

**GO MICROSERVICES I (RUN BATCH # 4)**

**ASSIGNMENT SUBMISSION**

**DUE DATE: 26th MAY 2021 (WEDNESDAY) – 2359HRS**

|  |  |  |
| --- | --- | --- |
| **SUBMITTED BY** | **:** | **AMANDA SOH CHIEW PHENG** |

**Content Page**

[INTRODUCTION 4](#_INTRODUCTION)

[CLIENT 4](#_CLIENT)

[ADD COURSE 4](#_ADD_COURSE)

[UPDATE COURSE 5](#_UPDATE_COURSE)

[DELETE COURSE 6](#_DELETE_COURSE)

[RETRIEVE ALL COURSES 7](#_RETRIEVE_ALL_COURSES)

[HTTP/TLS 7](#_HTTP/TLS)

[CROSS SITE SCRIPTING 11](#_CROSS_SITE_SCRIPTING)

[REST API 13](#_REST_API)

[SECURING THE REST API USING ACCESS KEYS 13](#_SECURING_THE_REST)

[ADD COURSE 14](#_ADD_COURSE_1)

[UPDATE COURSE 14](#_UPDATE_COURSE_1)

[DELETE COURSE 15](#_DELETE_COURSE_1)

[GET COURSE/GET ALL COURSES 15](#_GET_COURSE/GET_ALL)

[DATABASE 17](#_DATABASE)

[DEPLOYING MYSQL DATABASE ON A DOCKER CONTAINER 17](#_DEPLOYING_MYSQL_DATABASE)

[USING MYSQL DATABASE 19](#_USING_MYSQL_DATABASE)

[ADD COURSE 19](#_ADD_COURSE_2)

[UPDATE COURSE 20](#_UPDATE_COURSE_2)

[DELETE COURSE 20](#_DELETE_COURSE_2)

[GET COURSE/GET ALL COURSES 20](#_GET_COURSE/GET_ALL_1)

[SANITIZATION 22](#_SANITIZATION)

[ERROR HANDLING 22](#_ERROR_HANDLING)

[USING ERRORS.NEW 22](#_USING_ERRORS.NEW)

[USING PANIC 22](#_LOG.FATAL)

[LOG.FATAL 23](#_Toc72944134)

[HTTP RESPONSE CODE 23](#_HTTP_RESPONSE_CODE)

[GO DOCUMENTATION 24](#_GO_DOCUMENTATION)

[INSTRUCTIONS ON HOW TO DEPLOY THE APPLICATION 25](#_Toc72944137)

[INSTALL CREATED PACKAGES FOR *CLIENT* AND *SERVER* 25](#_INSTALL_CREATED_PACKAGES)

[IMPORT 3RD PARTY PACKAGES 26](#_IMPORT_3RD_PARTY)

[RUNNING THE APPLICATION 26](#_RUNNING_THE_APPLICATION)

[GUIDE TO TEST THE APPLICATION 28](#_GUIDE_TO_TEST)

[RETRIEVE ALL COURSES 28](#_RETRIEVE_ALL_COURSES_1)

[ADD COURSE 29](#_ADD_COURSE_3)

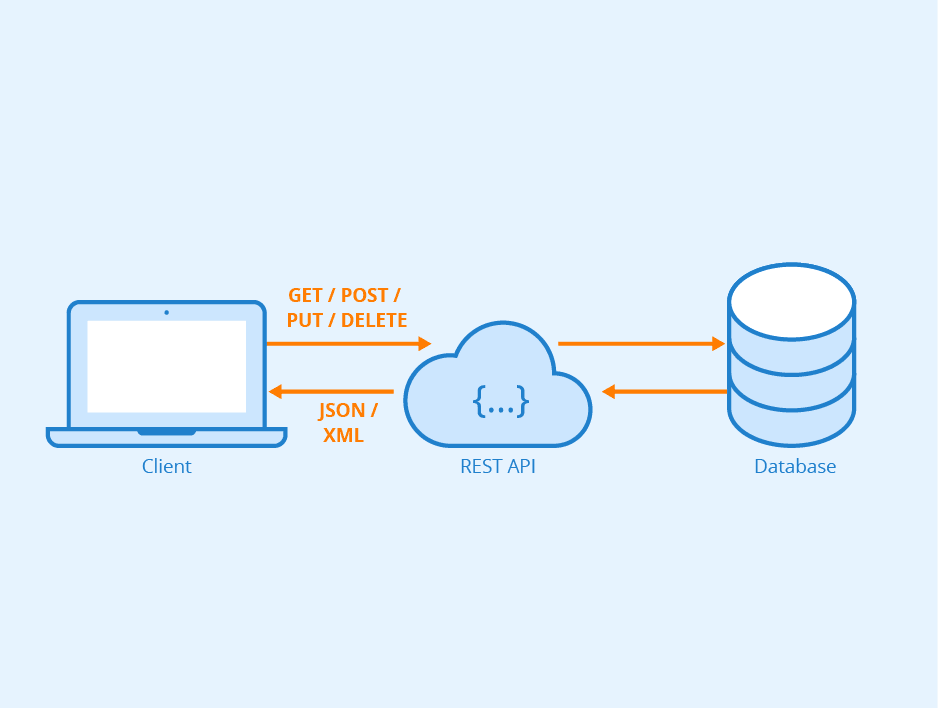
[UPDATE COURSE 31](#_UPDATE_COURSE_3)

[DELETE COURSE 33](#_DELETE_COURSE_3)

[REFERENCES 35](#_REFERENCES)

# INTRODUCTION

In this assignment, we are required to build a REST API using the Go language. The REST API would store information of courses. The details of the course will be stored in a MySQL database server. A web-based client is built to allow users to communicate with the REST API.



localhost:5000

localhost:55474

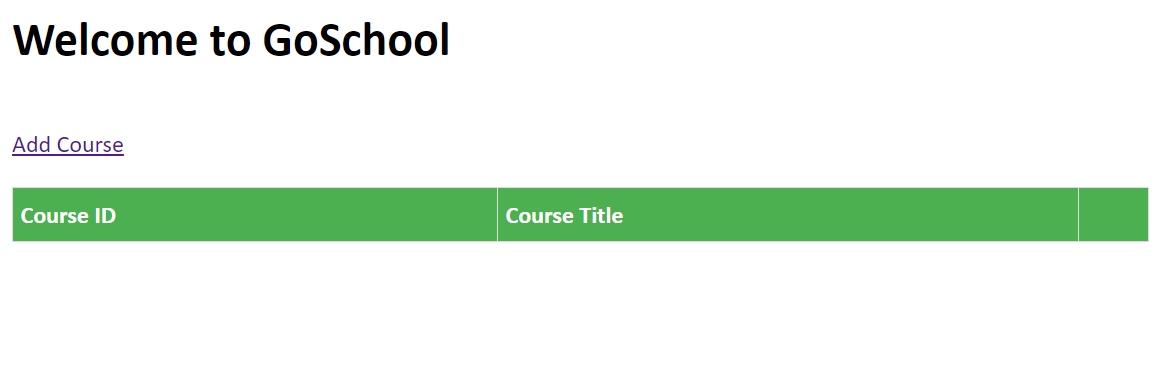
localhost:5221

# **CLIENT**

I have implemented the client as a web-based application which allows user to add course, update course, delete course and to retrieve all courses. The client is set to listen at port 5221 and is started at [*https://localhost:5221*](https://localhost:5221).

## **ADD COURSE**

The following is shown in the index page when the application is started. To add a course, click on the ‘Add Course’ hyperlink.

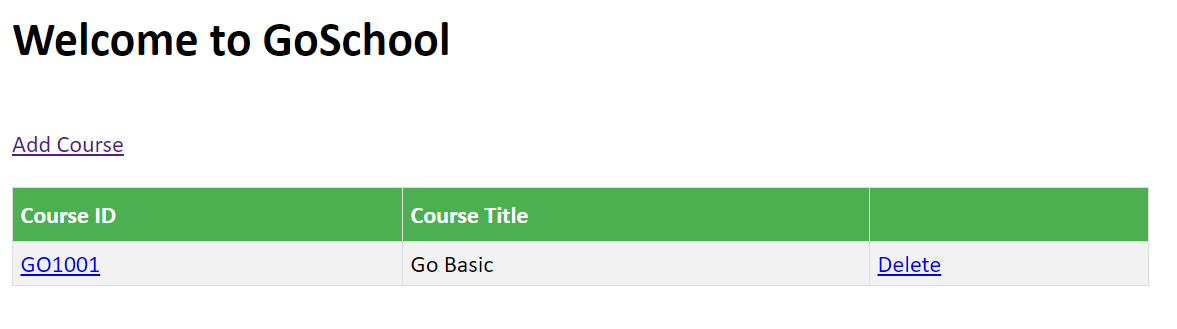
****

Enter course information to add a course and click on the ‘Submit’ button. The method “POST” is invoked to the REST API which then attempts to insert the record into the database table. A HTTP response code is received to determine if the action is successful and an appropriate message will be displayed to the user.



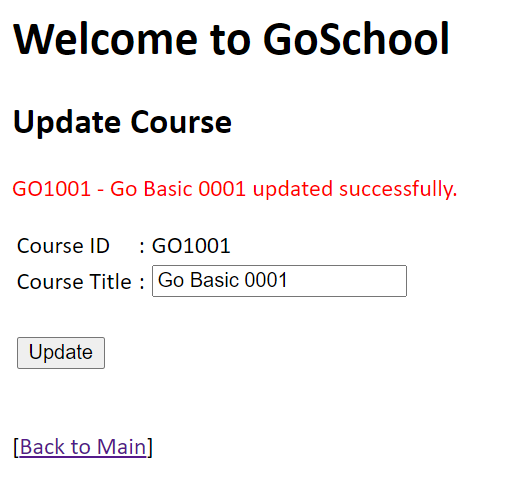
## **UPDATE COURSE**

When a course has been added, it will be displayed in the index page as shown. Click on the hyperlink on the ‘Course ID’ to go into the ‘Update Course’ page



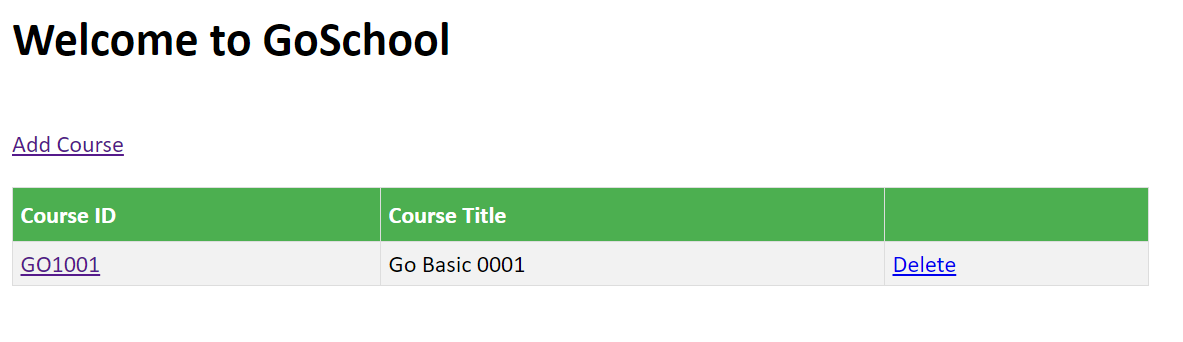
The course detail is retrieved and displayed whereby the method “GET” is invoked to the REST API which then attempts to retrieve the records from the database table. If records are found, a JSON data object will be sent back to the client. Upon un-marshalling the JSON, the course detail will be displayed.

For update, note that only the course title can be updated. Click on the ‘Update’ button to update the course. The method “PUT” is then invoked to the REST API which then attempts to update the record in the database table. A HTTP response code is received to determine if the action is successful and appropriate message will be displayed to the user.

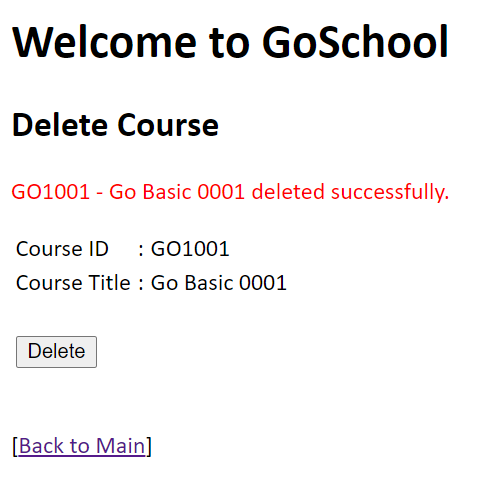
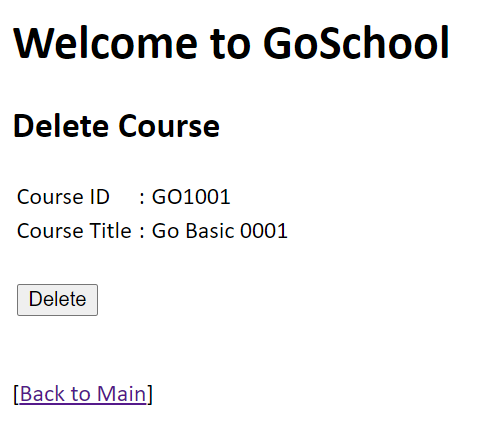


## **DELETE COURSE**

To delete a course, click on the ‘Delete’ hyperlink on the index page.

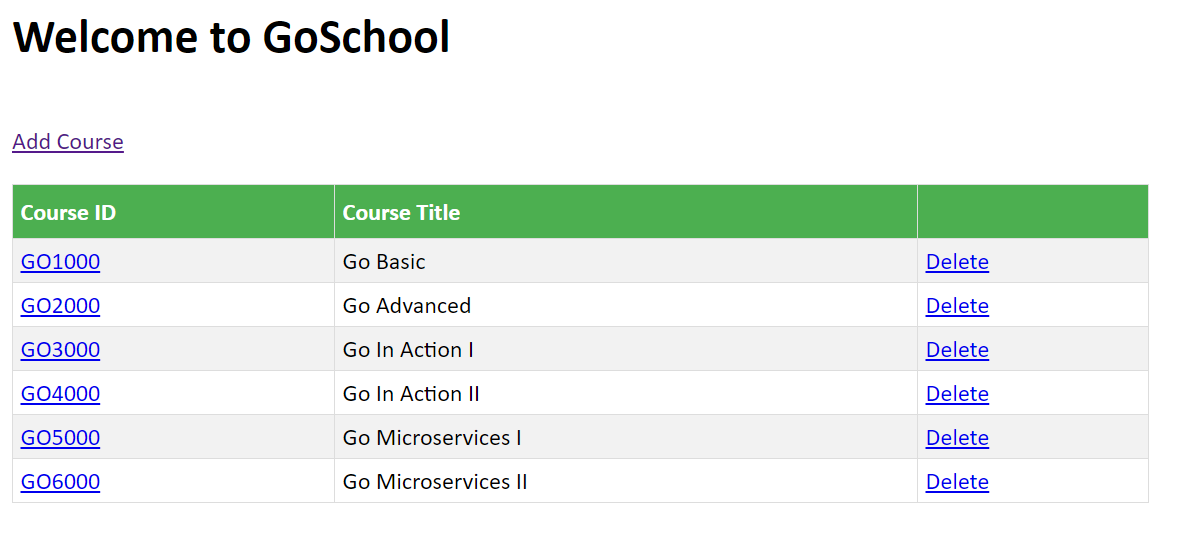


The details of the course is retrieved and displayed as per Update Course. Click on the ‘Delete’ button to delete the course. The method “DELETE” is invoked to the REST API which then attempts to delete the record in the database table. A HTTP response code is received to determine if the action is successful and appropriate message will be displayed to the user.



## **RETRIEVE ALL COURSES**

The index page retrieves all courses that have been created and displays them in a table format. As with Update/Delete Course, the method “GET” is invoked to the REST API which attempts to retrieve the records from the database table. If records are found, a JSON data object will be sent back to the client. Upon un-marshalling the JSON, all the course detail will be displayed.

