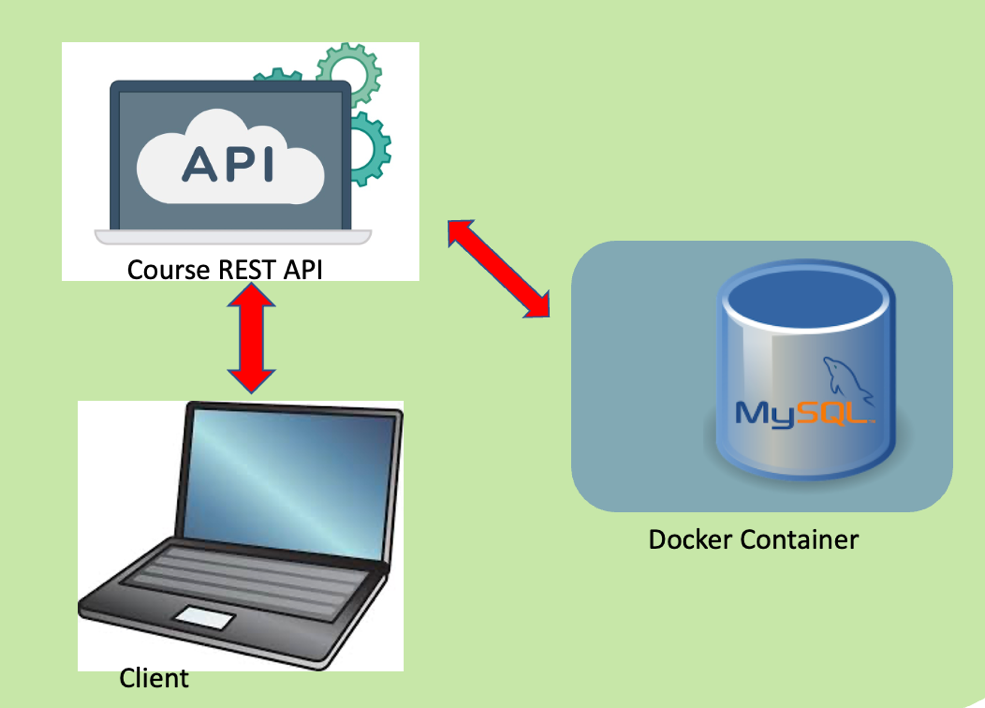
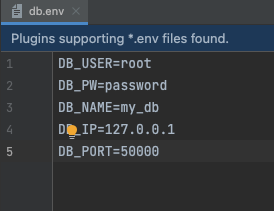
**GO Microservices 1**

