edge Service and Testing

Instructor	Mike Mulder (mmulder10@bcit.ca)
Total Marks	10
Due Dates	Demo and Submission by End of the Lesson 3 Class

Purpose

- Build a Edge Service (Receiver Service) that receives your two event types and logs them to a file
- Testing of your Edge Service with PostMan and Apache jMeter

Part 1 – Stubbing Out the Service with Connexion

Reference: https://connexion.readthedocs.io/en/latest/

In your IDE (i.e., PyCharm, Visual Studio Code), create a new Python project. Call it **Receiver**. This will make it easier to know which service this represents later than calling it something like Lab2.

Install the following packages:

- connexion
- swagger-ui-bundle (needed for your app to generate a UI endpoint)

Copy the openapi.yml file containing your OpenAPI specification from Lab 1 into your project.

Create a basic connexion app called app.py:

```
import connexion
from connexion import NoContent

# Your functions here

app = connexion.FlaskApp(__name__, specification_dir='')
app.add_api("openapi.yml")

if __name__ == "__main__":
    app.run(port=8080)
```

For each event in your OpenAPI specification, create a function as follows:

- Named based on the operationId
- By default, the parameter for the requestBody data of an endpoint must be named as **body**
- Prints the parameter to the console (i.e., the raw Python dictionary)
- Returns a 201 response code (example: return NoContext, 201)

Run the connexion application. Go the following URL: http://localhost:8080/ui

You should see your API specification and are able to test it out.

Use PostMan to verify both of your service API endpoints.

Part 2 – Logging Your Event Data

Log your requests to a JSON file:

- Remove your print statements
- Replace them with a simple logging mechanism that writes the last 12 received requests to an events.json file.
    - The requests should be stored in json format, as an array of objects
    - New requests written to the file are added to the end of the array
    - Once 12 requests are written to the events.json file, the oldest request in the array should be removed before the new request is added so that the length of the array does not exceed 12
    - There should only be one events.json for both request types.

  Note: You could create a new function that takes in the request body and writes it to the file.
- Use a constant to define the maximum number of events to store (i.e., MAX_ EVENTS)
- Use a constant to define the filename where the events are stored (i.e., EVENT_FILE)

The contents of the JSON file will look something like this (with the event data matching that of your system):


```
[
    {
        "device_id": "A12345",
        "heart_rate": 109,
        "patient_id": "502594",
        "timestamp": "2016-08-29T09:12:33.001000+00:00"
    },
    {
        "device_id": "A12345",
        "heart_rate": 73,
        "patient_id": "615272",
        "timestamp": "2016-08-29T09:12:33.001000+00:00"
    },
    …

]
```


Use PostMan to verify both of your service API endpoints and the contents of the JSON file.

Part 3 – Strict Validation

Enable validation on the request and response of your API.

Add the following arguments to app.add_api:

- `strict_validation=True`
- `validate_responses=True`

Use PostMan to verify that your API endpoints still work and that invalid request messages are rejected.

Part 4 – Load Testing the Stubbed Service

Install Apache JMeter on your laptop (http://jmeter.apache.org/)

- Download and extract the latest binary. Note that you need Java 8 installed on your laptop.
- Run the .bat (Windows) or .sh (Linux) file from the bin folder

Load test your two API endpoints by creating a Test Plan with the following:

- Thread Group with X users and 10 loop count, where X is your expected peak load of concurrent events
- The Thread Group contains:
    - HTTP Request for your first API endpoint
    - HTTP Request for your second API endpoint
    - Listener -> View Results Tree

You can follow this tutorial as a reference:

- https://www.blazemeter.com/blog/rest-api-testing-how-to-do-it-right/

Make sure to set the headers for your HTTP Requests to application/json (see https://www.testingexcellence.com/jmeter-tutorial-testing-rest-web-services/)

Make sure all your HTTP Requests are successful in the View Results Tree. You may have to adjust your X value (i.e., peak load) depending on the number of concurrent events your service can support.

Demostration and Submission

Demonstrate the following to your instructor.

app.py file with two functions corresponding to your POST API endpoints that log the most recent requests for each to JSON files as per the specifications above *You will only get the full 4 marks if all the requirements are met – correct logging, contents of events.log is valid JSON, using constants, etc.*	4 marks
The http://localhost:8080/ui shows the documentation for your two endpoints	2 marks
Valid requests are successfully received by end of your endpoints and invalid are rejected	2 marks
Load testing demo using Apache jMeter	2 marks
Total	**10 marks**

Upload your app.py and openapi.yaml files to the Lab 2 dropbox in the Learning Hub as your submission to receive the marks for your demo.