Course Listing Microservice (REST API)

With
MySQL database &
Deployment of MySQL database to Docker

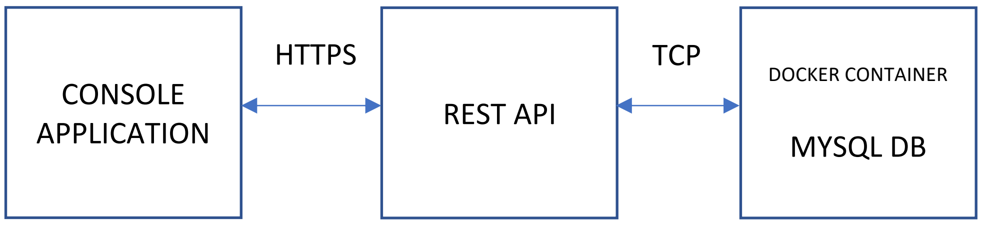
Created by
Oon Xin Ni
(Batch 2)

Go Microservice1 Assignment

Table of Contents

1. Overview

For this assignment, an REST API with CRUD (Create, Read, Update and Delete) features has been developed. Based on the request that the REST API received from console application (client), the API will process and send query to MySQL database to perform necessary action. In the console application, request is composed/built from the user inputs and sent to the REST API. After receiving the response from the REST API, it will feedback the necessary information back to the user.



2. Course Listing Microservice Components

2.1 REST API

Besides building the REST API to meet the basic assignment requirement of CRUD operation, the following considerations have been added in to improve the microservice.

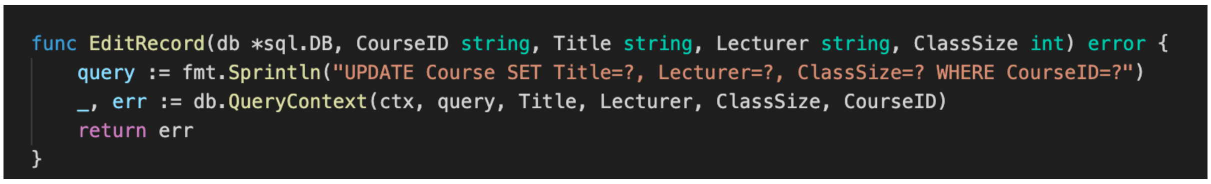
1. All requests from the console application will be validated using REGEX to ensure only inputs with desired format is accepted.

2. Sanitization of the request has also been done using bluemonday, third party packages.

3. To prevent expose of the access key and database information in the source code, these information were stored in .env files. These files should not be uploaded if we were to upload our API online. The file has been included in .gitignore.

4. HTTPS connection has been setup for the communication between console application (client) and REST API (server) to ensure an encrypted communication.

5. Besides that, to prevent possible SQL injection, all the SQL queries are setup with prepared statement. The parameter will be added in the later stage.



```go
func EditRecord(db *sql.DB, CourseID string, Title string, Lecturer string, ClassSize int) error {
    query := fmt.Sprintln("UPDATE Course SET Title=?, Lecturer=?, ClassSize=? WHERE CourseID=?")
    _, err := db.QueryContext(ctx, query, Title, Lecturer, ClassSize, CourseID)
    return err
}
```

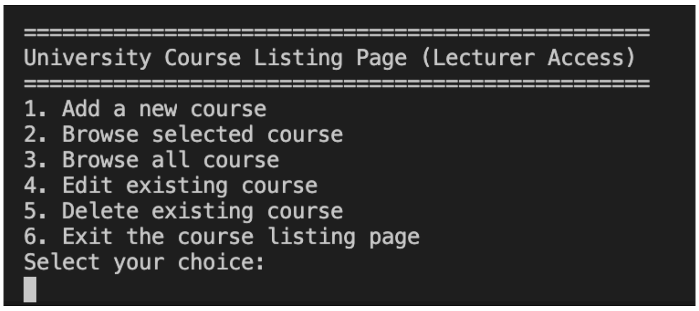
6. Logging feature has also been setup to log all the important event and unusual error/activities using third party packages, logurus.