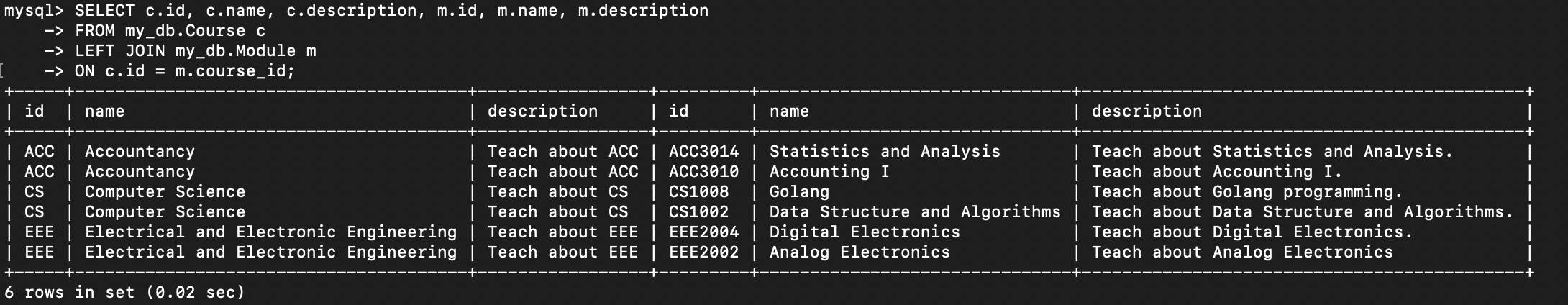
1. **Three Main Components**

****

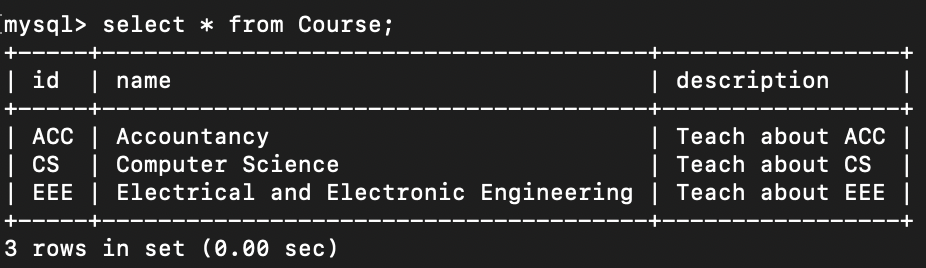
1. **REST API** – an University course REST API is built. Users are able to retrieve, add, update and delete the course and module data from the database via the REST API. API keys are used for securing the REST API which only allow authenticated user with their respective API key to use the service. The API keys are generated using 32 randomly generated bytes with the SHA256 hashing algorithm. An env file is used to store all the database information like user, password, database name as well as database IP and port so that all these important information is not hard-coded in the go file so that this information are not exposed as shown below.

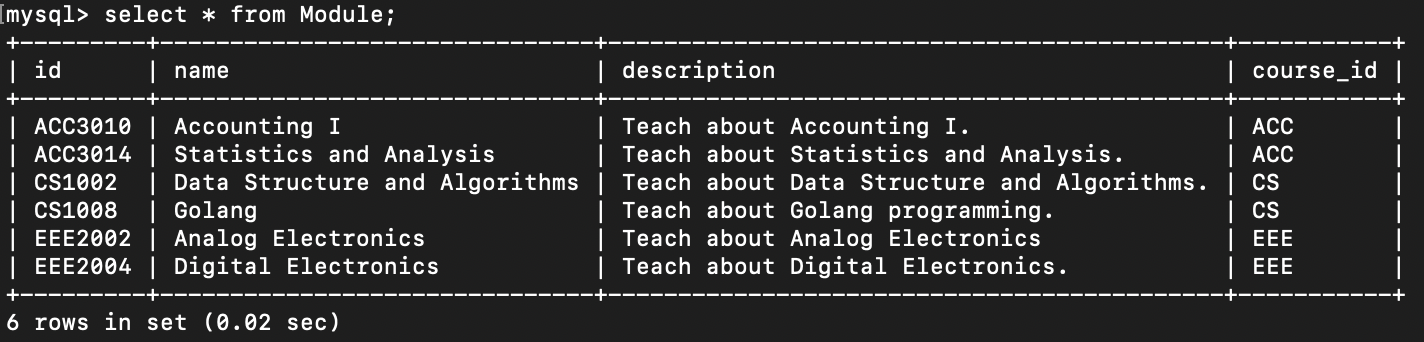


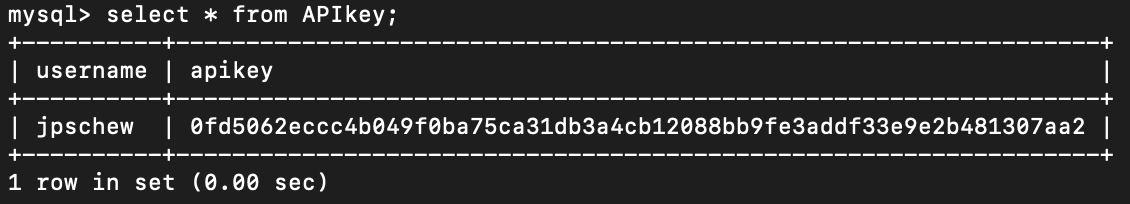
Each course will consist of information such as the course code, course name, description of the course as well as the modules for the respective courses as shown below.