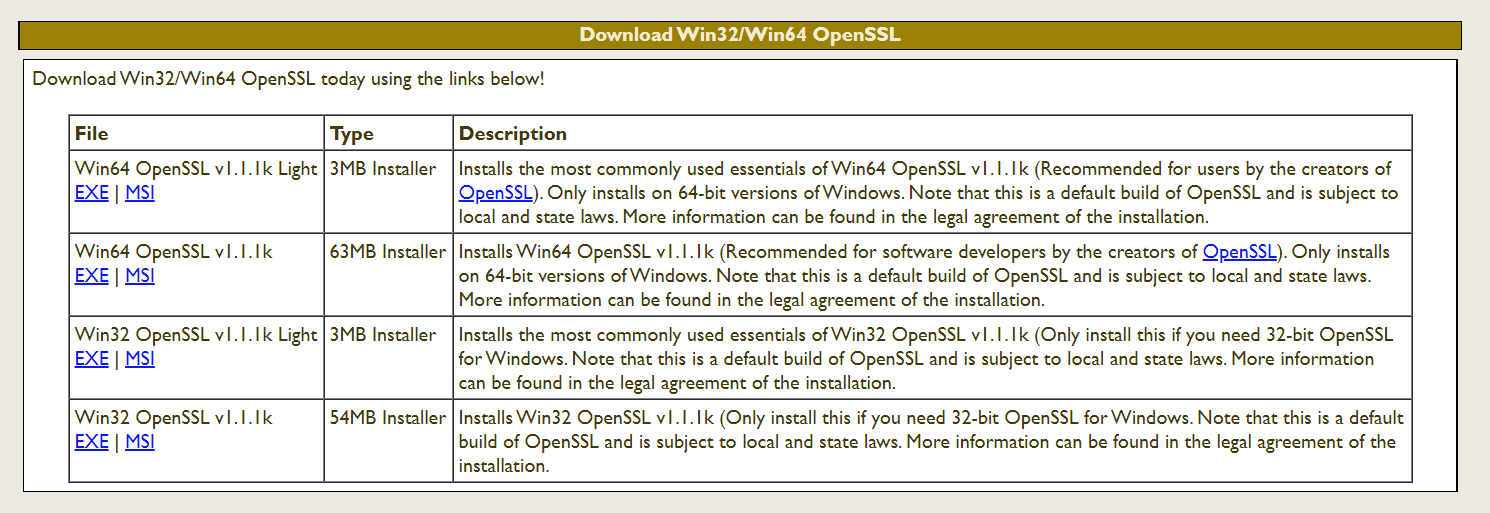


## **HTTP/TLS**

To ensure data integrity, and protect against common attacks related to communication security, this application uses ***HTTPS*** to encrypt and protect the communications between client and server. For testing purposes, OpenSSL is used to generate a certificate and key for establishing a handshake between the server and client to agree on a single session key to encrypt their packets.

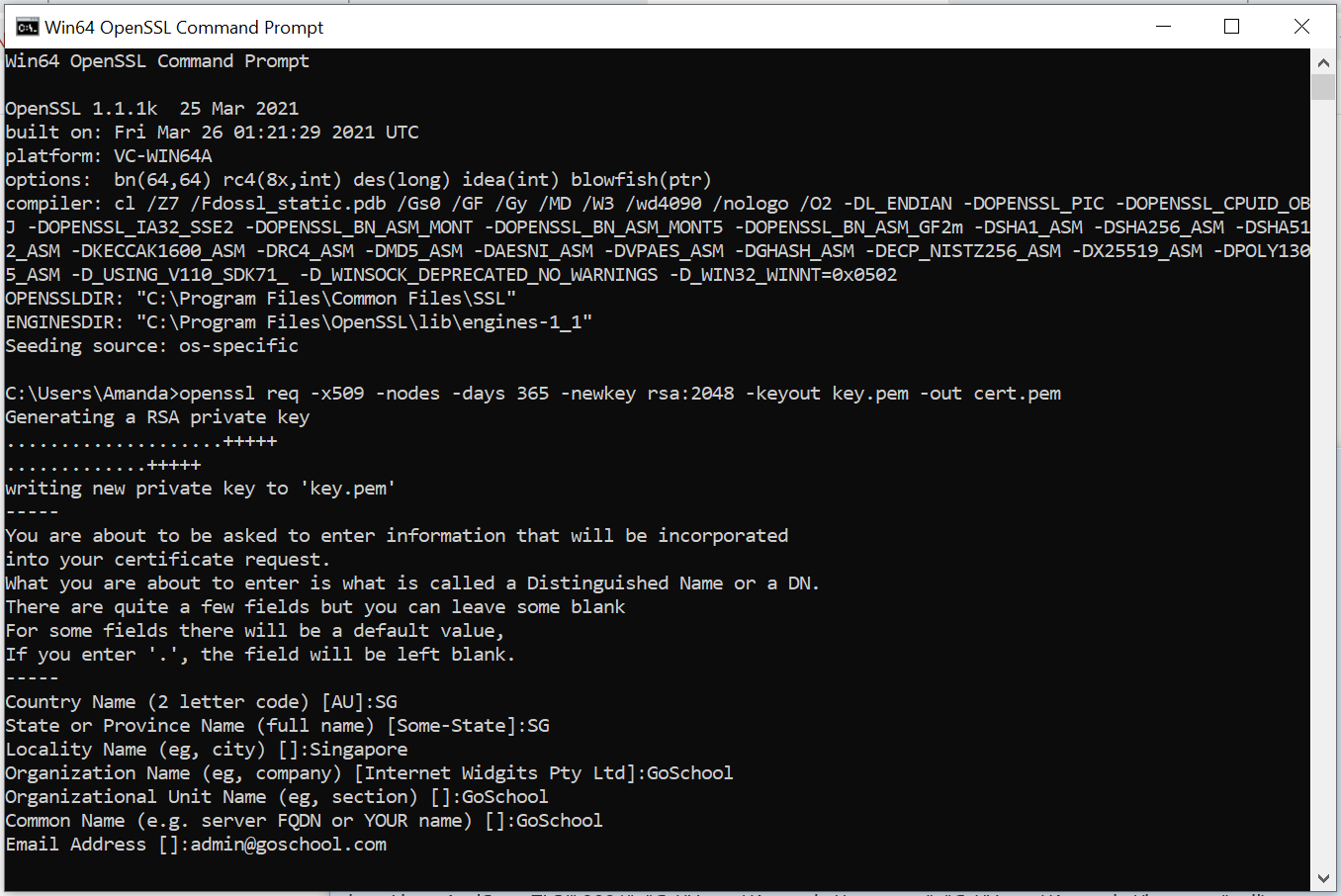
Note: The OpenSSL certificates is only used for development and testing. A CA certified SSL certificate is needed for production.

To use OpenSSL, we have to download and install it from <https://slproweb.com/products/Win32OpenSSL.html>

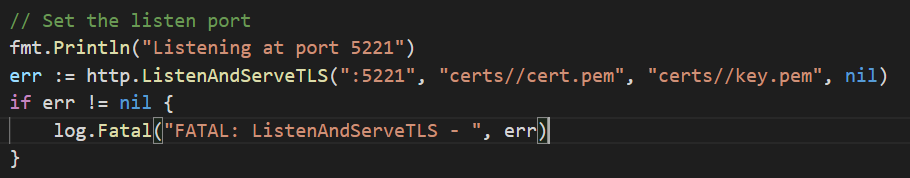


Once install, run the OpenSSL Command Prompt. Type the following to generate the server certificate and private key with OpenSSL:

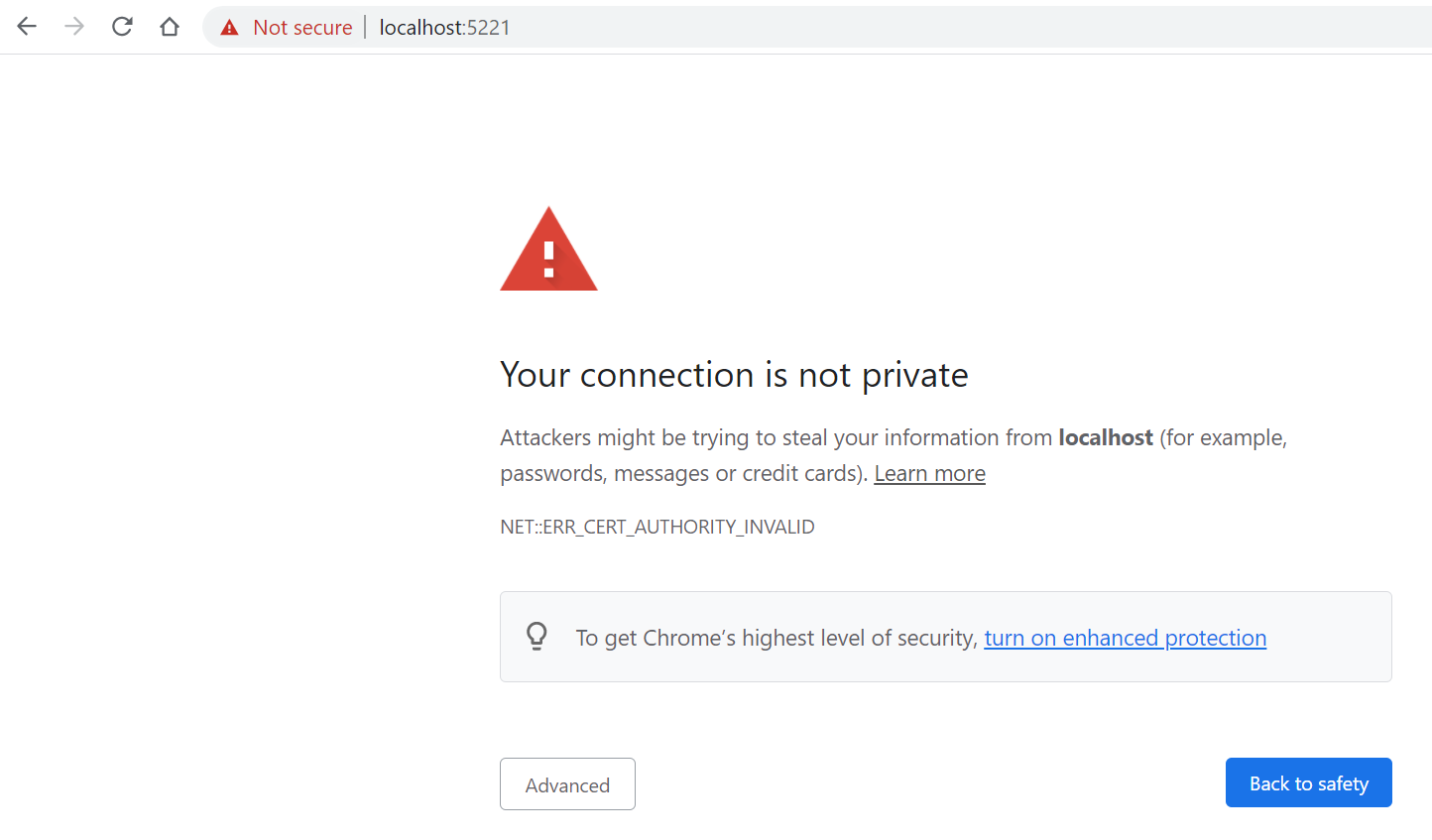
**> openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout key.pem -out cert.pem**



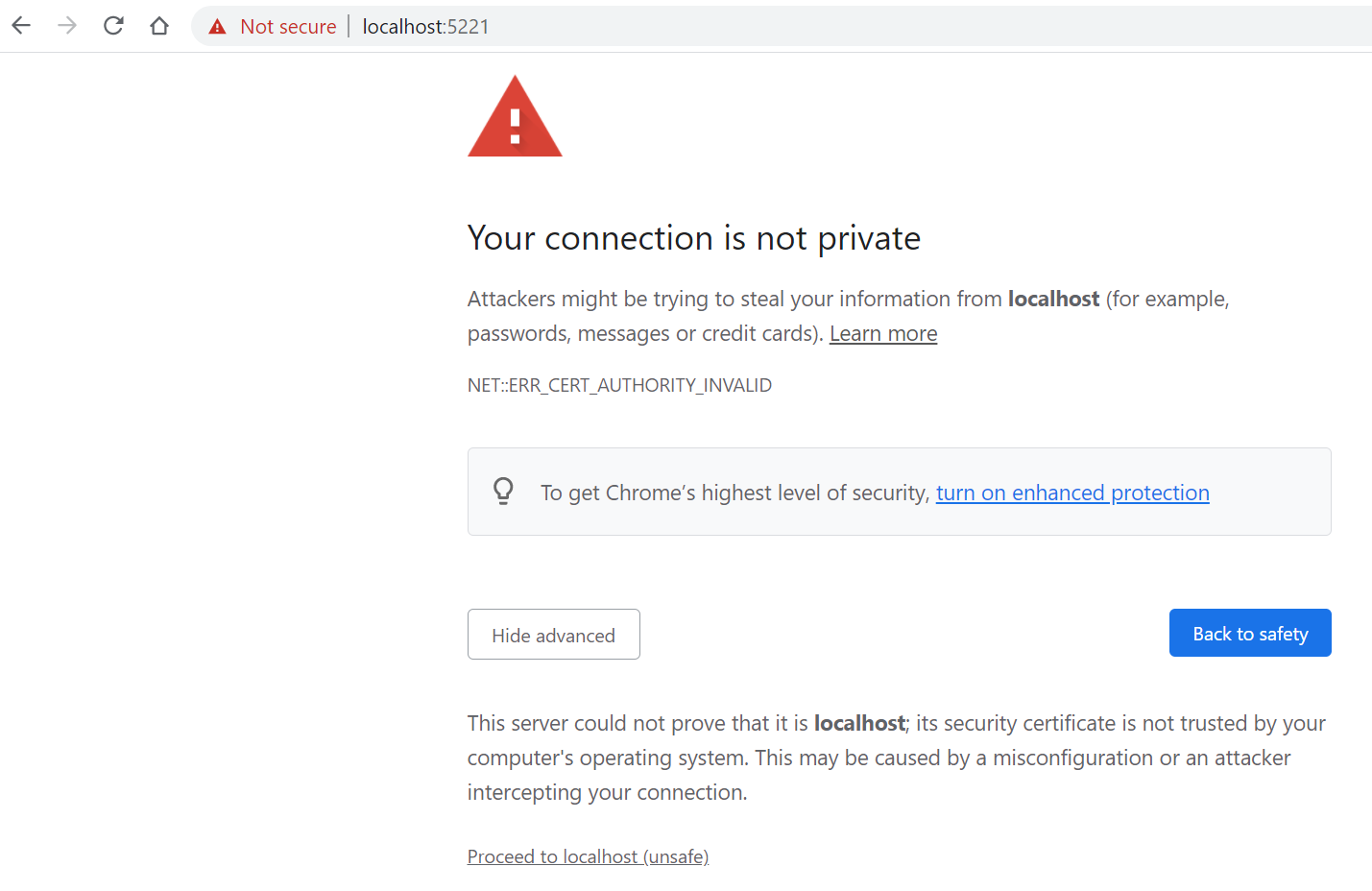
The code to use the certificate/key and start the server is a follows.

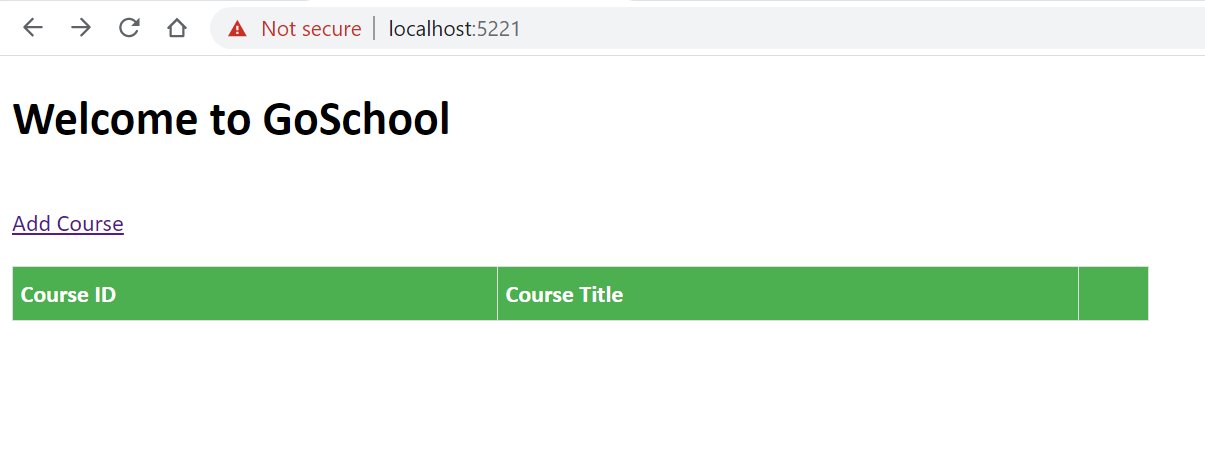


When the server is run, launch the client from the browser at <https://localhost:5221/>. It will show the following warning page. This is because the certificate is not a CA certified SSL certificate. For development and testing, we can ignore the error and click ‘Advanced’.