2.2 MySQL Database

As per assignment requirement, MySQL database has been setup and deploy to Docker container. The database that has been deployed to Docker should persist as long as the container do not get deleted. For the MySQL database that has been created, courseID was set as the Primary key of the Course table, the field of course ID cannot be left empty. Besides that, the rest of the field has been setup with the validation as below:

1. CourseID field accept a max length of 7 of VARCHAR
2. Title field accepts a max length of 30 VARCHAR
3. Lecturer field accepts a max length of 30 VARCHAR
4. ClassSize field accepts only INTEGER

2.3 Client Console Application



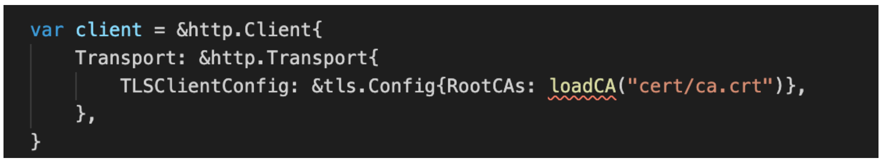
```
=================================================
University Course Listing Page (Lecturer Access)
=================================================
1. Add a new course
2. Browse selected course
3. Browse all course
4. Edit existing course
5. Delete existing course
6. Exit the course listing page
Select your choice:
```

For client console application, it mainly use to allow the user to select whatever operation they desired. The application will only be exited once option 6 is selected else the menu will be presented to the user once one of the CRUD operation ended. Besides the basic requirement, the following considerations has been added to the application.

1. Validation and sanitization of user inputs is especially important at the client side as we would not want to insert those unsecured data into our request sent to the server.

2. Logging has also been added to record important activities and errors.

3. Lastly the console application will be trying to communicate with the REST API through HTTPS connection. http. Client has been loaded with customised root certificate authority(CA) that will be used to verify the server certificate that has been generated through Open SSL.



```
var client = &http.Client{
    Transport: &http.Transport{
        TLSClientConfig: &tls.Config{RootCAs: loadCA("cert/ca.crt")},
    },
}
```

3. Setup Guide for Application Deployment

3.1 Deployment of MySQL database to Docker

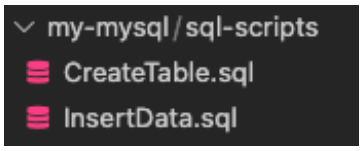
To run the microservice, we have to first setup the MySQL Container to store the existing database. In order to do this, the machine that is running this microservice has to have the following:

1. Docker installed.

2. goMicroService1Assignment folder should be placed under your /usr/local or home directory

    a. *This step is important to ensure the database is loaded properly and mount bind to the docker container. If the folder is not placed in the directory requested above, please update the path of the directory (highlighted in red below) accordingly.*

    b. Below folder is where all the data are stored.



3. Open your terminal and run the code below.

    docker run -d -p 54812:3306 --name goMS1-mysql -v ~/goMicroService1Assignment/RESTAPI/my-mysql/sql-scripts:/docker-entrypoint-initdb.d/ -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=my_db_goMicroservice1 mysql:latest

4. After the docker container is created, please run the following to setup the TCP connection to MySQL database in docker container.

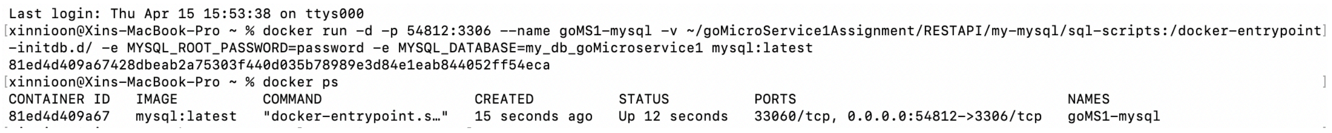
    mysql -P 54812 --protocol=tcp -u root -p