1. **mySQL in docker container –** docker container is utilized to run mySQL database as a microservice and can communicate with the REST API for retrieving, adding, updating and deleting of data from/to mySQL database. The course, module as well as the api-key data are stored in the Course, Module and APIkey table respectively which are shown below. These tables are initialized using a Dockerfile with a “create-local-db.sql” file when building up mySQL container.

****

****

****

1. **Client application –** created for user to communicate with the REST API to retrieve, add, update or delete data to/from the database. The client application will ask for the user input and then send the respective url/endpoints to access the data.
2. **Set-up Guide**
   1. **mySQL docker container**
3. First, we need to initialize the Course, Table and APIkey table in “my\_db” database using the Dockerfile stored under the apicourse/database folder.
4. The apicourse/database folder need to include an .sql file to initialize the Tables needed in the database as stated in part i), in this case is “create-local-db.sql”
5. Change the directory to apicourse/database folder and run the following command:

“docker build -t arm64v8/sql:oracle” for mac users

“docker build -t mysql:latest” for window users

1. Start and run the container using mySQL image to create mySQL database inside a container as well as assigning local host port 50000 to connect to port 3306 of container which will connect to mySQL database in the container using the command below:

“docker run --name My-mysql -p 50000:3306 -e MYSQL\_ROOT\_PASSWORD=password -d arm64v8/sql:oracle” for mac users

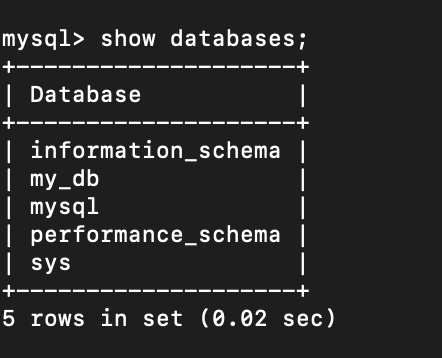
“docker run --name My-mysql -p 50000:3306 -e MYSQL\_ROOT\_PASSWORD=password -d arm64v8/sql:oracle” for window users

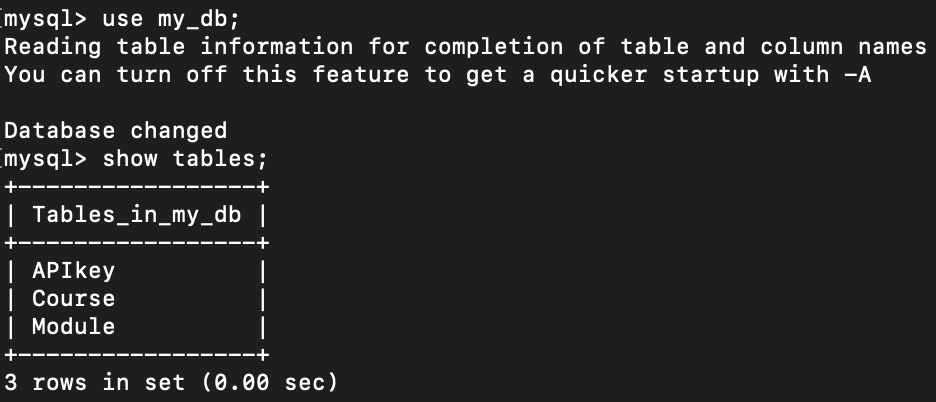
This will also set the container name to “My-mysql” and also set the root password.

1. By now, the container is running and you can use mySQL CLI to query the database in the container by the command below:

“mysql -P 50000 --protocol=tcp -u root -p”

And key in the password assigned in the previous instructions. Now, you can start querying mySQL database in the container as shown below.





* 1. REST API

1. After starting up mySQL container, you can now run the REST API server by navigating to the /courseapi folder and then run the command below:

“go run .”