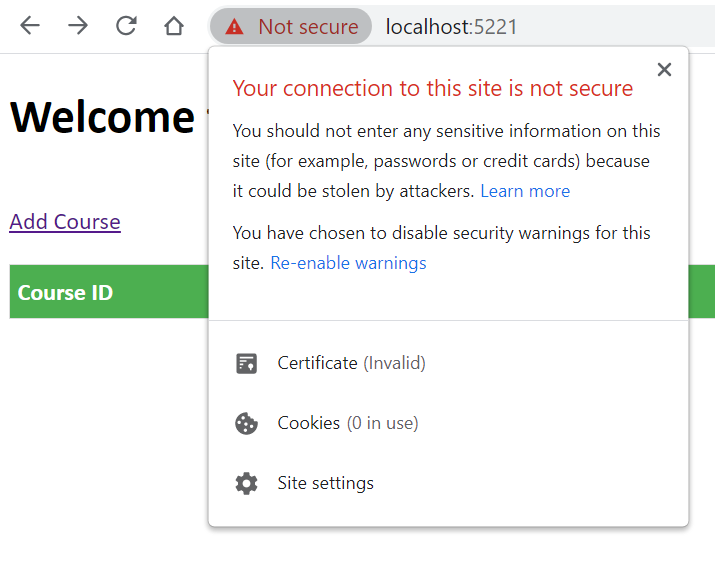


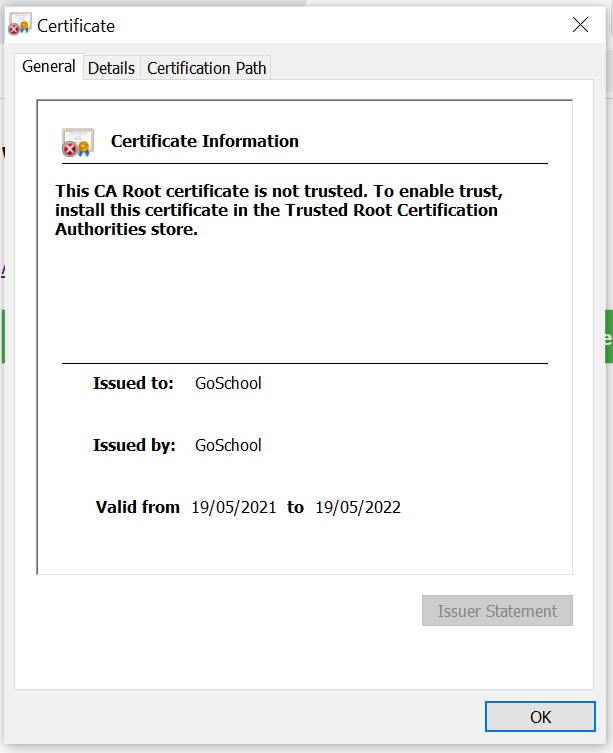
Click ‘Proceed to localhost (unsafe)’.





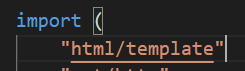
As this is a self-signed certificate, the browser will not trust the connection. But the fact this appears, that means handshaking between server and client already done and thus the SSL/TLS is working.



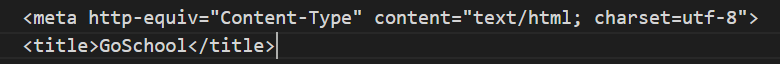


## **CROSS SITE SCRIPTING**

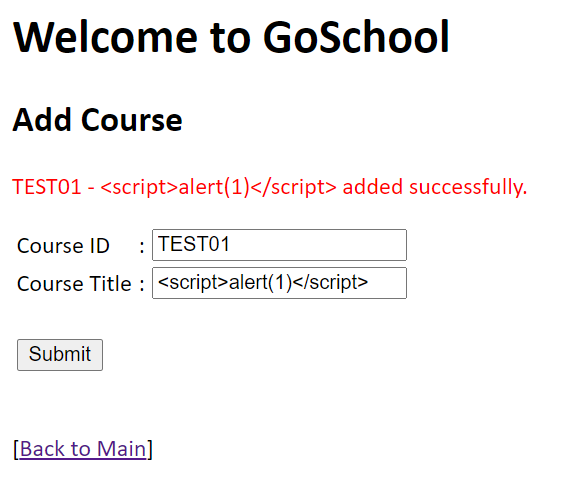
To prevent cross site scripting, it is important NOT to allow content type to be set automatically and to NOT use "text/plain" as implemented in the code. “html/template” is used for proper escaping.



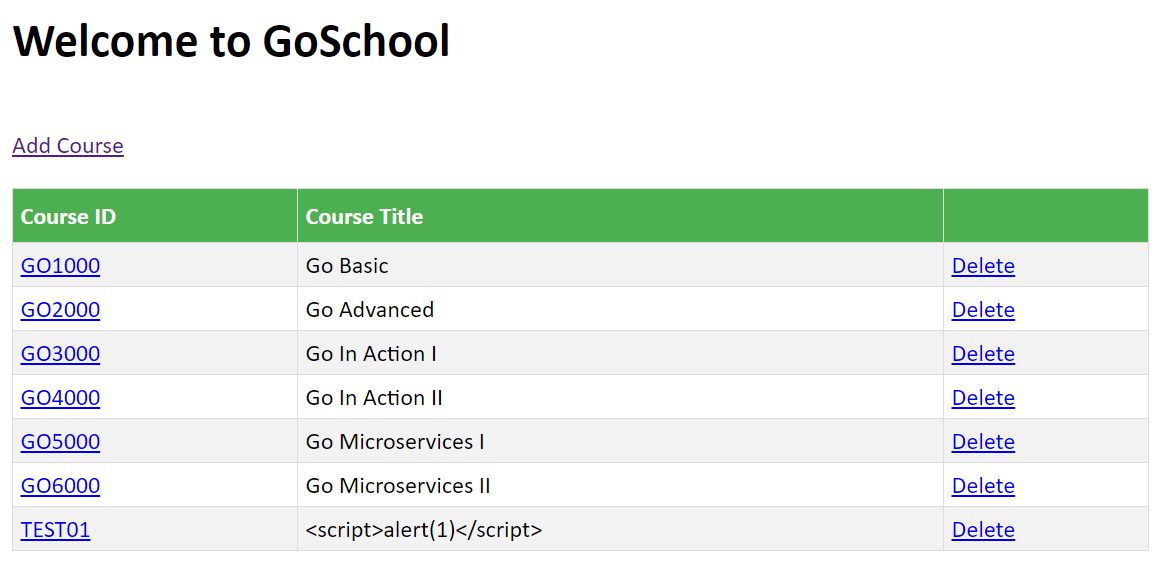
Within the html <head> tag, the <meta> data set for content is set with attribute content=”text/html” as shown.



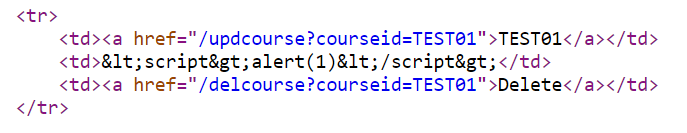
With the above, a test is done to check if cross site scripting will occurs by inputting “<script>alert(1)</script>” into Course Title field in Add Course and clicking submit.



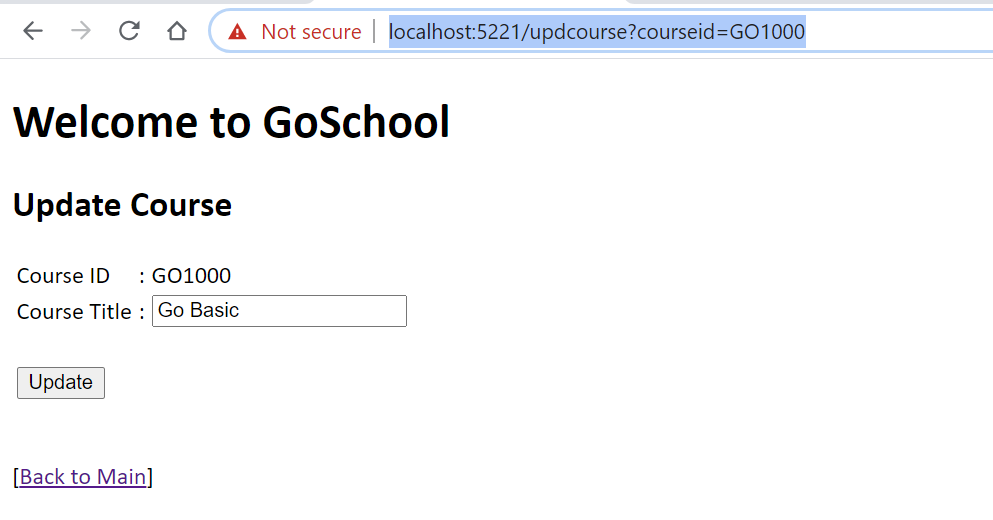
The following displayed shows the script is accepted as a line of text and was not executed with a pop-up alert.



The following is captured in “View page source”.

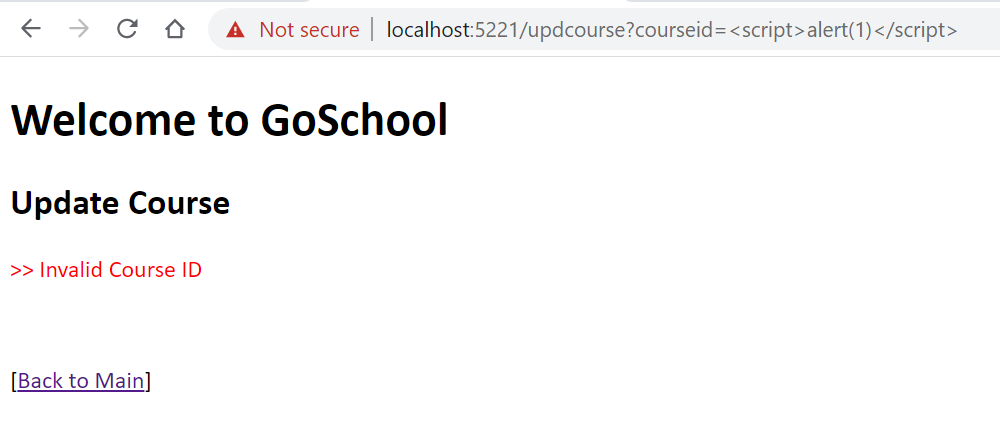


Another test is done on the URL path as follows to ensure that the site is safe.



Update the URL to https://localhost:5221/updcourse?courseid=<script>alert(1)</script> and run it

The script is not run and error message is displayed to the user.

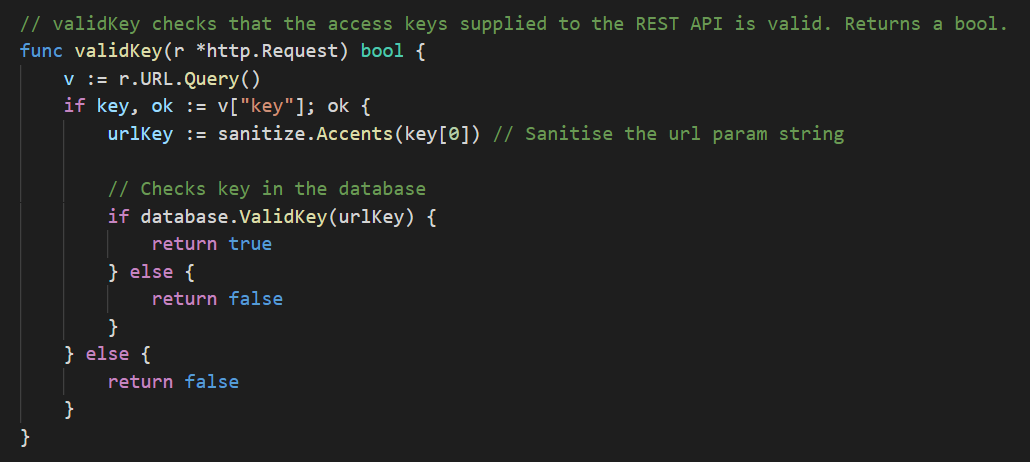


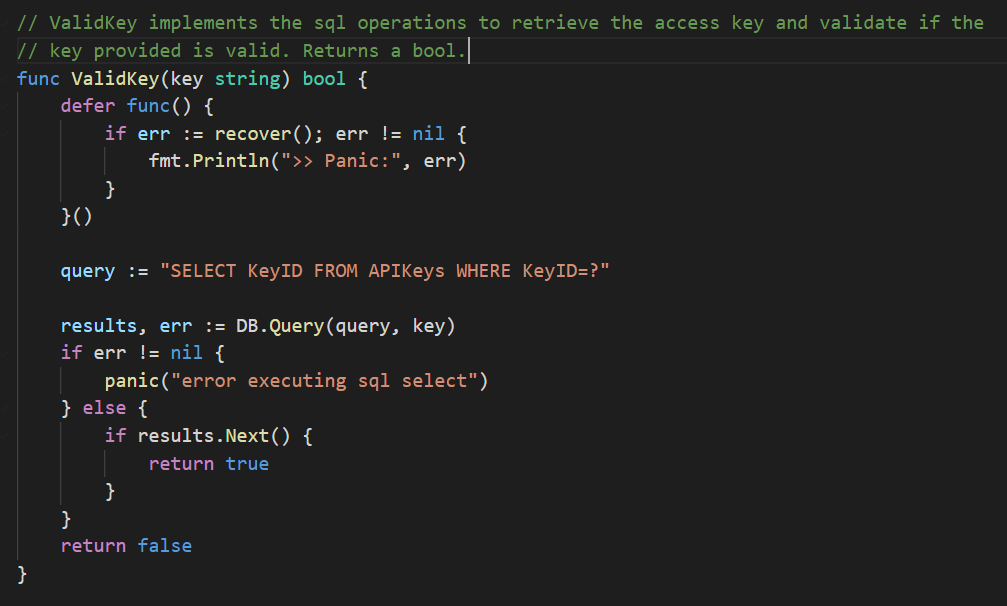
# **REST API**

The REST API allows the client to interface via different method calls. It determines the methods of GET, POST, PUT and DELETE to perform the various CRUD operations requested by the client. The REST API is set to listen at port 5000.

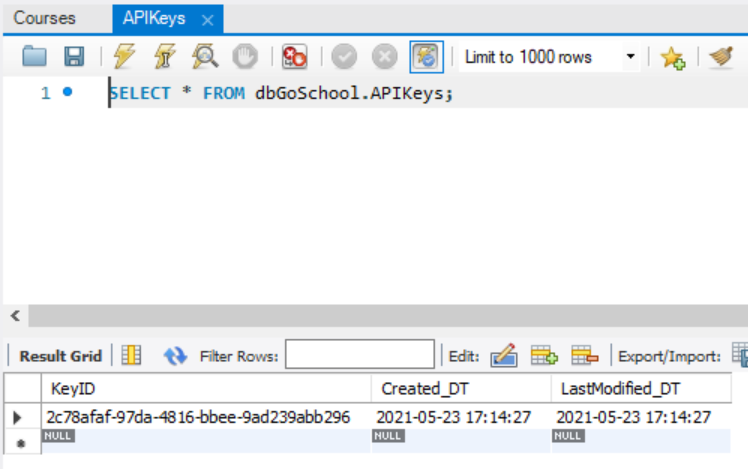
## SECURING THE REST API USING ACCESS KEYS

When the client interfaces with the REST API, it will pass in an access key. The REST API will validate the access key in the database table – “APIKeys” to determine if the client is able to perform the requested CRUD operations.



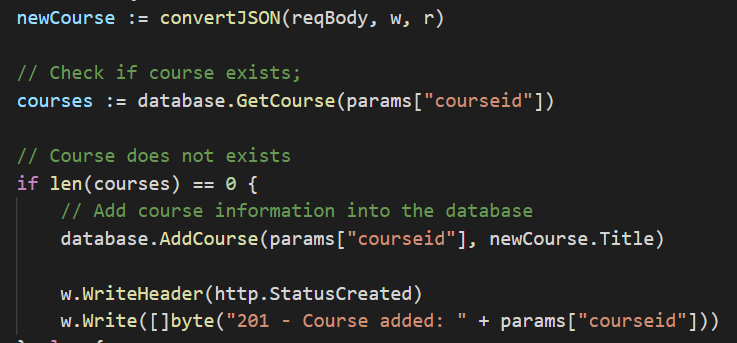


For added security, the access keys in the database could be encrypted. However, due to time constraints, I was not able to implement this at the moment.