5. Please enter "password" for the password requested.

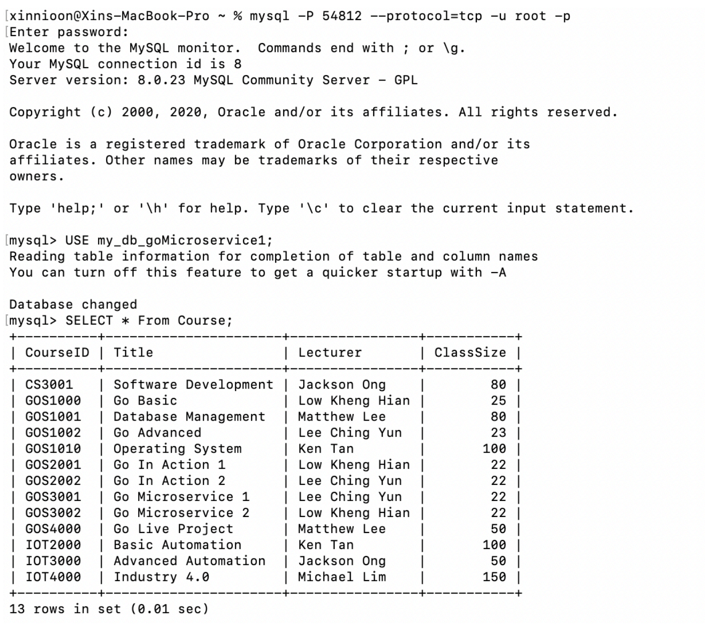
6. You should successfully access to your MySQL server after this. To test it, you could run the following SQL query.

    a. USE my_db_goMicroservice1;
    b. SELECT * From Course;

7. If you are able to see the table as below, you have successfully deployed MySQL database to docker container.



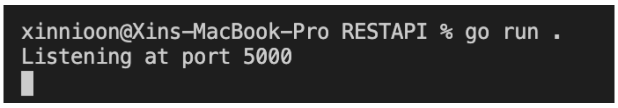
```
Last login: Thu Apr 15 15:53:38 on ttys000
xinnioon@Xins-MacBook-Pro ~ % docker run -d -p 54812:3306 --name goMS1-mysql -v ~/goMicroService1Assignment/RESTAPI/my-mysql/sql-scripts:/docker-entrypoint
-initdb.d/ -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=my_db_goMicroservice1 mysql:latest
81ed4d409a67428dbeab2a75303f440d035b78989e3d84e1eab844052ff54eca
xinnioon@Xins-MacBook-Pro ~ % docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                NAMES
81ed4d409a67   mysql:latest   "docker-entrypoint.s…"   15 seconds ago   Up 12 seconds   33060/tcp, 0.0.0.0:54812->3306/tcp   goMS1-mysql
```



```
xinnioon@Xins-MacBook-Pro ~ % mysql -P 54812 --protocol=tcp -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.23 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE my_db_goMicroservice1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * From Course;
+----------+----------------------+---------------+-----------+
| CourseID | Title                | Lecturer      | ClassSize |
+----------+----------------------+---------------+-----------+
| CS3001   | Software Development | Jackson Ong   |        80 |
| GOS1000  | Go Basic             | Low Kheng Hian |       25 |
| GOS1001  | Database Management  | Matthew Lee   |        80 |
| GOS1002  | Go Advanced          | Lee Ching Yun |        23 |
| GOS1010  | Operating System     | Ken Tan       |       100 |
| GOS2001  | Go In Action 1       | Low Kheng Hian |       22 |
| GOS2002  | Go In Action 2       | Lee Ching Yun |        22 |
| GOS3001  | Go Microservice 1    | Lee Ching Yun |        22 |
| GOS3002  | Go Microservice 2    | Low Kheng Hian |       22 |
| GOS4000  | Go Live Project      | Matthew Lee   |        50 |
| IOT2000  | Basic Automation     | Ken Tan       |       100 |
| IOT3000  | Advanced Automation  | Jackson Ong   |        50 |
| IOT4000  | Industry 4.0         | Michael Lim   |       150 |
+----------+----------------------+---------------+-----------+
13 rows in set (0.01 sec)
```

