Assignment 2

CSCI 5410 (Serverless Data Processing)
Date Given: May 29, 2020
Due Date: Jun 10, 2020 at 11:59 pm
Late Submissions are not accepted and will result in scoring "0" in the assignment.

To avoid any additional charges for resource consumption - Delete any GCP storage, database, and AWS Service after fulfilling the assignment submission requirements

---

Objective:
This assignment covers concepts of containerization and Serverless components of cloud computing. The primary objective of this assignment is to introduce you to the cloud computing containerization application using Docker and creation of a chatbot using Lex.

Plagiarism Policy:

- This assignment is an individual task. Collaboration of any type amounts to a violation of the academic integrity policy and will be reported to the AIO.
- Content cannot be copied verbatim from any source(s). Please understand the concept and write in your own words. In addition, cite the actual source. Failing to do so will be considered as plagiarism and/or cheating.
- The Dalhousie Academic Integrity policy applies to all material submitted as part of this course. Please understand the policy, which is available at: https://www.dal.ca/dept/university_secretariat/academic-integrity.html

Grading Scheme:

A. Containerized Application Development: 40 points
    a. Completeness of tasks: 30 points, (if all parts are completed. points **a.** to **h.**)
    b. Testing: 5 points (Write test cases, and perform testing point **i**.)
    c. Quality of document: 5 points (sentence formation, categorization, grammatical errors etc. – point **j.**)
B. Building Chatbot: 10 points
    a. Completeness of tasks: 8 points (if all parts are completed)
    b. Quality of document: 2 points (sentence formation, categorization, grammatical errors etc.)

Tasks:
This assignment has 2 parts. Part A is related to coding, and development. Part B is related to exploring a service.

Part A. Build, deploy, and run a Containerized Application using GCP.

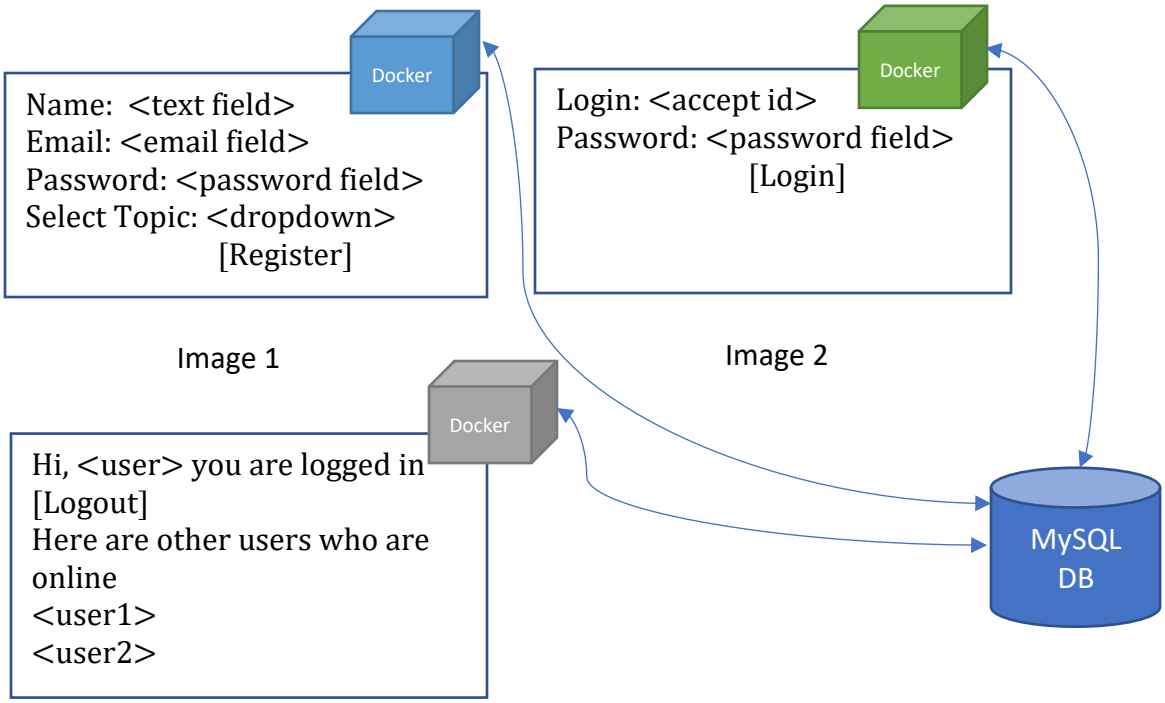
Using GCP create and validate an online meeting account.

take screenshots at every step and submit as part of the PDF:

a. Create three containers using **Docker**. These containers are responsible for the backend logic. The database you will be using here is, **MySQL**
b. **Container #1** is responsible for accepting registration details from frontend and store it in backend database. (image 1)
c. **Container #2** is responsible for validating the Login information (image 2)
d. Once a user is logged in – the state changes to online, and it appears on the front page (image 3)
e. Your database should contain only 2 tables. One to contain data, another to contain user state (online, offline, timestamp etc.) information.
f. **Container #3** is responsible for extracting state information from database. E.g. who is online. You need to maintain the session from login to logout. The session must expire after clicking the logout, which should update the state table.
g. Once the docker images are built, you need to run those using Google **Cloud Run**.
h. In order complete the tasks, and perform interaction, you need build 3 simple web pages (or 1), using any technology of your choice.

i. Write test case to test your application, and perform testing

j. You need to explore, Google Cloud Run, GCR, Docker Container documents, and write a summary of ½ page explaining how you have used these technologies in your application.



Part A - Submission requirement:

For (a to h), submit screenshots of every steps. Please do not exclude any steps.
For (a to h), submit one SQL file (export SQL structure and data)
For (a to h), submit your program files (Source code) as well.
However, from the source code - add the important methods, or program instructions as part of the pdf. E.g. Login Validation method etc.
For (i), add the test cases and screenshots of tests in the pdf
For (j), add the summary as part of the pdf file

Part B. Building a Chatbot:

Using AWS Educate account perform the following:
take screenshots at every step

a. Using AWS Lex - Create a chatbot on OrderFood
b. Consider it as a pizza place. (assumptions: they have 3 types of regular size pizzas –veg, cheese, pepperoni.
c. The chatbot can accept information on food **delivery** or **pickup**
d. If it is delivery, then customer address, delivery date, and time is important
e. If it is takeaway, then assuming same day, it should ask arrival time of customer.

E.g.
Utterances – "I want to place an order for pickup"
Prompts - "When are you coming to get your parcel?"
Slots – "I will come around noon"
Prompts – "What do you want today?"
Slots – "cheese pizza"
Prompts – "How many?"
Slots – "2"
Fulfillment –
"You have ordered 2 regular cheese pizza, and you will be arriving at 12:00 pm"
"Yes"
"Your order has been placed successfully"

Part B - Submission requirement:

A pdf file with the screenshots of AWS Lex chatbot creation, customization, test etc.
Paragraph explaining how the operation is performed.