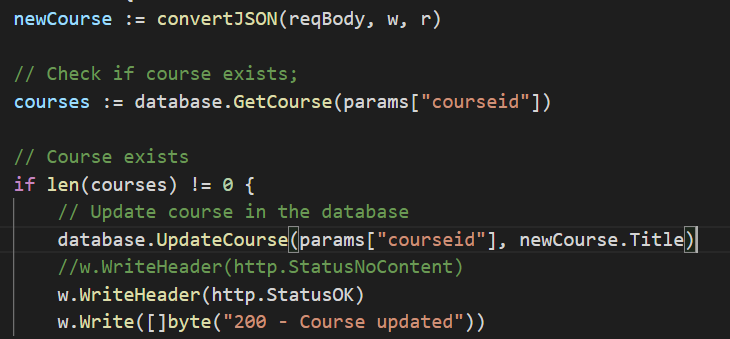
## **ADD COURSE**

When the request method of “POST” is received from the client, the REST API will invoke the AddCourse function in the database package to insert the course details in the database table. It then writes a HTTP response code which the client will receives to determine if the operation is successful.



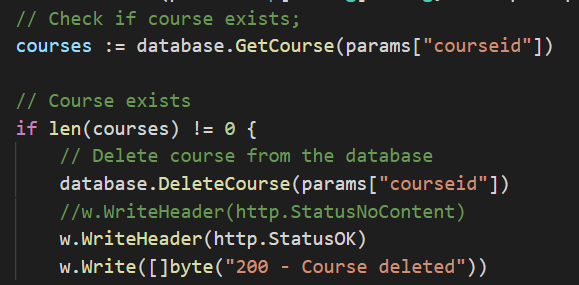
## **UPDATE COURSE**

When the request method of “PUT” is received from the client, the REST API will invoke the UpdateCourse function in the database package to update the course title in the database table. It then writes a HTTP response code which the client will receives to determine if the operation is successful.



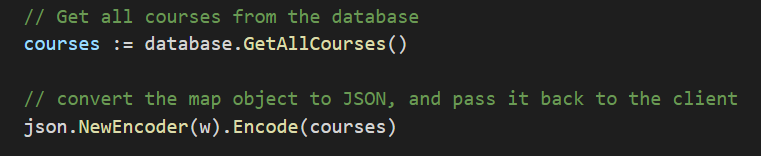
## **DELETE COURSE**

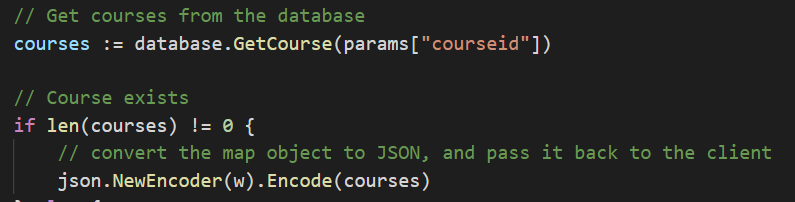
When the request method of “DELETE” is received from the client, the REST API will invoke the DeleteCourse function in the database package to delete the course details in the database table. It then writes a HTTP response code which the client will receives to determine if the operation is successful.



## **GET COURSE/GET ALL COURSES**

When the request method of “GET” is received from the client, the REST API will invoke the GetCourse or GetAllCourses function in the database package to retrieve the course details from the database table. The information retrieved is then stored into a map object which will be then be encoded as a JSON to be passed to the client.

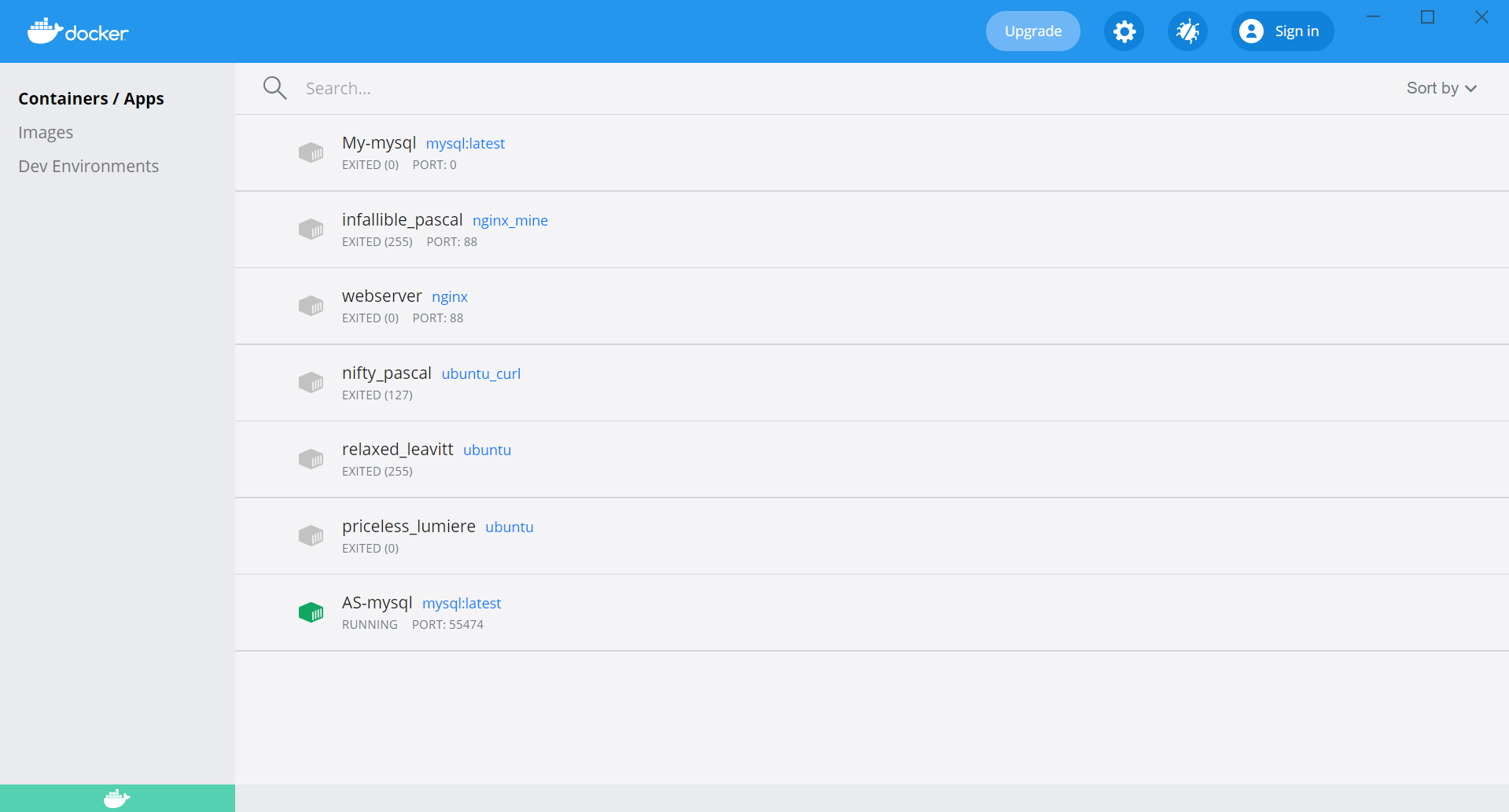




# **DATABASE**

## DEPLOYING MYSQL DATABASE ON A DOCKER CONTAINER

Open Docker Desktop app and ensure that Docker is running.



Docker is running when this is “green”

Start cmd prompt and run the following commands one by one. The port to run the database is defaulted as **55474.**

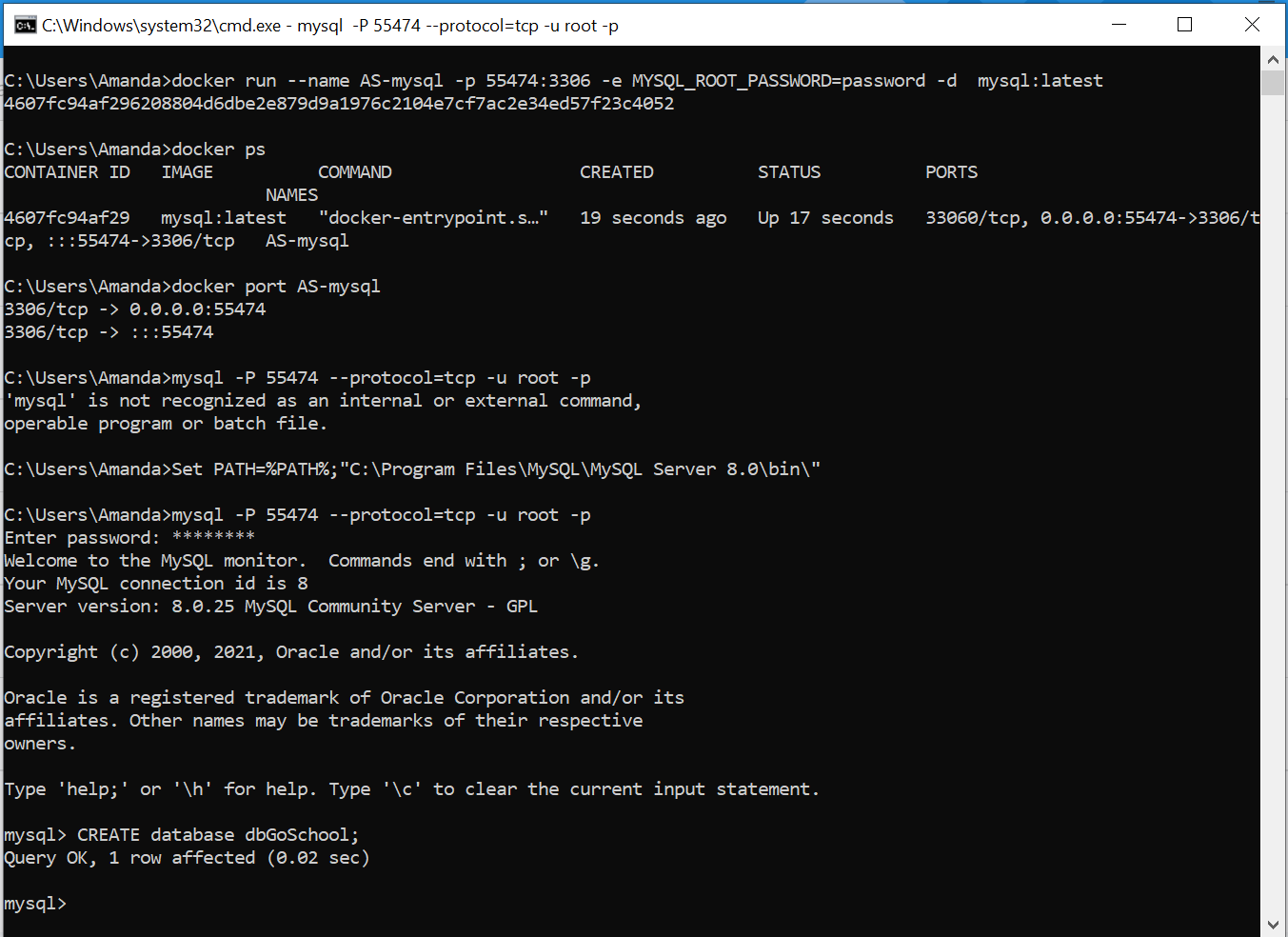
* **docker run --name AS-mysql -p 55474:3306 -e MYSQL\_ROOT\_PASSWORD=password -d mysql:latest**
* **docker ps**
* **docker port AS-mysql**
* **mysql -P 55474 --protocol=tcp -u root –p**
* **Set PATH=%PATH%;"C:\Program Files\MySQL\MySQL Server 8.0\bin\"** *(run this if the previous statement has 'mysql' is not recognized … issue)*
* **mysql -P 55474 --protocol=tcp -u root –p** *(re-run this statement)*
* Enter password as “*password*” when prompted
* At the SQL prompt, perform the following SQL statements to:
  + Create Database

CREATE DATABASE dbGoSchool;

* Use Database

USE dbGoSchool;

* The following screenshots show the results of the above.



* Continue at the SQL prompt, perform the following SQL statements to:
  + *CREATE 2 TABLES*

“Courses” table manages the CRUD operations for courses.

CREATE TABLE `dbGoSchool`.`Courses` (

`CourseID` VARCHAR(6) NOT NULL,

`CourseTitle` VARCHAR(45) NOT NULL,

`Created\_DT` DATETIME NULL,

`LastModified\_DT` DATETIME NULL,

PRIMARY KEY (`CourseID`));

“APIKeys” table keeps track of the access keys for validations of the keys sent by the client

CREATE TABLE `dbGoSchool`.`APIKeys` (

`KeyID` VARCHAR(45) NOT NULL,

`Created\_DT` DATETIME NULL,

`LastModified\_DT` DATETIME NULL,

PRIMARY KEY (`KeyID`));

Insert the key value in the “APIKeys” table to validate the key sent by the client

* + *INSERT INTO TABLES*

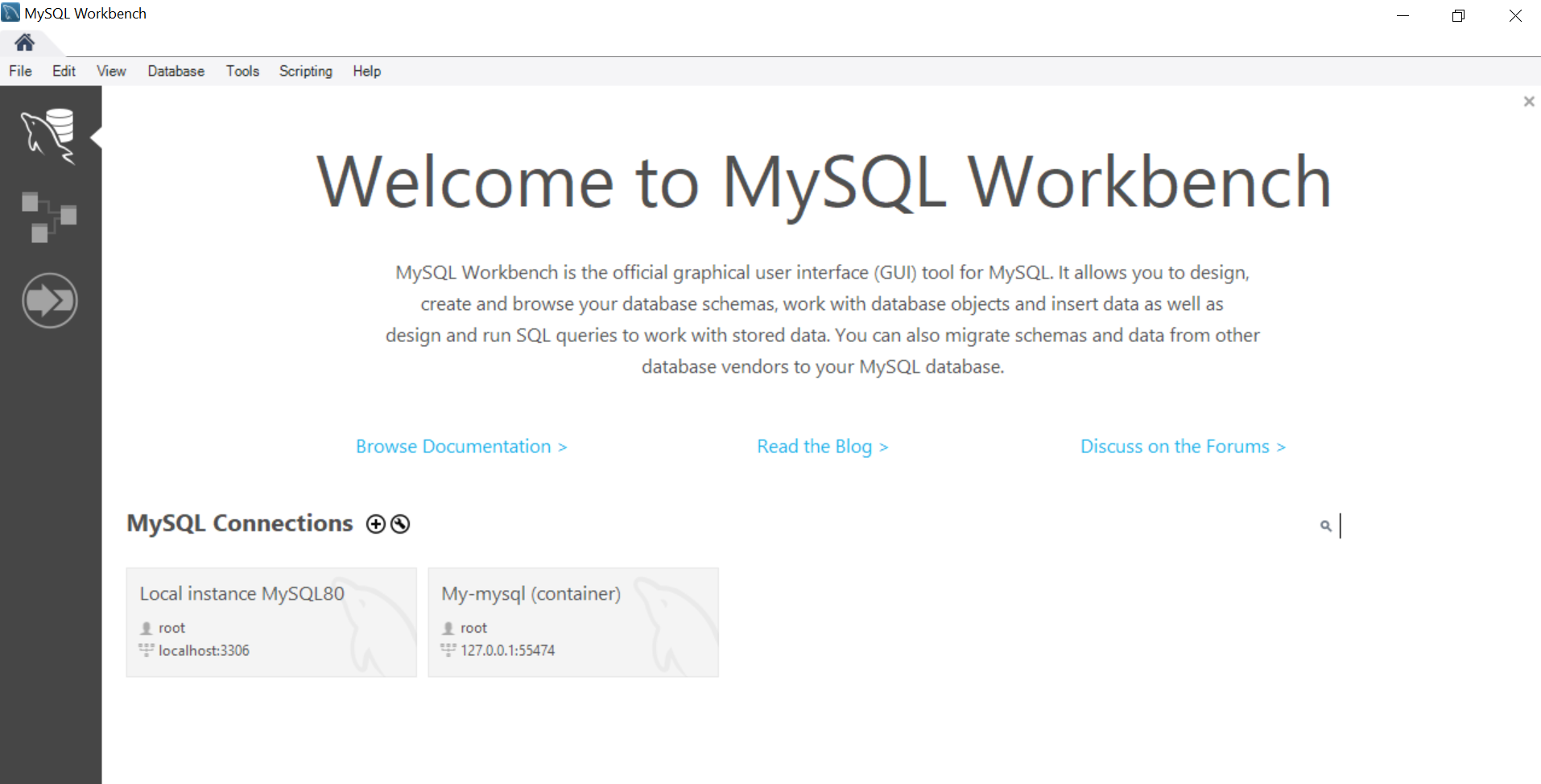
INSERT INTO `dbGoSchool`.`APIKeys` (

`KeyID`, `Created\_DT`,`LastModified\_DT`)