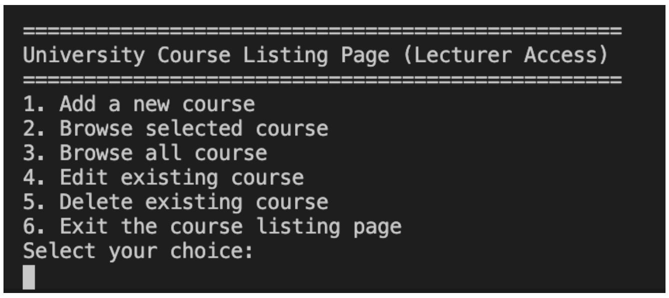
3.2 Running the Microservice

After deploying the database to docker container, you should open your REST API directory under goMicroservice1Assignment in your terminal, run the program with ***"go run ."*** command. The REST API should be start listening at its HTTPS connection at port 5000.



```
xinnioon@Xins-MacBook-Pro RESTAPI % go run .
Listening at port 5000
```

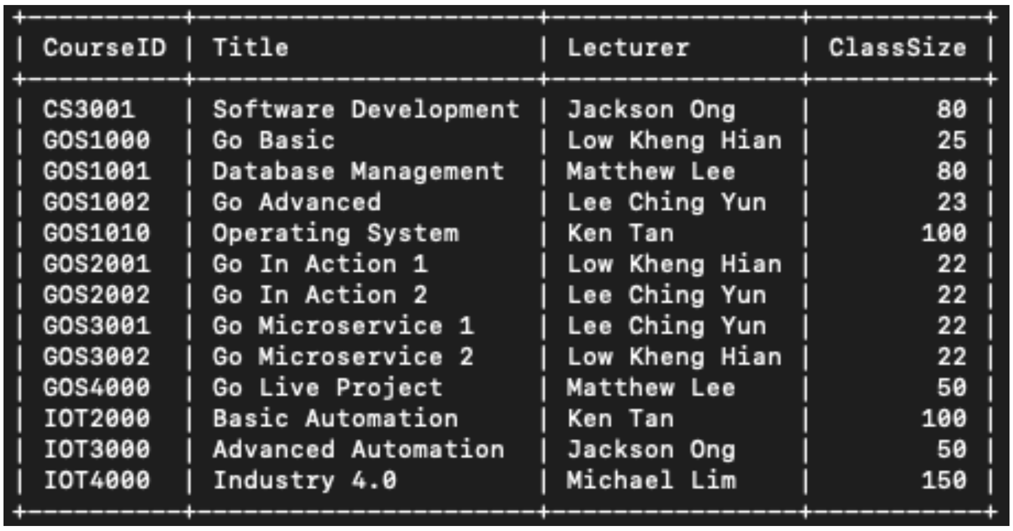
After that, open up another terminal to launch your console application. Open your consoleApplication directory and run the program with ***"go run ."*** command. From here, you are set to perform any CRUD operation that you desired.



```
================================================
University Course Listing Page (Lecturer Access)
================================================
1. Add a new course
2. Browse selected course
3. Browse all course
4. Edit existing course
5. Delete existing course
6. Exit the course listing page
Select your choice:
```

4. Test Cases

Below is the screenshot of the course data that has been preloaded to the MySQL Database. To test out the program, you can use test cases from 4.1-4.5



```
+----------+----------------------+----------------+-----------+
| CourseID | Title                | Lecturer       | ClassSize |
+----------+----------------------+----------------+-----------+
| CS3001   | Software Development | Jackson Ong    |        80 |
| GOS1000  | Go Basic             | Low Kheng Hian |        25 |
| GOS1001  | Database Management  | Matthew Lee    |        80 |
| GOS1002  | Go Advanced          | Lee Ching Yun  |        23 |
| GOS1010  | Operating System     | Ken Tan        |       100 |
| GOS2001  | Go In Action 1       | Low Kheng Hian |        22 |
| GOS2002  | Go In Action 2       | Lee Ching Yun  |        22 |
| GOS3001  | Go Microservice 1    | Lee Ching Yun  |        22 |
| GOS3002  | Go Microservice 2    | Low Kheng Hian |        22 |
| GOS4000  | Go Live Project      | Matthew Lee    |        50 |
| IOT2000  | Basic Automation     | Ken Tan        |       100 |
| IOT3000  | Advanced Automation  | Jackson Ong    |        50 |
| IOT4000  | Industry 4.0         | Michael Lim    |       150 |
+----------+----------------------+----------------+-----------+
```

