Overview:

This submission contains the following.

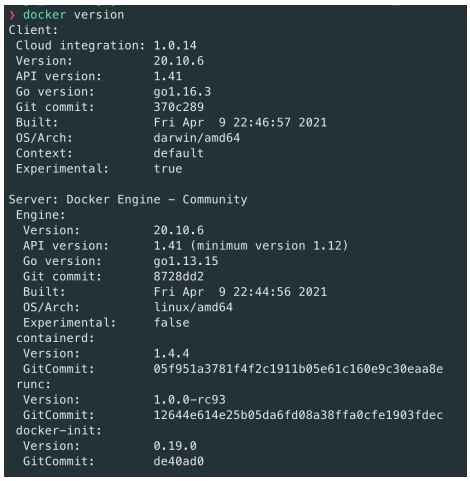
1. api: all relevant .go files for the Courses REST API.
   - The entry point (main function) is located in server.go.
   - Every function mapped to the gorilla mix router is in a separate file by itself (e.g. course.go, index.go)
   - Functions that execute CRUD operations within course.go have been refactorized into separate files (e.g. addCourse.go, viewall.go etc)
   - Contains an api file that was created with the 'go build' command.
2. cli: all relevant .go files for the Courses REST API client.
   - The entry point (main function) is located in client.go.
   - Functions that execute CRUD operations within client.go have been refactorized into separate files (e.g. addCourse.go, viewAllCourses.go etc)
   - Contains a cli file that was created with the 'go build' command.
3. database: an empty folder that will be populated after the setup process.
4. documentation: documentation automatically generated by godoc.
5. Writeup: an overview of the files created for this assignment.
6. database_password.txt: a file used for creating the database.
7. GoMS1.mov: a video guide to aid testing the application.

Concepts implemented:

1. REST API that allows courses to be created, updated, deleted and retrieved.
2. The above-mentioned API stores the course information in a MySQL database deployed on a Docker container.
3. REST API client (command line interface) that allows users to perform CRUD operations on the database.
4. Communication between the API and client, API and MySQL database is carried out over http using the gorilla mux package.
5. The client uses a 3rd party package "github.com/asaskevich/govalidator" to validate user input.

Setting up:

1. Using a Web browser, download Docker for the OS you are using.
   - Mac: https://docs.docker.com/docker-for-mac/install/
   - Windows: https://docs.docker.com/docker-for-windows/install/



2. Using either Terminal (Mac OS) or Command Prompt (Windows), run the following command to confirm Docker is installed.

   docker version

   You should see an output similar to the above.

3. Run the following command to create a Docker container with MySQL installed. Note that you have to be navigate to the GoMS1 submission folder before running this command.
   - MYSQL_ROOT_PASSWORD is set to a random value generated by the openssl rand command and is stored in database_password.txt
   - We use the -v option to bind a local volume to the "/var/lib/mysql" logical directory in the Docker container
   - We map port 5001 to port 3306 in order to connect to the MySQL container.
   - Confirm that there is an empty database subdirectory under the GoMS1 submission folder. If not, create it before proceeding with the following command.

     docker run -d -p 5001:3306 -e MYSQL_ROOT_PASSWORD=$(cat database_password.txt) -v $(pwd)/database:/var/lib/mysql --name GoMS1-MySQL mysql:latest

4. Add MySQL's path to your OS's environment variable if this has not been done. You can do so on Mac OS by running the following command with the appropriate path.

   export PATH=$PATH:/usr/local/mysql/bin

5. Access the MySQL database using the following command, using the password stored in database_password.txt.

   mysql -P 5001 --protocol=tcp -u root -p

6. Create a new user that is used by the assignment's API to access the database, and give it the relevant privileges by running the following command.

   CREATE USER 'gomsuser' IDENTIFIED BY 'password';

   GRANT ALL PRIVILEGES ON * . * TO 'gomsuser';

7. Check that the user has been created with the following command. 'gomsuser' must appear in the output of the command.

   SELECT user from mysql.user;

8. Create a new database and table with the following commands.

   CREATE database goms_db;

   USE goms_db;

   CREATE TABLE Courses (ID int NOT NULL AUTO_INCREMENT PRIMARY KEY, CourseCode varchar (30) UNIQUE, CourseName varchar(30));

Application testing guide:

1. Using Command Prompt (Windows) or Terminal (Mac OS), navigate to the directory where this assignment submission is located at.
2. Run the ‘go build’ command in both the api/ and cli/ subdirectories.
3. On separate command prompt/terminal sessions, run the following commands.

   go run cli (REST API client)

   go run api (REST API)
4. Select the relevant CRUD options in the session running the API client to test out the application. For a more detailed guide for testing the application, please watch GoMS1.mov.