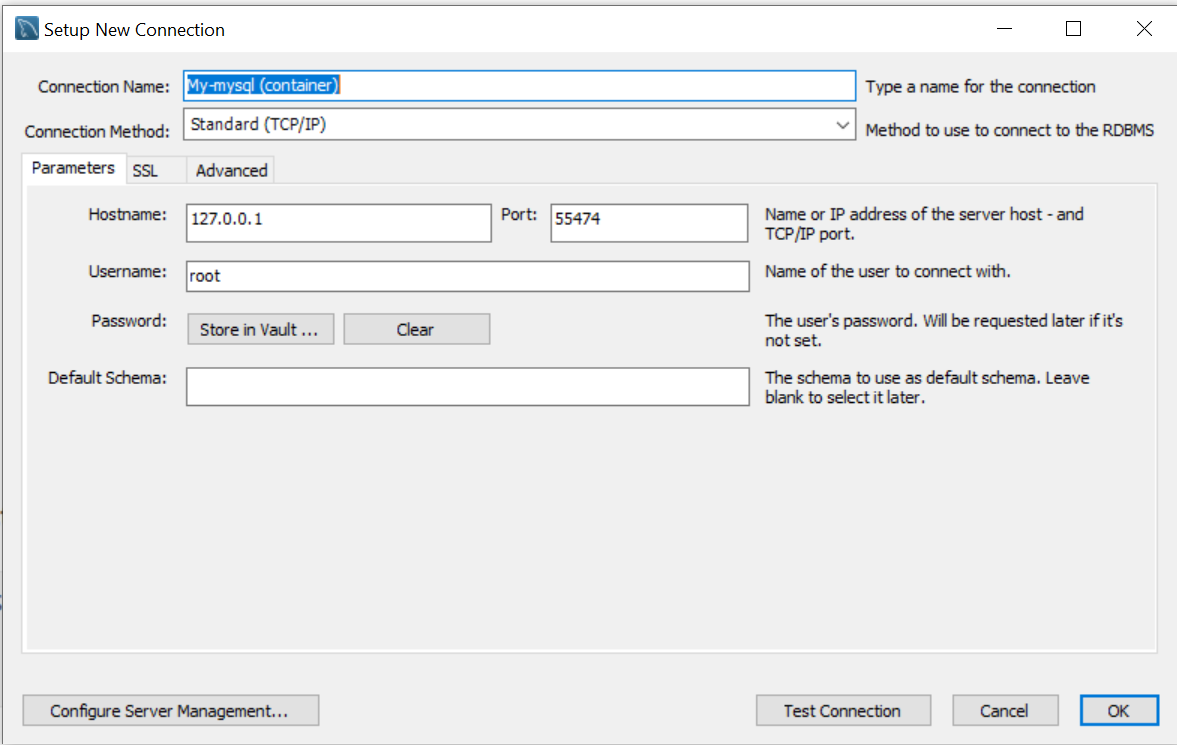
VALUES ("2c78afaf-97da-4816-bbee-9ad239abb296", now(), now());

## SETUP MYSQL WORKBENCH

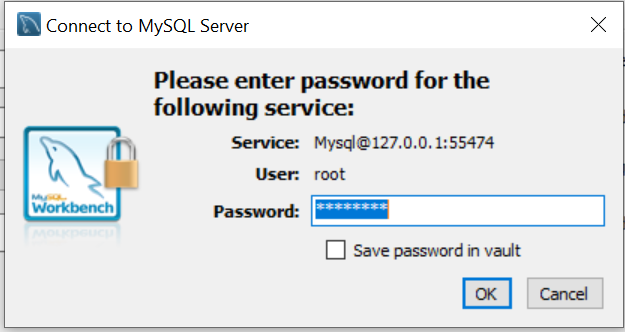
Start MySQL Workbench. Click ‘+’ to add new connection.



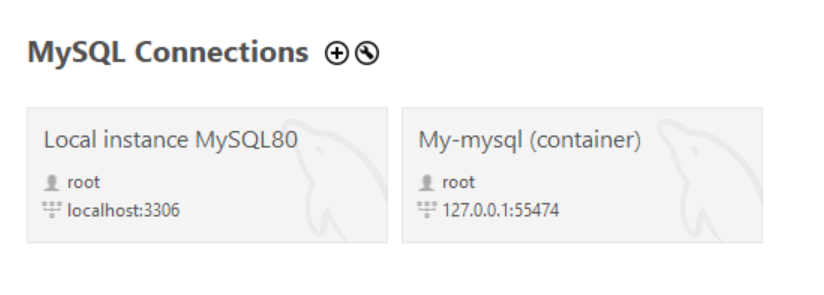
Provide a Connection Name and set the port as setup from docker previously. Click ‘Test Connection’ button.



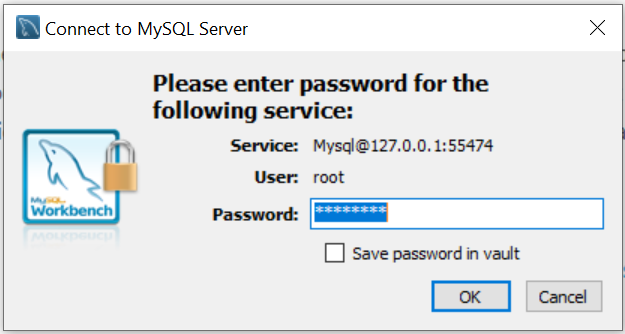
Enter ‘password’ and click ‘OK’



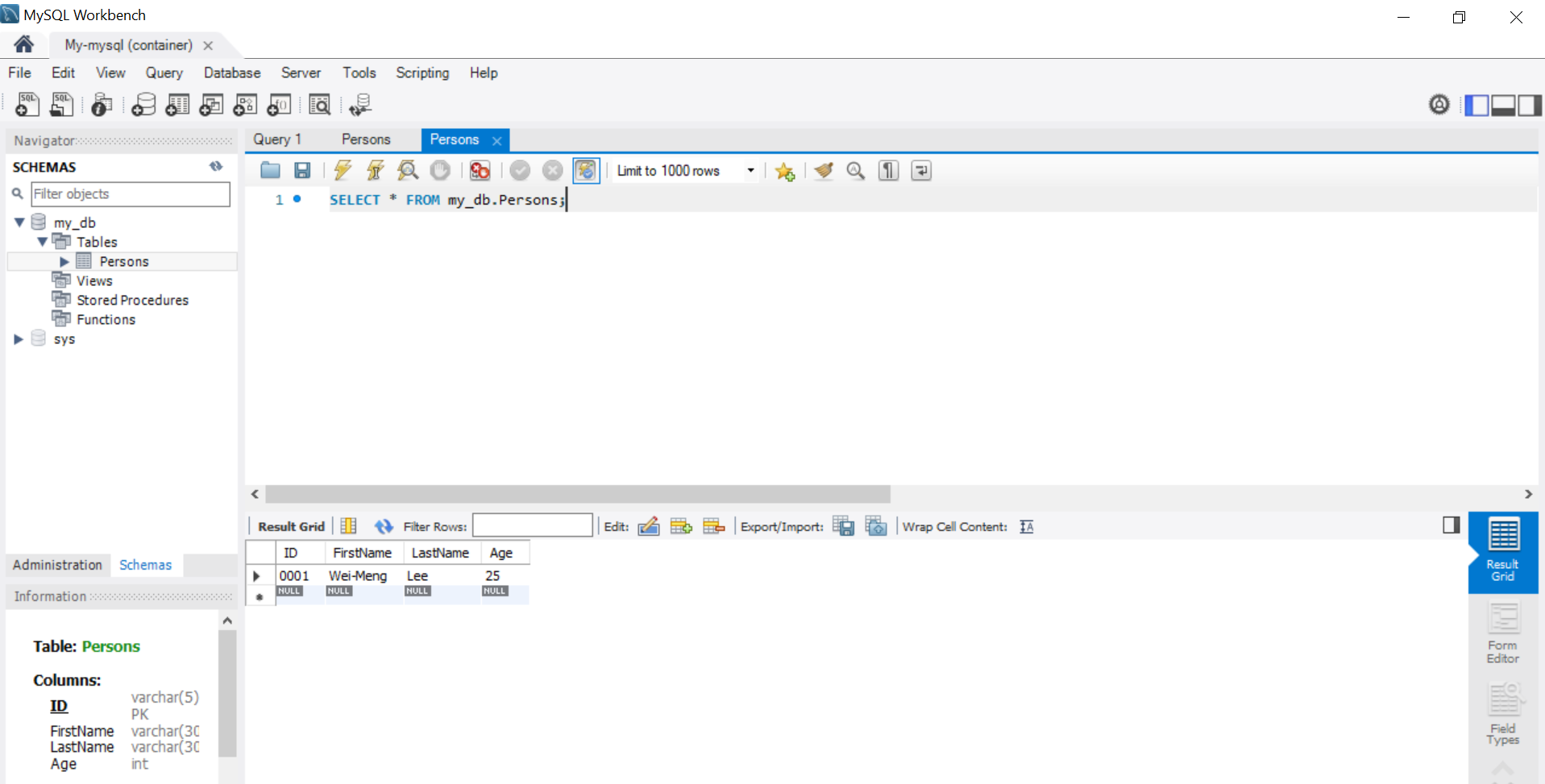
Select the new connection.



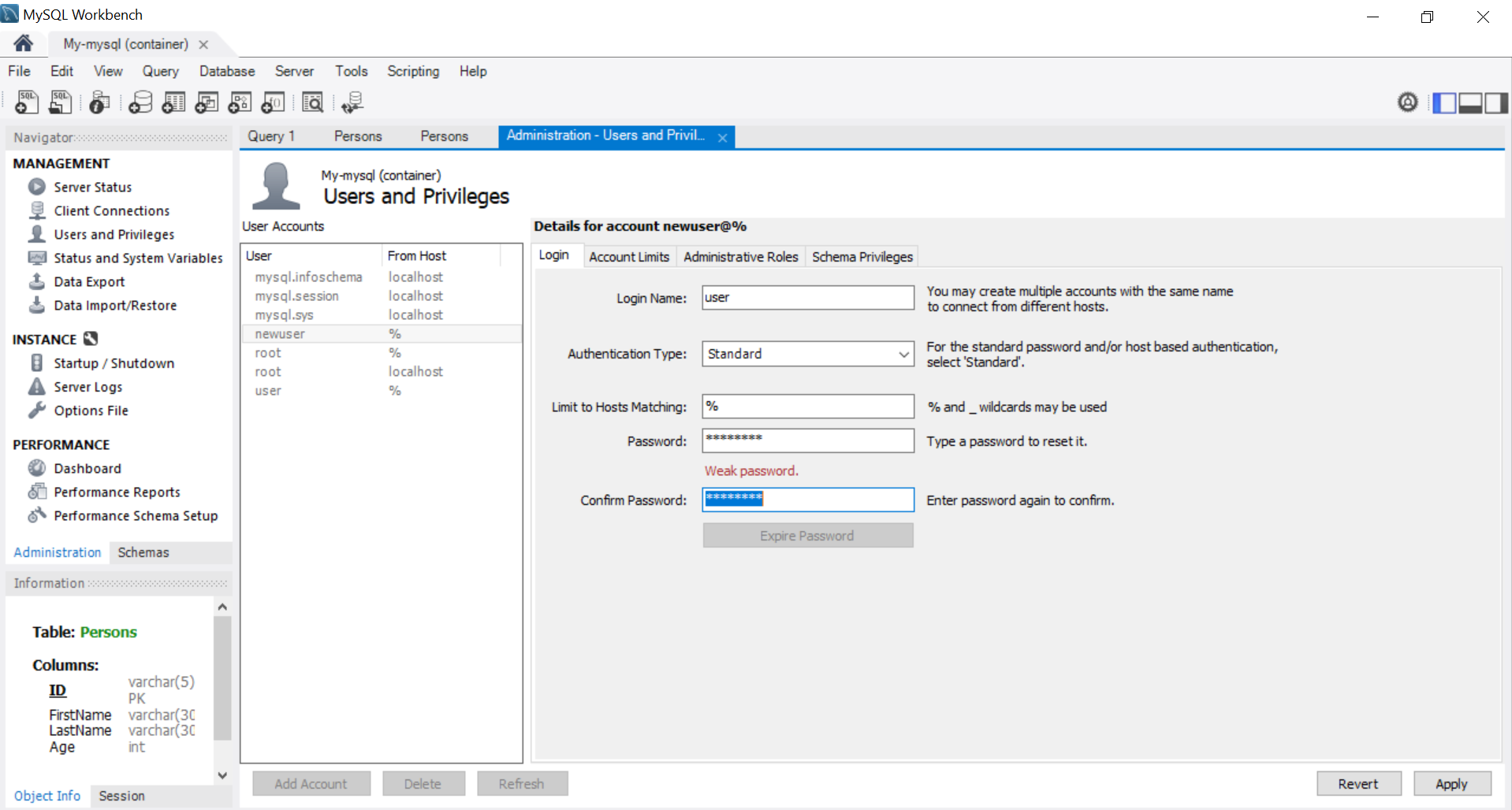
Validate the password and click ‘OK’.



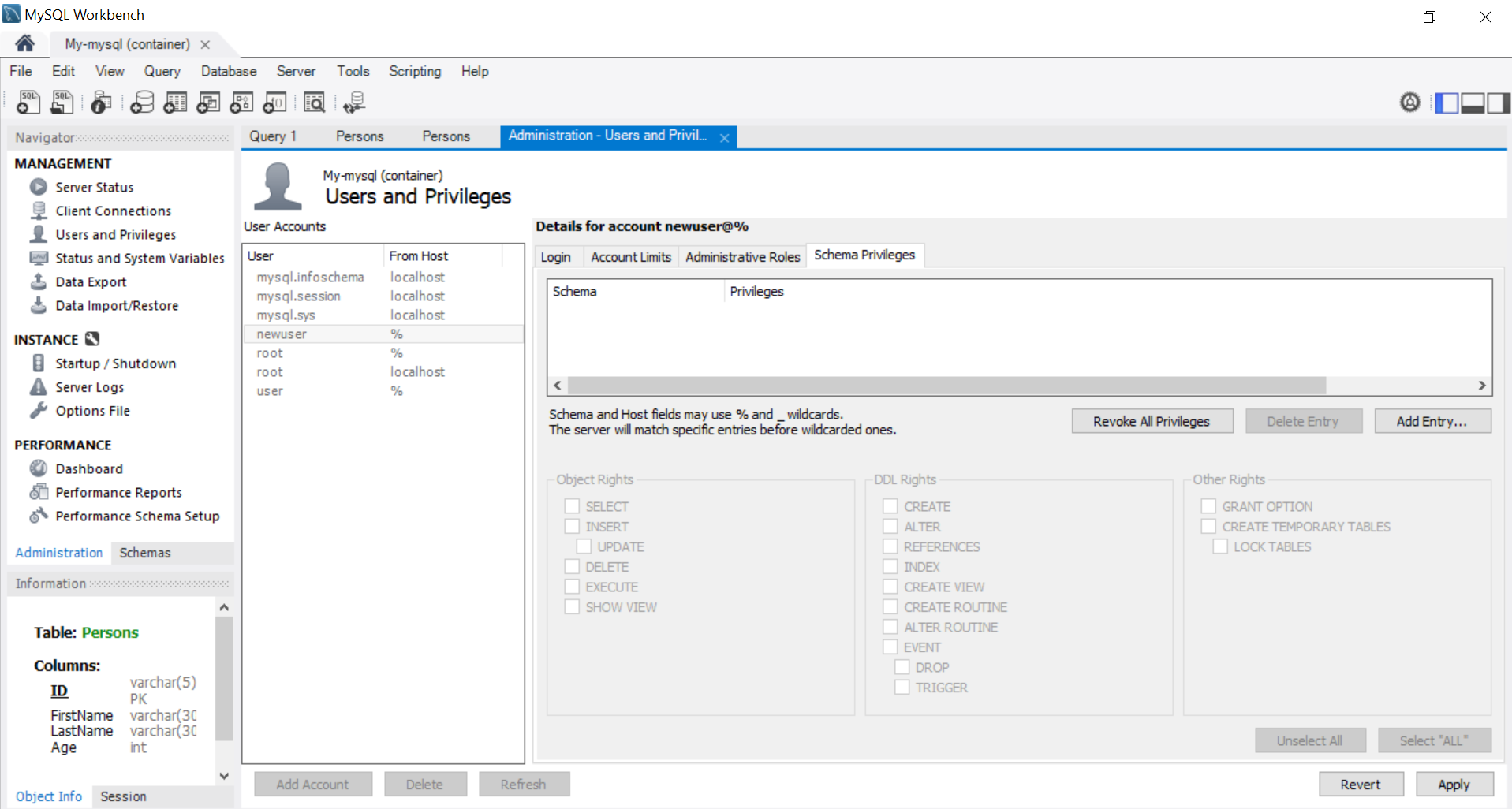
Under ‘Schemas’ tab, find my\_db -> Tables -> right-click ‘Persons’ table and select rows. You should see the following.