Take note that you need to start mySQL database before running the REST API, if not the REST API cannot connect to the server.

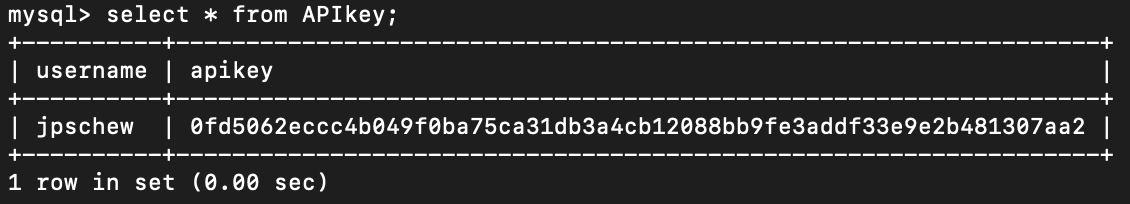
* 1. REST API

1. After the REST API server is running, you can now run the client application to communicate with the REST API by navigating to the /client folder and then run the command below:

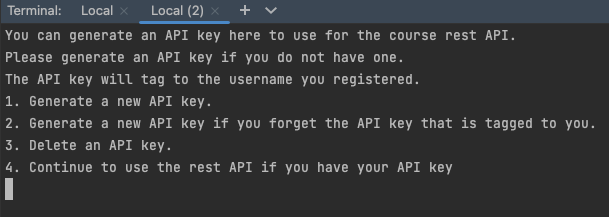
“go run .”

1. **Testing steps**

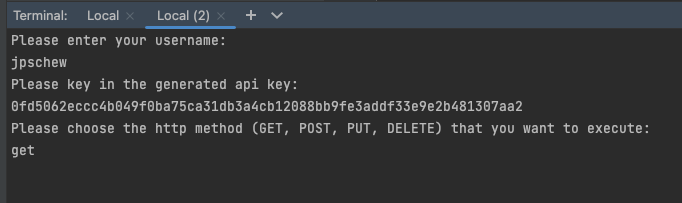
For a start, we can test by using an existing user with an API key that is pre-loaded in mySQL database container as shown below.



When the client program starts, it will prompt the user whether he/she wants to generate a new API key or continue if he/she already has an existing API key. Let’s choose 4 for the testing for all the HTTP methods for a start.

****

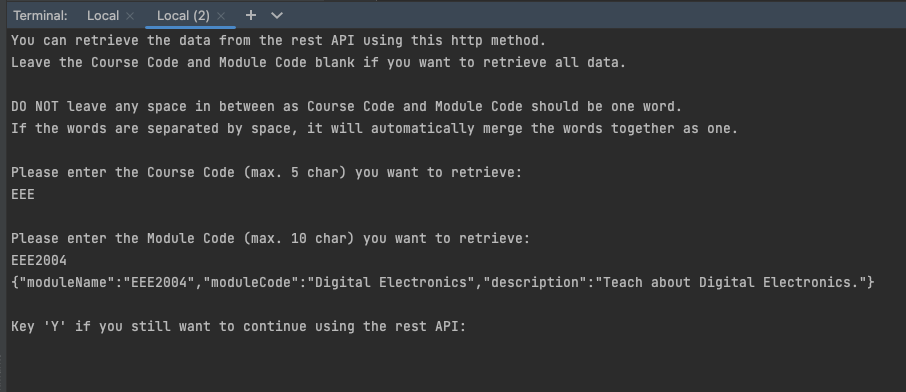
After choosing option 4 which is to continue to use the REST API with an existing API key, it will prompt you for username and api key as shown below. After keying in the username and api key, it will ask you for the HTTP method that you want to use.

