ACIT 3855 – Lab 1 – Project Description and RESTful API

Instructor	Mike Mulder (mmulder10@bcit.ca) – Sets A and C Tim Guicherd (tguicherd@bcit.ca) – Set B
Total Marks	10
Due Dates	Due midnight prior to the Lesson 2 class: • Sunday, Jan. 14th at midnight for Set C • Wednesday, Jan. 16th at midnight for Sets A and B

Purpose

- Select a sample software project for the course and describe how it will meet the minimum requirements.
- Create the API specification for the first service.

Software Project Selection

Write a description of your project software for the course, including:

- One or two sentences describing the purpose of the software.
- The events that are received, stored and processed by your software system. There should be two types of events received.
- Define a peak number of concurrent events that may be received by your software system. This can just be a guesstimate.
- Who are the different users of the system? Identify and describe each of them.

Follow the pattern in the examples provided in the class lecture slides.

Service API Specification

The first service you will be creating is one that receives "events" from other software applications (i.e., mobile apps, web apps, IoT devices) through a RESTful API. Assume that the "events" will be received regularly with spikes at certain periods of the day where the service will experience high load.

This lab you will be using OpenAPI 3.0 to create the RESTful API specification of the service to receive your events. Next lab you will be using the Python Connexion framework (built on top of Flask) to create the service and PostMan and Apache jMeter to test the service.

Part 1 – OpenAPI Specification

Create an account on SwaggerHub:

https://swagger.io/tools/swaggerhub/

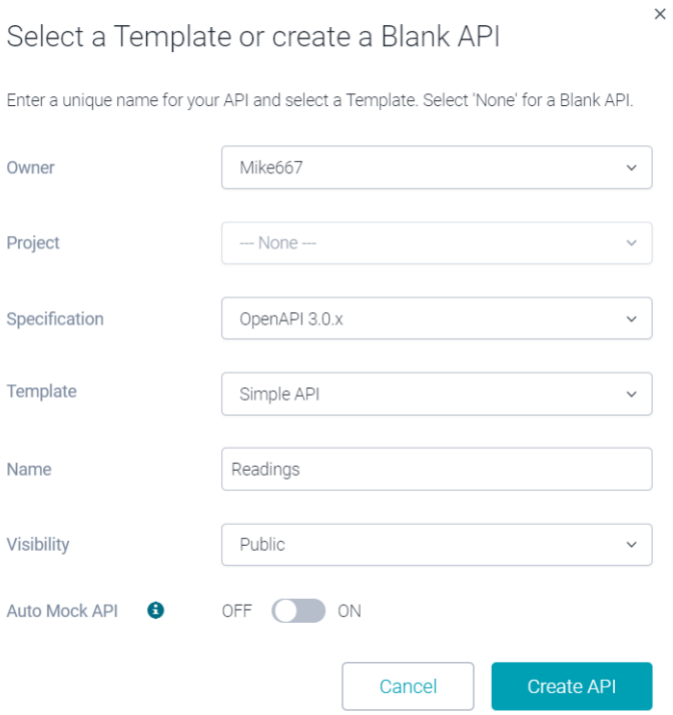
Create an OpenAPI 3.0 specification for the RESTful that receives your events. It should have the following:

- Two POST API endpoints to receive each of the two types of events.
- Each event should have several properties (4 or more)
    - At least one property must be numeric

- One property must be a Universally Unique Identifier (UUID) that represents the ID of the user or device sending the event. So, this UUID would be the same for all events sent from the same user or device.

Once logged in to SwaggerHub:

- Select Create New -> Create New API
- Fill out the form that pops up to create an OpenAPI 3.0.x API. Below is an example, but give it a name relevant to your project.
- ***Note: Turn OFF the Auto Mock API otherwise it will insert fake URLs that will cause troubles next lab.***



- You will get a stubbed out OpenAPI specification. Modify it to have the two POST endpoints for your two events. Remove any unneeded content. See the annotated example at the end of this lab. Your request message for each endpoint should be defined in one or more objects.
- SwaggerHub will show errors if your API specification is invalid. Fix any errors.
- SwaggerHub will automatically generate documentation for your API specification (typically on the left side of the page). This document should accuractly describe the two POST endpoints in your API.