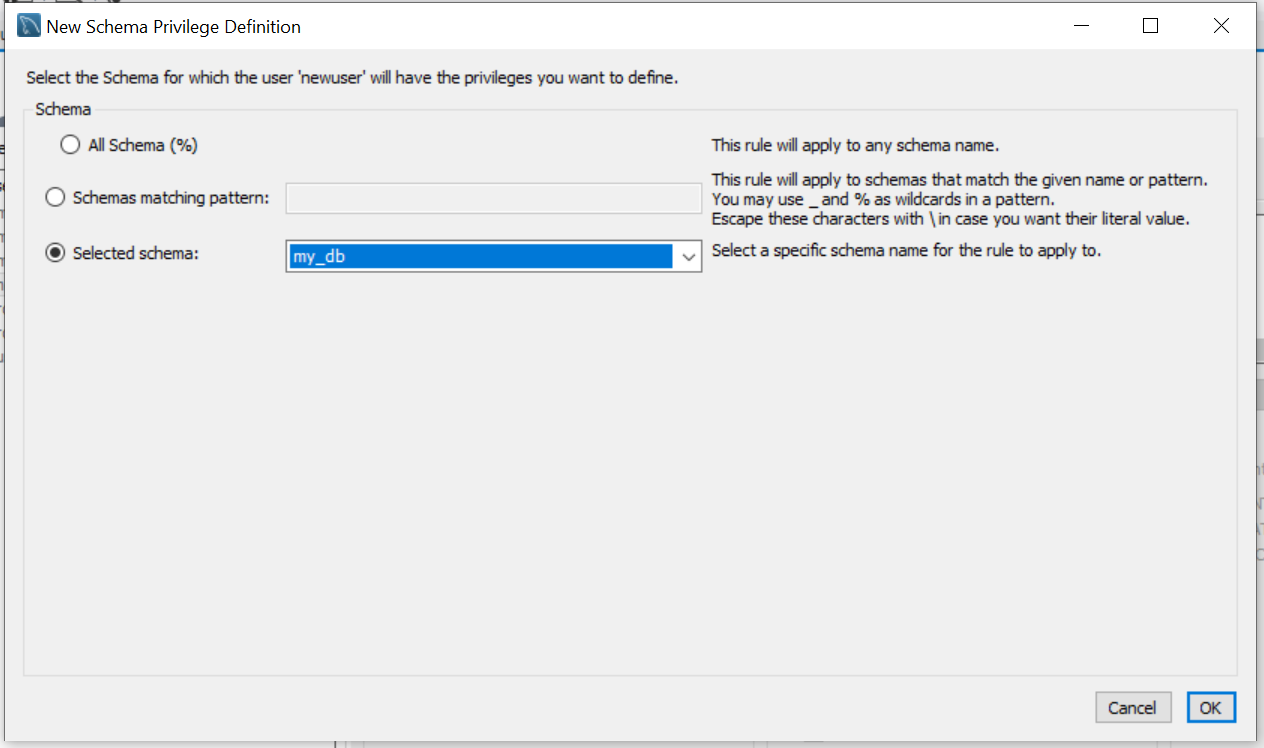
Under ‘Administration’ tab, click ‘Users and Privileges’ under MANAGEMENT. Click ‘Add Account’. Enter new Login Name -> ‘user’ and Password is ‘password’. Do not click ‘Apply’ yet. Select ‘Schema Privileges’ tab.



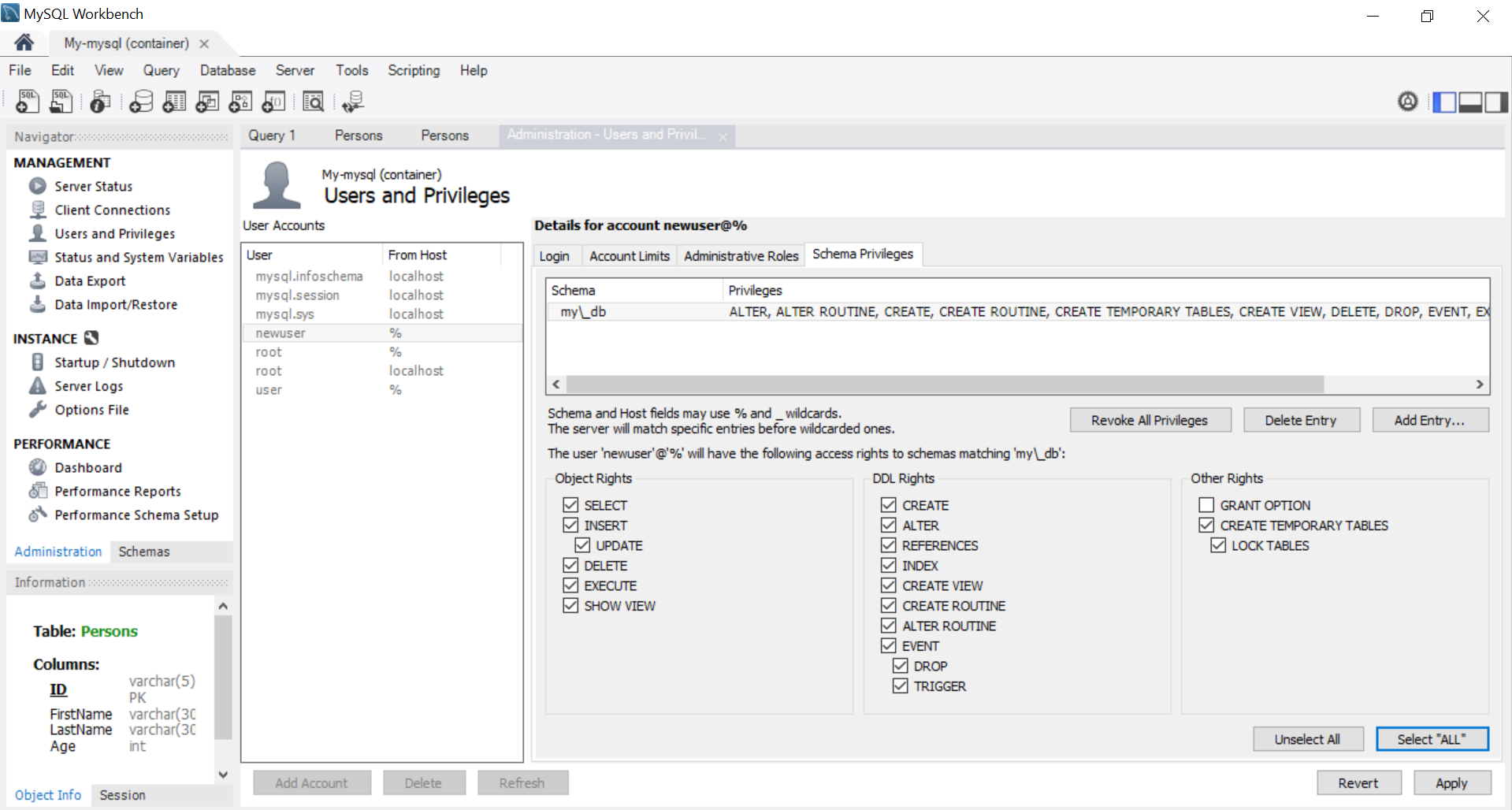
Under ‘Schema Privileges’ tab, select ‘Add Entry…’



Select ‘Selected schema’, choose ‘my\_db’. Click ‘OK’

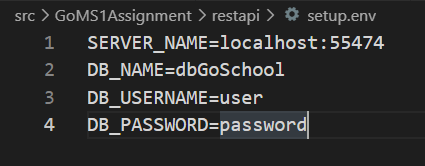


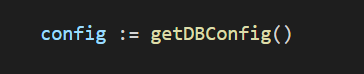
Click ‘Select “ALL”. Click ‘Apply’.

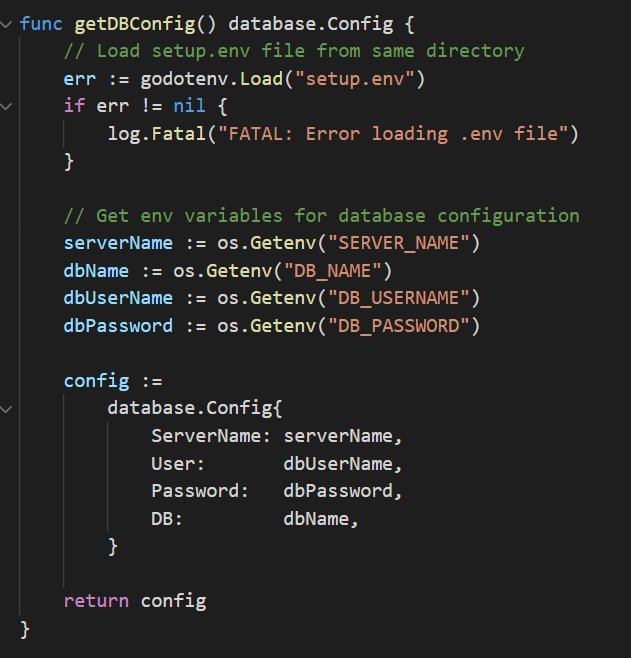


## USING MYSQL DATABASE

The configuration to connect to the database is stored in a .env file so that the data will not be stored in the codes. The database activities are consolidated into a **database** package.

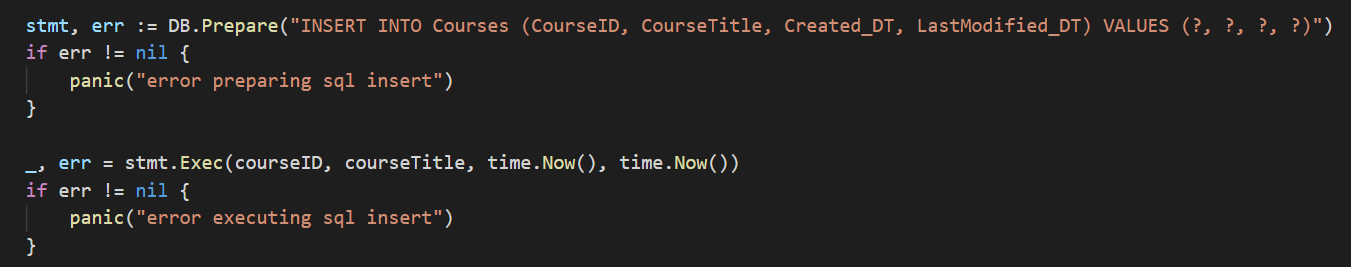






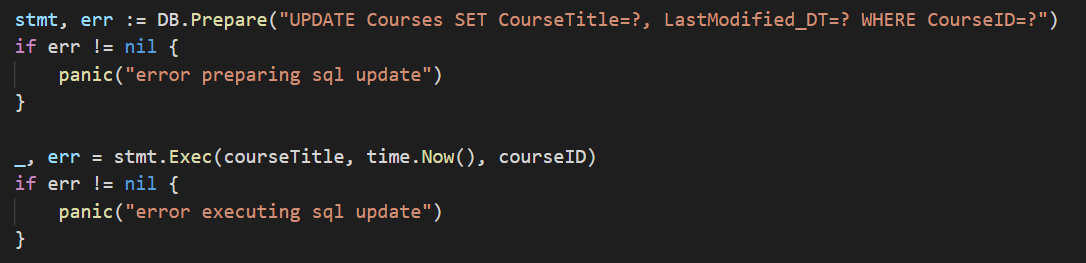
## **ADD COURSE**

The AddCourse function in the database package is invoked by the REST API when it receives a “POST” method from the client. The function will insert the course details into the database table using the “INSERT” SQL statement.



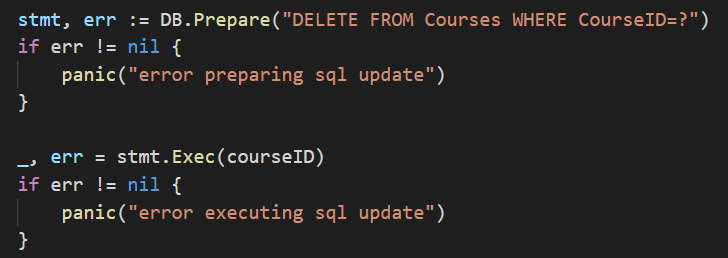
## **UPDATE COURSE**

The UpdateCourse function in the database package is invoked by the REST API when it receives a “PUT” method from the client. The function will update the course title in the database table using the “UPDATE” SQL statement.



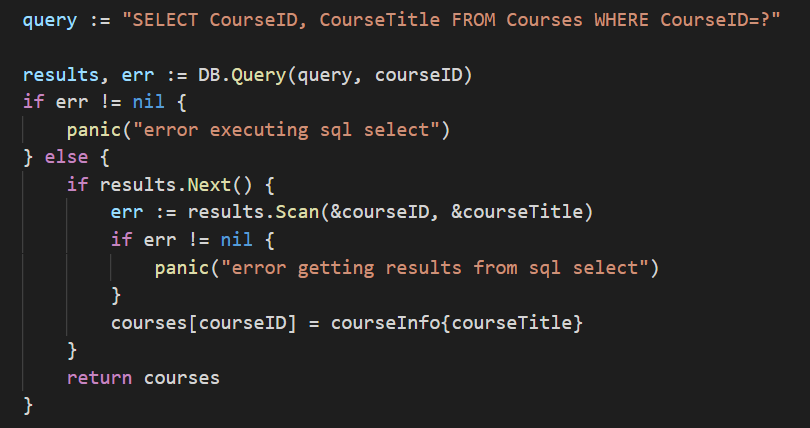
## **DELETE COURSE**

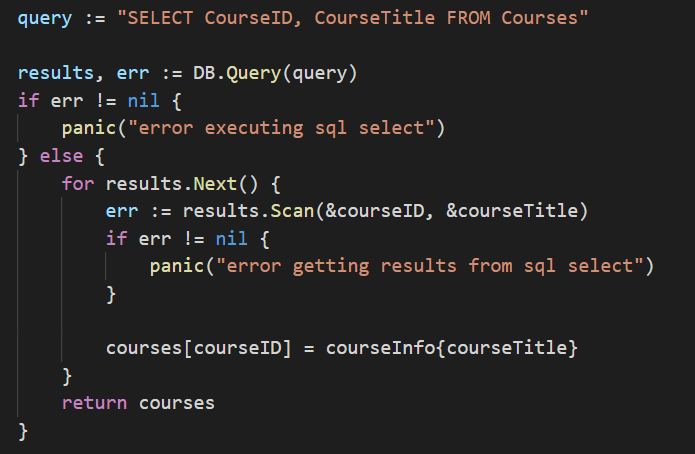
The DeleteCourse function in the database package is invoked by the REST API when it receives a “DELETE” method from the client. The function will delete the course details in the database table using the “DELETE” SQL statement.



## **GET COURSE/GET ALL COURSES**

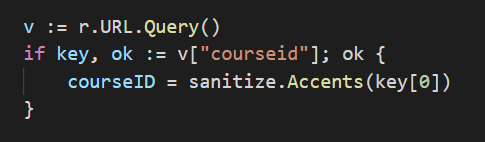
The GetCourse and GetAllCourses function in the database package is invoked by the REST API when it receives a “GET” method from the client. The function will retrieve the course details from the database table using the “SELECT” SQL statement and stores the information into a map object.