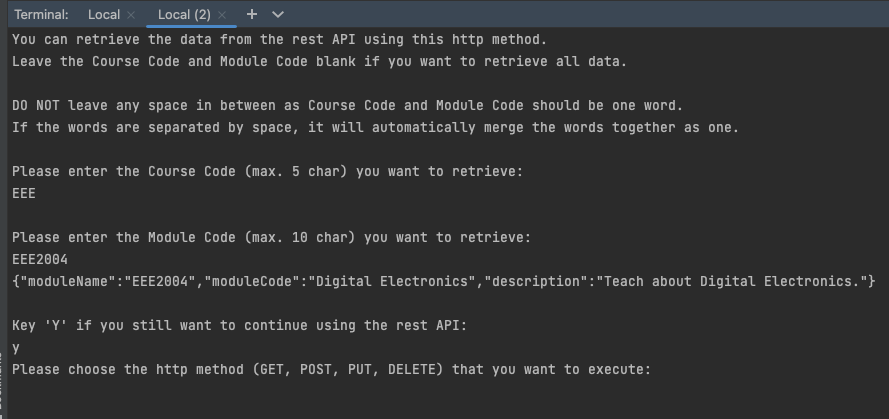
****

* 1. HTTP GET Method

For HTTP GET Method, it will prompt you for Course and Module code. If you leave both the Course and Module code blank, it will fetch you all the course data. If you specify just the Course code, it will fetch you the Course data such as the Course code, name and description together with all the modules information under that particular course. If you specify both the Course code and Module code, it will fetch you the particular module that you are looking for as shown below. However, if the Course/Module code or name is not available, it will send an error message telling you that you have entered both Course/Module code and name.

After each call to the REST API, you can choose to continue by enter “Y” or exit the program by enter any other characters in the second picture below.

