ACIT 3855 – Lab 3 – Data Storage Service and Synchronous Communication

Instructor	Mike Mulder (mmulder10@bcit.ca) – Sets A and C Tim Guicherd (tguicherd@bcit.ca) – Set B
Total Marks	10
Due Dates	Demo and Submission by End of the Lesson 3 Class: • Monday, Jan. 29th for Set C • Thursday, Feb. 1st for Sets A and B

Purpose

- Create and test the API specification and implementation for a second service that stores events in a database. This service follows the "database per service" pattern.
- Integrate the first service from Lesson 2 (the Receiver service) with today's second service (the Storage service). This integration uses synchronous communication through a RESTful API.

Part 1 – Data Storage Service (Database Per Service Pattern)

Use the ACIT3855_Lab3_Sample_Code.zip as a reference.

- Create a new project for Lab 3 in your IDE called Storage, but also keep the project for Lab 2 (Receiver service) open as you will need it for Part 2 and Part 3.
- Copy over the openapi.yml and app.py files from Lab 2 (Receiver) into the Lab 3 Storage project as your starting point.
- You'll need to install the following packages:
  - connexion
  - swagger-ui-bundle
  - sqlalchemy
- We will leave the openapi.yml as-is for this lab. We will add to it in a later lab to allow the stored events to be retrieved.
- Modify the app.py as follows:
  - Change the port used for this service. The Receiver Service should use port 8080 and the Data Storage Service port 8090. This allows them to both run concurrently on your laptop without conflicting on ports.
- Create a create_database.py script to create two database tables, one for each of your events, in a SQLite database.
- Create a SQLAlchemy declarative for each of the events that map to your database tables. In addition to the attributes you defined, it should also have a date_created column that is set to the current datetime when the record is added to the database. See the sample code for an example.
- Modify the app.py as follows:
  - Your POST methods should create an instance of the event to add using the SQLAlchemy declarative and that object should be added and comitted to the database session. Remember to create and close the session.
- Test out the API endpoints of your new service using:
  - PostMan

  - DB Browser for Sqlite – to check if the records are actually stored. You can download it here if you don't already have it: https://sqlitebrowser.org/

Part 2 – Integration with Receiver Service (Synchronous Communication)

Here you will be modifying your Lab2 code as follows:

- Install the Python requests package and make sure to import it into app.py
- In each of your POST methods, use requests.post function to call the POST endpoints on your new Data Storage Service on port 8090. Pass the event data (i.e., the body parameter) as the json payload in the request and make sure to set the Content-Type in the headers to application/json. Return the NoContent and the status_code value received from the Data Storage Service from these methods.
  Note: You should remove the code that writes your request to the JSON file so it doesn't impact our performance anymore.
- Test the POST API endpoints of your Receiver Service with PostMan and DB Browser for Sqlite to make sure the synchronous communication works and the records are stored.

Part 3 – Load Testing

Update your jMeter script as follows (***if you haven't already done this in Lab 2***):

- Currently your HTTP requests send the same data each time so your database will be populated with identical requests.
- Use one of the methods defined at the URL below to add random data to your HTTP requests in your jMeter script:
  - https://www.blazemeter.com/blog/three-ways-to-generate-random-variables-in-jmeter/
  - For example, I used the Random Variable Config Element to define values for heart rate, systolic and diastolic blood pressure for my sample application.
  - Alternately (and perhaps better) would be to use the random functions here: https://jmeter.apache.org/usermanual/functions.html (specifically Random, RandomDate, RandomString, UUID)
  - For UUID, you could also look into creating a spreadsheet of values and loading those into jMeter

When running your jMeter script against your updated Receiver Service, you make see some errors in the results and on the Data Storage Service console. Why might Sqlite not be a good choice of database for a system that will be under high load such as the one you are building? Why might MySQL be a better choice?

Grading and Submission

Demonstrate the following to your instructor before the end of the next class.

Data Storage Service Code (1 mark each) • openapi.yaml file • POST – event1 that persists to the DB • POST – event2 that persists to the DB • SQLAlchemy declarative for event1 • SQLAlchemy declarative for event2 • Sqlite database creation script	6 marks
Demo of Data Storage Service, with either PostMan or the UI endpoint, for both endpoints and DB Browser for SQLite to show the data.	1 marks
Demo of integration of the Receiver Service with the Data Storage Service through the jMeter load test • Must have concurrent threads • Events must have unique values	3 marks
Total	**10 marks**

Upload the code for your Storage Service (as a zipfile) to the Lab 3 dropbox to receive the marks for your demo.