# **SANITIZATION**

To perform data sanitisation in the program, I am using third party package from “github.com/kennygrant/sanitize”.

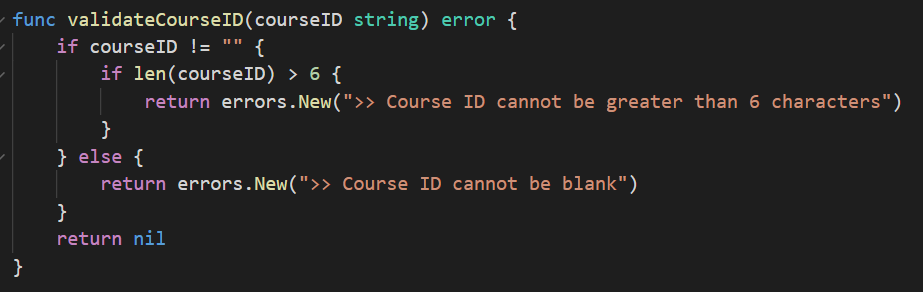


# **ERROR HANDLING**

The following error handling methods are implemented in the application.

## **USING ERRORS.NEW**

errors.New creates a new error with its dedicated message.



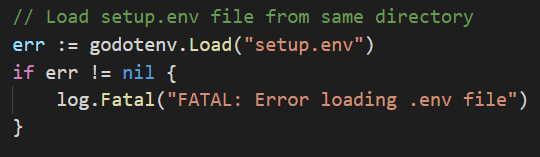
## **USING PANIC**

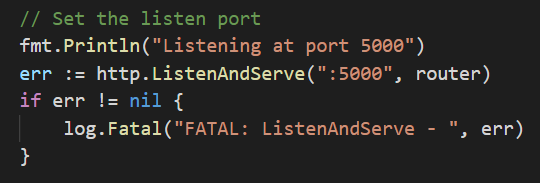
When panic occurs due to database errors, the *defer func for recover()* ensures that the application can continue to run.

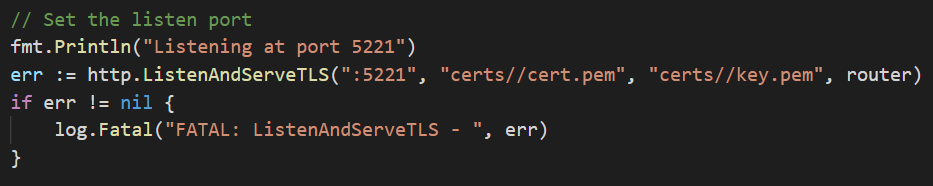


## **LOG.FATAL**

log.Fatal is used in the application with the understanding that Fatal effectively logged a message, then call os.Exit (1). It is used knowing that the error is essentially unrecoverable and the application will not be able to run properly.

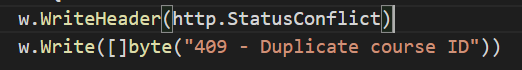






## **HTTP RESPONSE CODE**

The REST API writes the http response code for any errors.



# **GO DOCUMENTATION**

* Click on the following link to view the full documentation in pdf format or open it from the same project folder.

[***GoMS1Assignment - Go Documentation.pdf***](GoMS1Assignment%20-%20Go%20Documentation.pdf)

* Alternatively, you can install the godoc library.

**go get -u golang.org/x/tools/godoc**

* Run the following command in the VSC terminal or cmd prompt. Ensure that you have godoc installed.



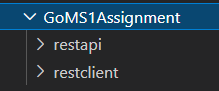
* Open up the browser, enter the URL:

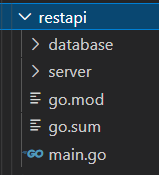
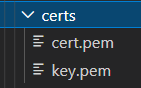
[***http://localhost:7001/pkg/GoMS1Assignment/?m=all***](http://localhost:7001/pkg/GoMS1Assignment/?m=all)

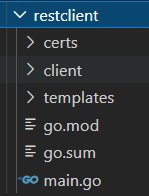
* Note: For illustration purposes to display the comments of unexported functions in godoc, I am appending ***“/?m=all***” in the URL link. Otherwise, the documentation only shows the comments of exported functions (functions which are capitalized).

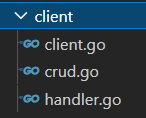
# INSTRUCTIONS ON HOW TO DEPLOY THE APPLICATION

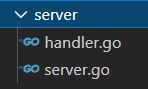
For information, my program is currently running Go version - Go1.16.3. My current project files resides in my local ***C:\Users\Amanda\go\src\GoMS1Assignment\*** with the following directories/files structures.

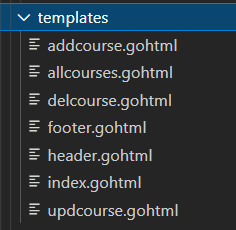












## INSTALL CREATED PACKAGES FOR *CLIENT* AND *SERVER*

* Run in cmd prompt >***go install*** in the respective folders of the 2 packages that I have created.
  + Install ***server*** package in ***GoMS1Assignment/restapi/server/***



* + Install ***client*** package in ***GoMS1Assignment/restclient/client/***



* It is important to use the same folder name structure because the import of the self-created packages in the codes takes the following format.





## IMPORT 3RD PARTY PACKAGES

To get the 3rd party packages used in the application. Run in cmd prompt >***go mod init*** then ***>go mod tidy*** in ***restapi*** and ***restclient*** folders respectively



“restapi” folder



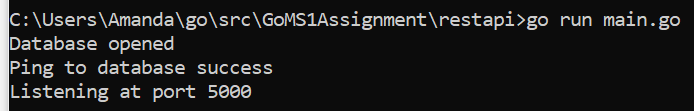
“restclient” folder

## RUNNING THE APPLICATION

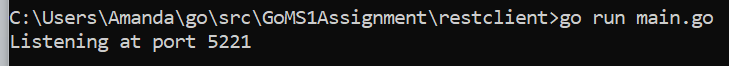
* Make sure the steps under the section in Database have been performed to deploy the database. Run the database by opening a cmd prompt and typing >***docker ps*** to ensure that the database is running in Docker. Otherwise type >***docker start <container\_id>*** to start it.



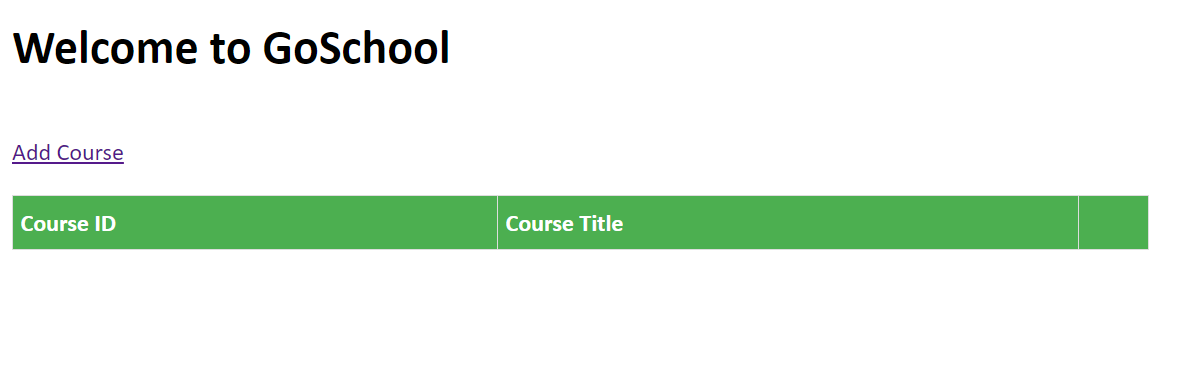
* Start the server: Open a cmd prompt window, type >***go run main.go*** in ***GoMS1Assignment/restapi/*** folder.



* Start the client: Open another cmd prompt window, type >***go run main.go*** in ***GoMS1Assignment/restclient/*** folder.



* Open the browser in Chrome and type in [***https://localhost:5221/***](https://localhost:5221/) to start the client web application. The following page will be displayed.

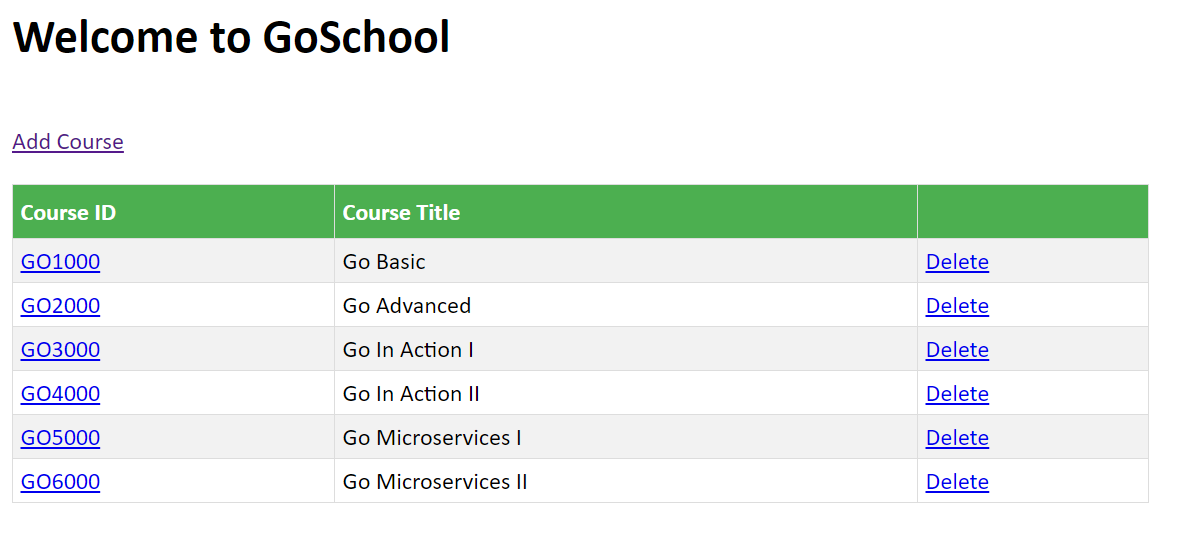


# GUIDE TO TEST THE APPLICATION

Start the client by keying in the URL: [***https://localhost:5221/***](https://localhost:5221/) in the browser***.***

## RETRIEVE ALL COURSES

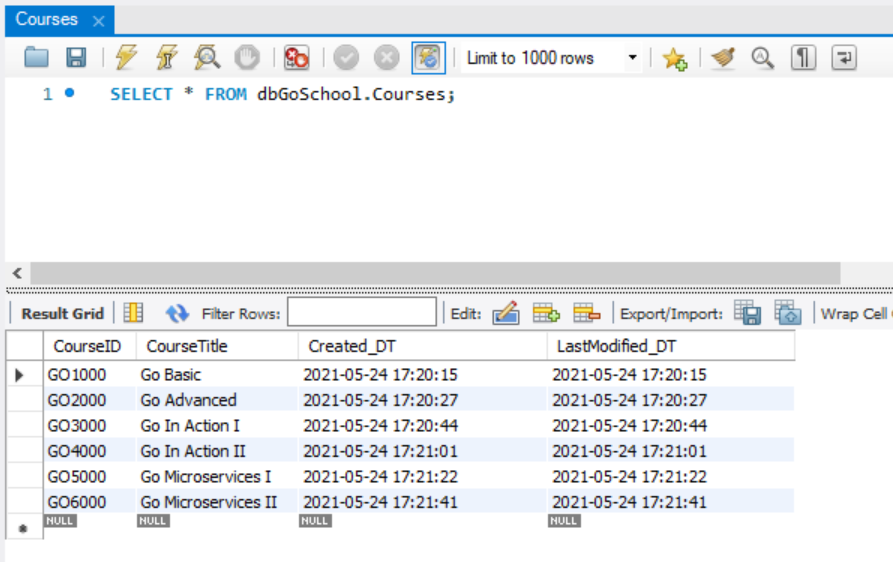
By default, the index page will list all the courses that are retrieved from the database. Checks that information in the database in MySQL Workbench tallies with what is displayed as follows.



Delete course

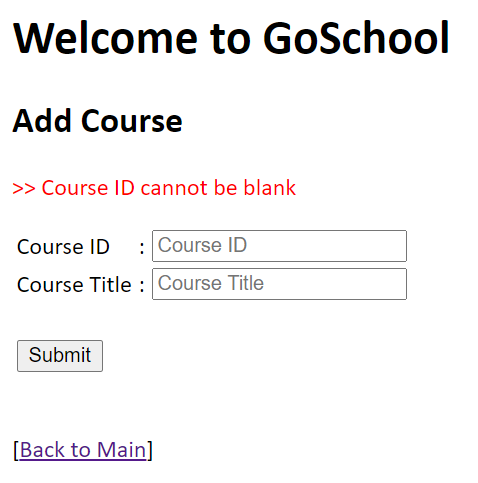
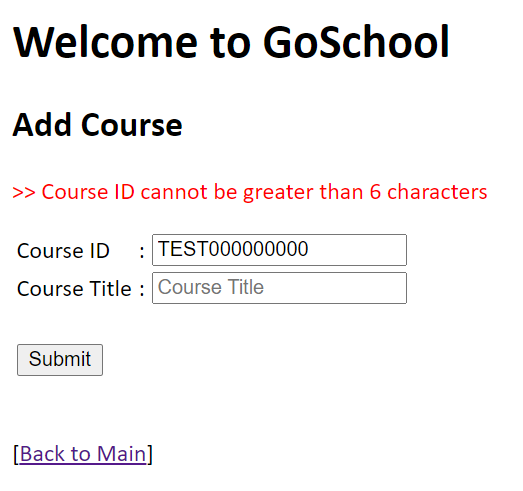
Update course

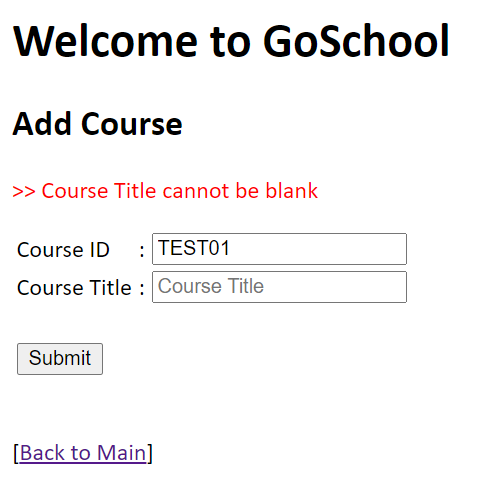
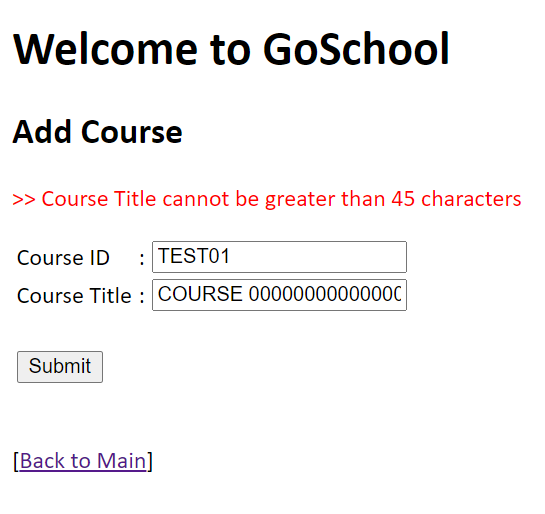
Add a new course

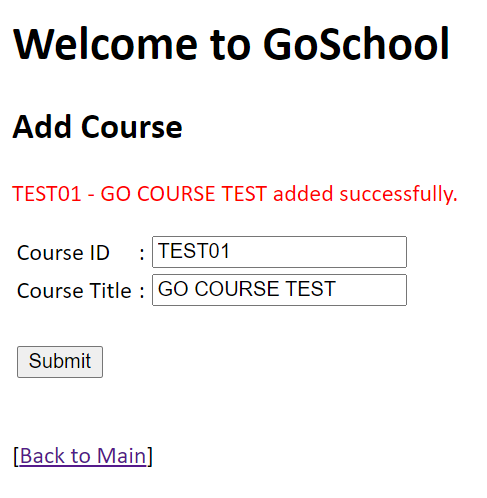


## ADD COURSE

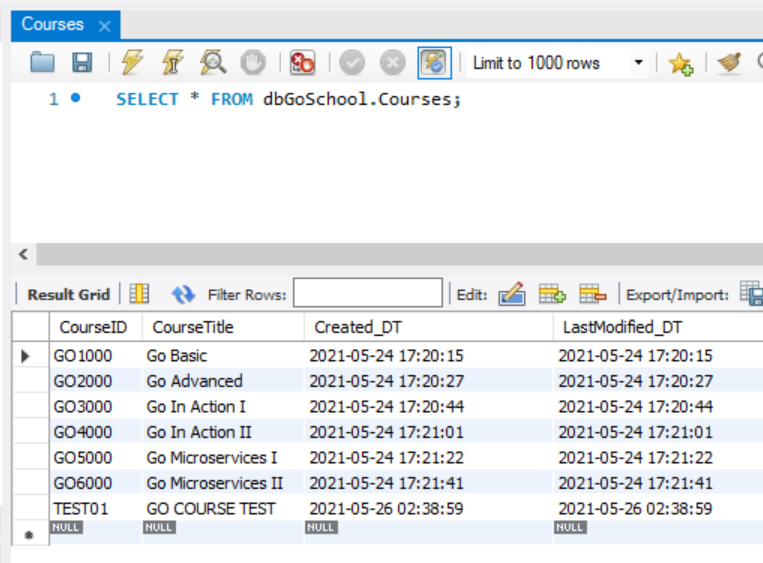
Click on ‘Add Course’ hyperlink in the index page. Test the text fields to ensure data input is valid.