If you want more information, here is a tutorial on OpenAPI 3.0:
https://app.swaggerhub.com/help/tutorials/openapi-3-tutorial

In Week 3 we will do more work with OpenAPI 3.0 for your next service.

Grading and Submission

Submit the following to the Lab 1 Dropbox on D2L:

- A text file with the description of your software project
- Your openapi.yml file

Project description with the required details • Description (1 mark) • Two Events (1 mark) • Peak Events (1 mark) • Users with descriptions (1 mark)	4 marks
openapi.yaml file: • Resource • Two POST endpoints • **At least four unique properties per endpoint. There must be at least one numeric and one UUID property per endpoint.** • Successfully validates in SwaggerHub with valid documentation All of the above must be present to get full marks on the openapi.yml file, otherwise the mark is zero.	6 marks
Total	**10 marks**

Sample OpenAPI Specification

```
openapi: 3.0.0
info:
  description: This API receives reading events from medical devices
  version: "1.0.0"
  title: Reading API
  contact:
    email: mmulder10@bcit.ca
tags:
  - name: devices
    description: Operations available to medical devices
paths:
  /readings/blood-pressure:
    post:
      tags:
        - devices
      summary: reports a blood pressure reading
      operationId: app.report_blood_pressure_reading
      description: Adds a new blood pressure reading to the system
      responses:
        '201':
          description: item created
        '400':
          description: 'invalid input, object invalid'
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/BloodPressureReading'
        description: Reading item to add


/readings/heart-rate:
    post:
      tags:
        - devices
      summary: reports a heart rate reading
      operationId: app.report_heart_rate_reading
      description: Adds a new heart rate reading to the system
      responses:
        '201':
          description: item created
        '400':
          description: 'invalid input, object invalid'
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/HeartRateReading'
        description: Reading item to add

components:
  schemas:
    BloodPressureReading:
      required:
      - patient_id
```

Info on your API

(i.e., metadata)

Defines tags. They can be used to categorize your API Endpoints

Endpoint for Blood Pressure Readings

operationId: Corresponds to the name of the module (i.e., app) and function (i.e., report_blood_pressure_reading) in your Python app that will handle this endpoint

All possible response codes from the endpont

References the object that defines the JSON request message

Endpoint for Heart Rate Readings

Defines a Blood Pressure Reading object

```
  - device_id
  - blood_pressure
  - timestamp
  properties:
    patient_id:
      type: string
      format: uuid
      example: d290f1ee-6c54-4b01-90e6-d701748f0851
    device_id:
      type: string
      example: A12345
    blood_pressure:
      $ref: '#/components/schemas/BloodPressure'
    timestamp:
      type: string
      format: date-time
      example: '2016-08-29T09:12:33.001Z'
  type: object

HeartRateReading:
  required:
  - patient_id
  - device_id
  - heart_rate
  - timestamp
  properties:
    patient_id:
      type: string
      format: uuid
      example: d290f1ee-6c54-4b01-90e6-d701748f0851
    device_id:
      type: string
      example: A12345
    heart_rate:
      type: integer
      example: 85
    timestamp:
      type: string
      format: date-time
      example: '2016-08-29T09:12:33.001Z'
  type: object

BloodPressure:
  required:
  - systolic
  - diastolic
  properties:
    systolic:
      type: integer
      example: 120
    diastolic:
      type: integer
      example: 80
  type: object
```

Defines the key/value pairs in the JSON

This is a nested object. It is defined below.

Defines a Heart Rate Reading object

Used by the BloodPressureReading object.