4.1 Add Course (Option 1)

Test Case 1: **(PASS)**

a. CourseID: CSS4000
b. Title: Design Pattern
c. Lecturer: Matthew Lee
d. Class Size: 50

Test Case 2: **(FAIL) – WRONG COURSE ID FORMAT**

a. CourseID: CST1000

Test Case 3: **(FAIL) – EXCEED COURSE TITLE ACCEPTABLE LENGTH**

a. CourseID: IOT1000
b. Title: Introduction to home automation

Test Case 4: **(FAIL) – EXCEED 9999 STUDENTS**

c. CourseID: IOT1000
a. Title: Home automation
b. Lecturer: Ken Tan
c. Class Size: 1000000

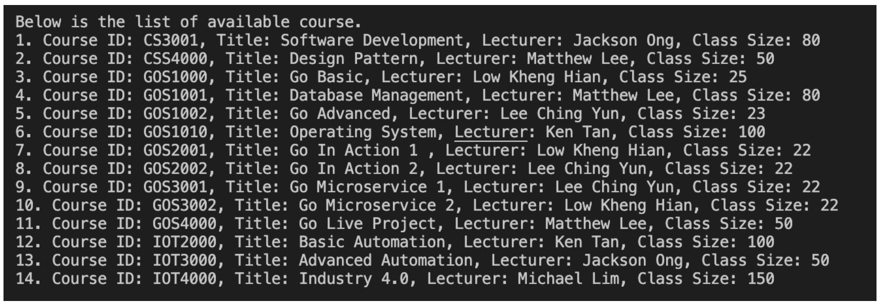
4.2 Browse Selected Course (Option 2)

Test Case 5: **(PASS)** - CourseID: GOS1001

Test Case 6: **(FAIL)** - CourseID: IOT5000

4.3 Browse All Course (Option 3)

Test Case 7: (PASS) – No input required. Expect to see whole list of courses.



```
Below is the list of available course.
1. Course ID: CS3001, Title: Software Development, Lecturer: Jackson Ong, Class Size: 80
2. Course ID: CSS4000, Title: Design Pattern, Lecturer: Matthew Lee, Class Size: 50
3. Course ID: GOS1000, Title: Go Basic, Lecturer: Low Kheng Hian, Class Size: 25
4. Course ID: GOS1001, Title: Database Management, Lecturer: Matthew Lee, Class Size: 80
5. Course ID: GOS1002, Title: Go Advanced, Lecturer: Lee Ching Yun, Class Size: 23
6. Course ID: GOS1010, Title: Operating System, Lecturer: Ken Tan, Class Size: 100
7. Course ID: GOS2001, Title: Go In Action 1 , Lecturer: Low Kheng Hian, Class Size: 22
8. Course ID: GOS2002, Title: Go In Action 2, Lecturer: Lee Ching Yun, Class Size: 22
9. Course ID: GOS3001, Title: Go Microservice 1, Lecturer: Lee Ching Yun, Class Size: 22
10. Course ID: GOS3002, Title: Go Microservice 2, Lecturer: Low Kheng Hian, Class Size: 22
11. Course ID: GOS4000, Title: Go Live Project, Lecturer: Matthew Lee, Class Size: 50
12. Course ID: IOT2000, Title: Basic Automation, Lecturer: Ken Tan, Class Size: 100
13. Course ID: IOT3000, Title: Advanced Automation, Lecturer: Jackson Ong, Class Size: 50
14. Course ID: IOT4000, Title: Industry 4.0, Lecturer: Michael Lim, Class Size: 150
```

4.4 Edit Course (Option 4)

Test Case 8: (PASS)

a. CourseID: IOT3000
b. Title: Robotics
c. Lecturer: *"Enter for no change"*
d. Class Size: *"Enter for no change"*

4.5 Delete Course (Option 5)

Test Case 9: (PASS) - CourseID: GOS1001