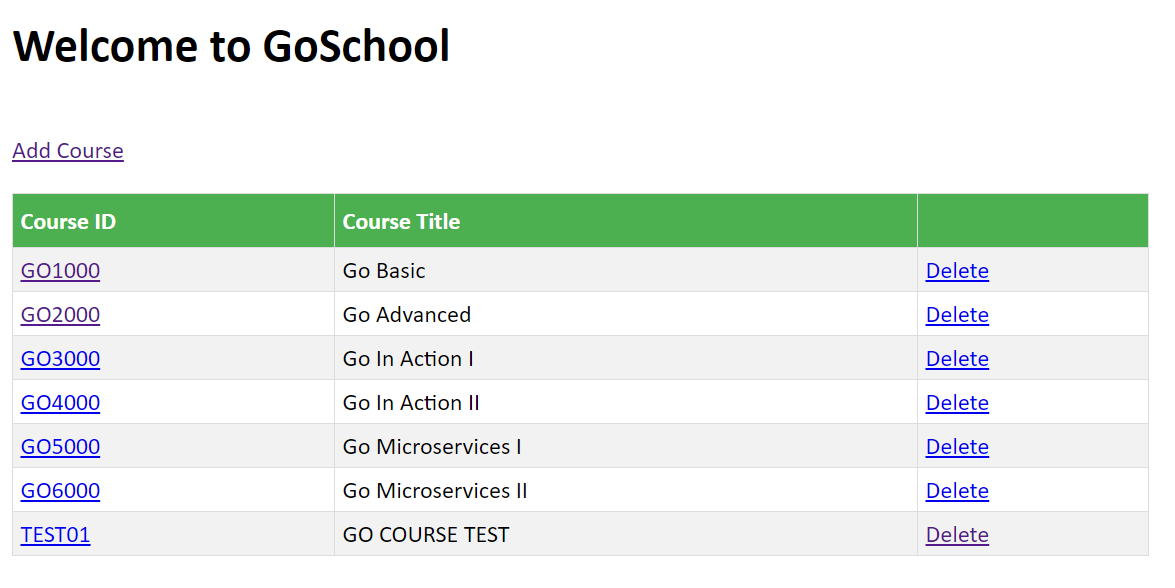
 



When the course is added successfully, checks the database to ensure that the record is inserted. For log purpose, the Created\_DT and LastModified\_DT columns are inserted with the current date/time.

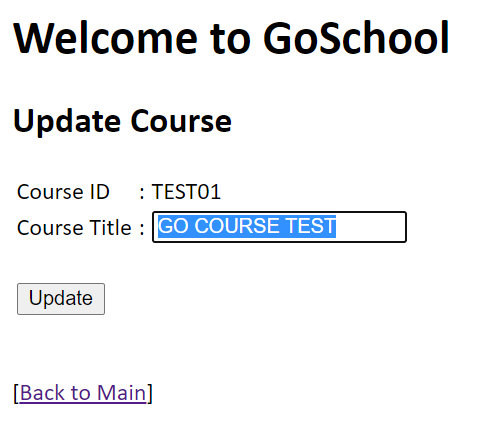
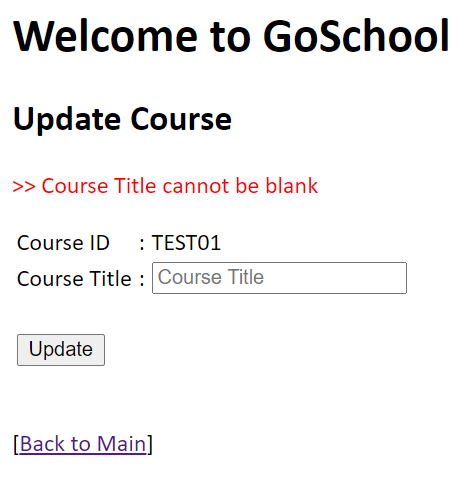


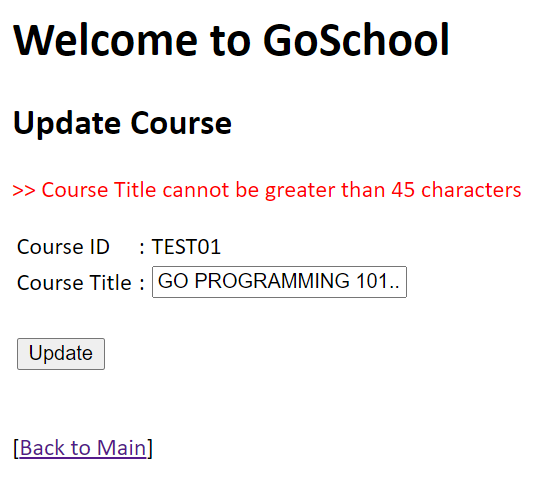
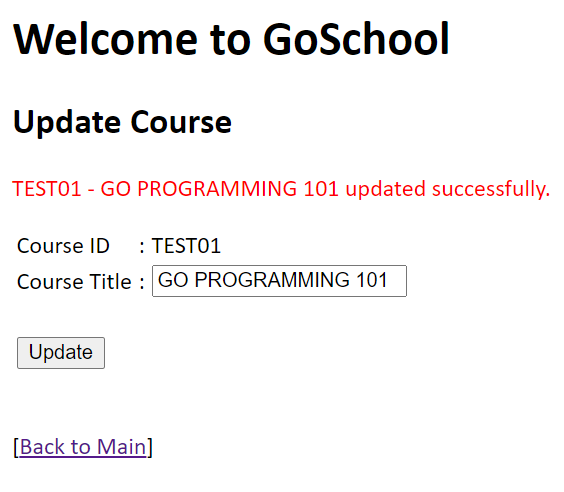
The new record should be displayed in the index page as well.



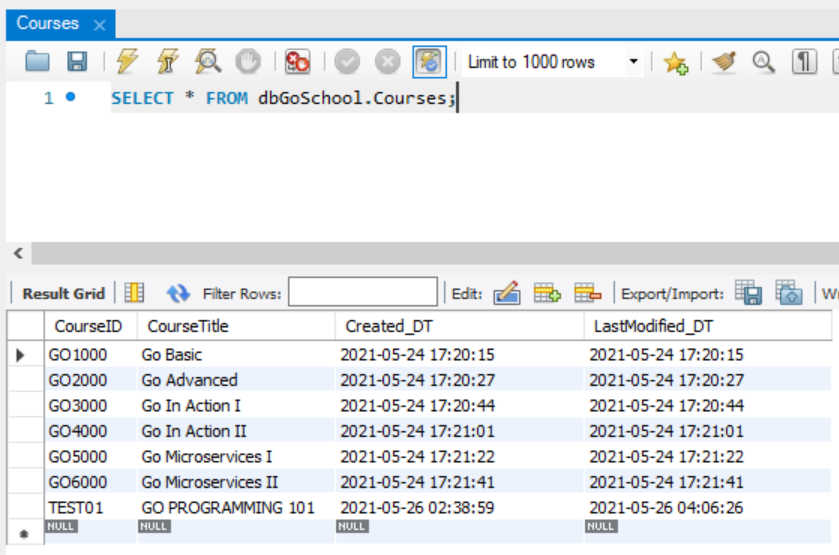
## UPDATE COURSE

At the index page, click on the hyperlink of Course ID to update. Note that Course ID cannot be updated thus it is displayed as a label. Test the text field to ensure data input is valid.

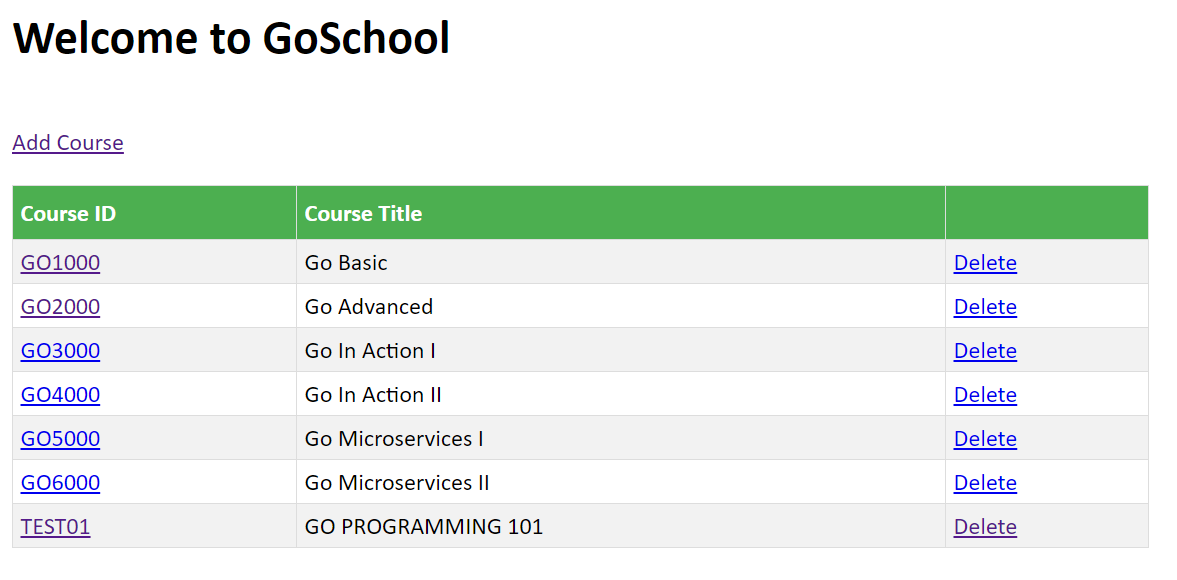
 

After the data is updated successfully, checks the database to ensure that record is also updated properly. For log purposes, the LastModified\_DT column is updated with the current date/time.

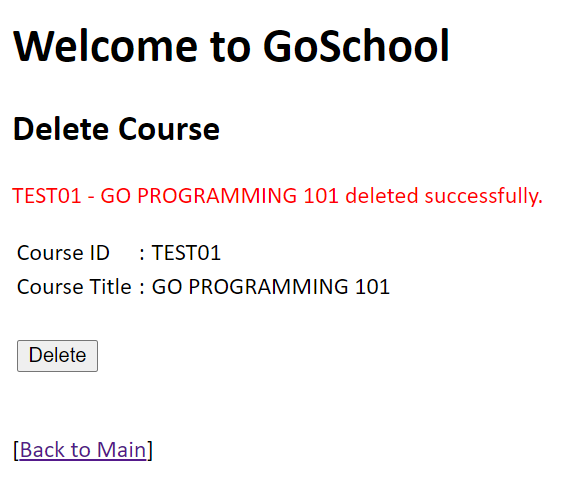


The updated record should be reflected in the index page as well.

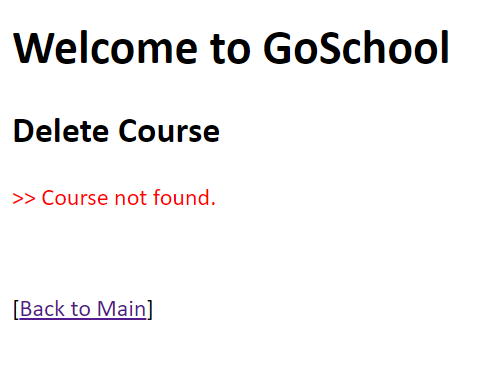


## DELETE COURSE

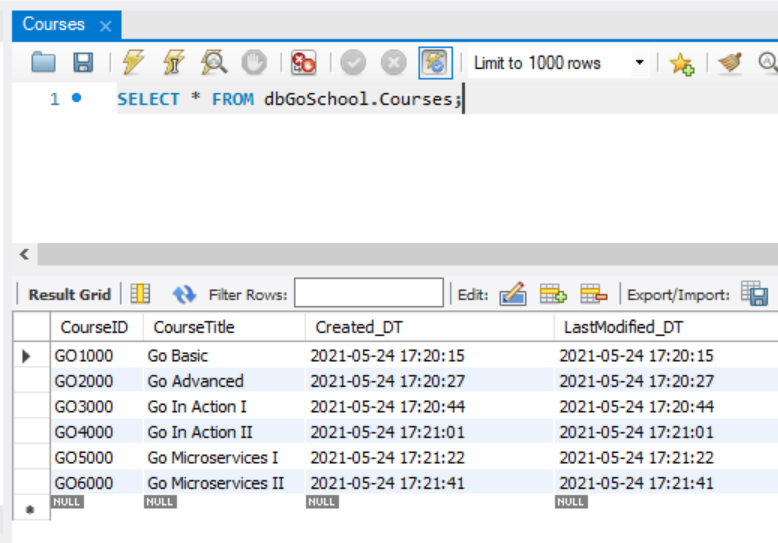
At the index page, click on the ‘Delete’ hyperlink to delete the specific row. The course details will be displayed.

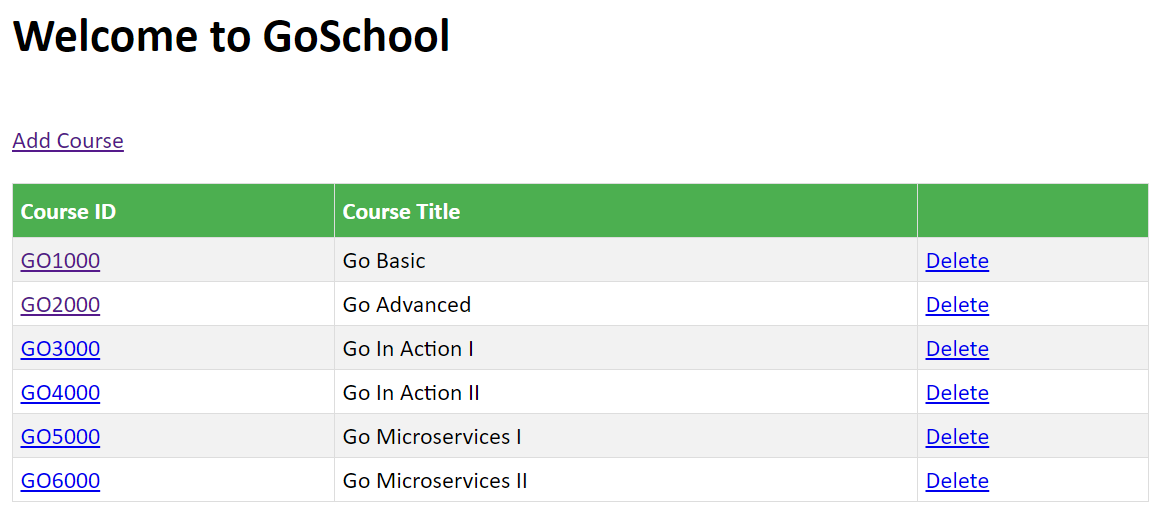
When delete is clicked again, the following error message will be displayed.



Check that the record no longer exists in the database after deletion.



The deleted record should not be reflected in the index page as well.



# REFERENCES

* <https://reqbin.com/>
* <https://github.com/Vishalj32/rest-go-demo>