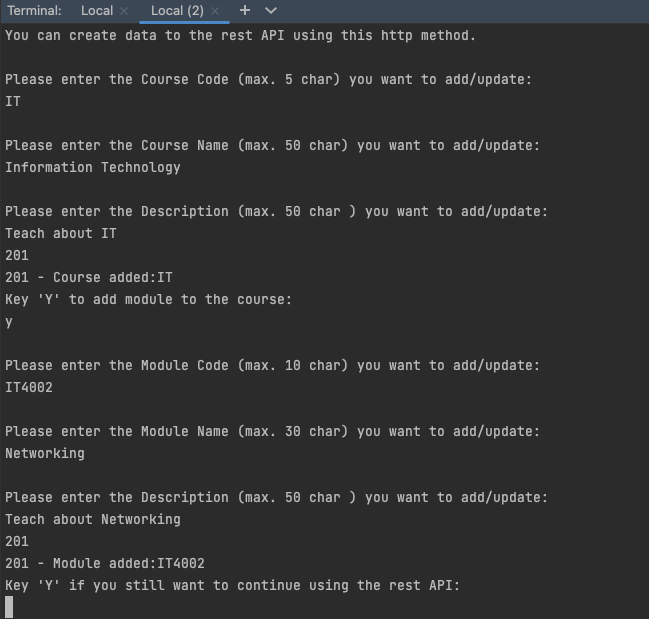




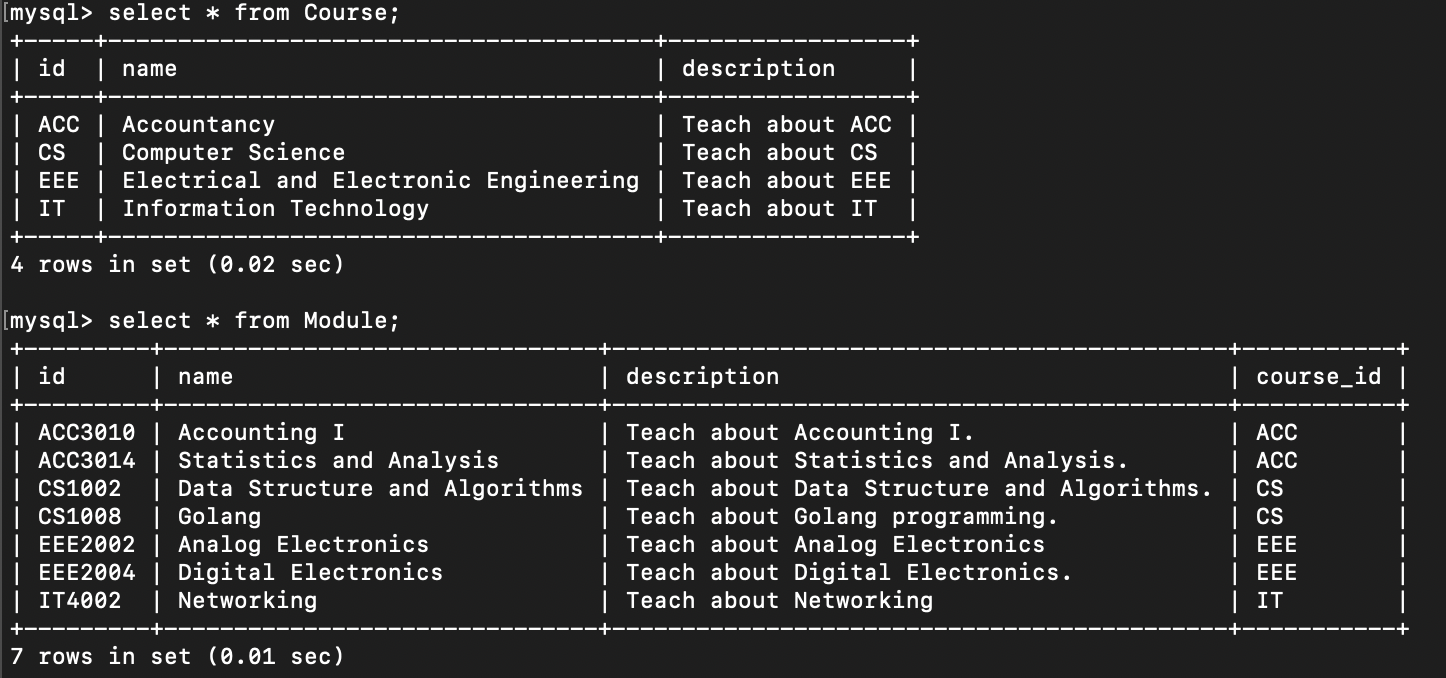
.

* 1. HTTP POST Method

For HTTP POST Method, it will prompt you for Course and Module information to add. If you want to add any Course/Module, the code and name of Course/Module must be present if not it will return an error message.

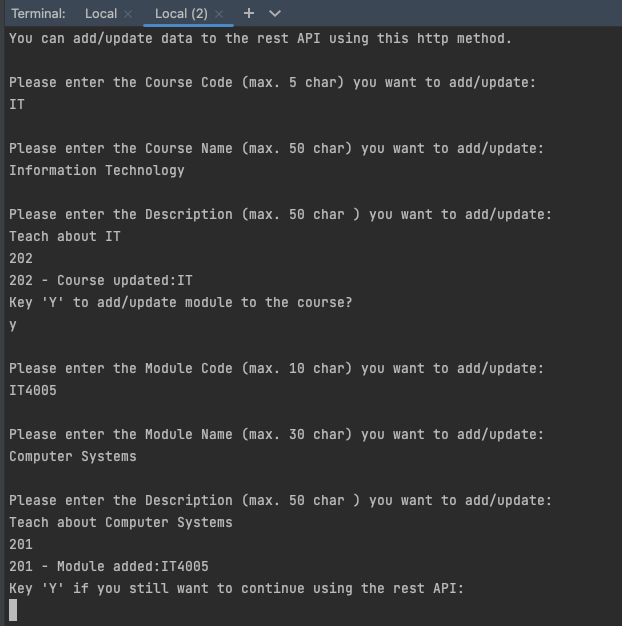


After adding the course/module, you will see that the database will be updated with the new Course and Module in the Course and Module Table respectively as shown below.

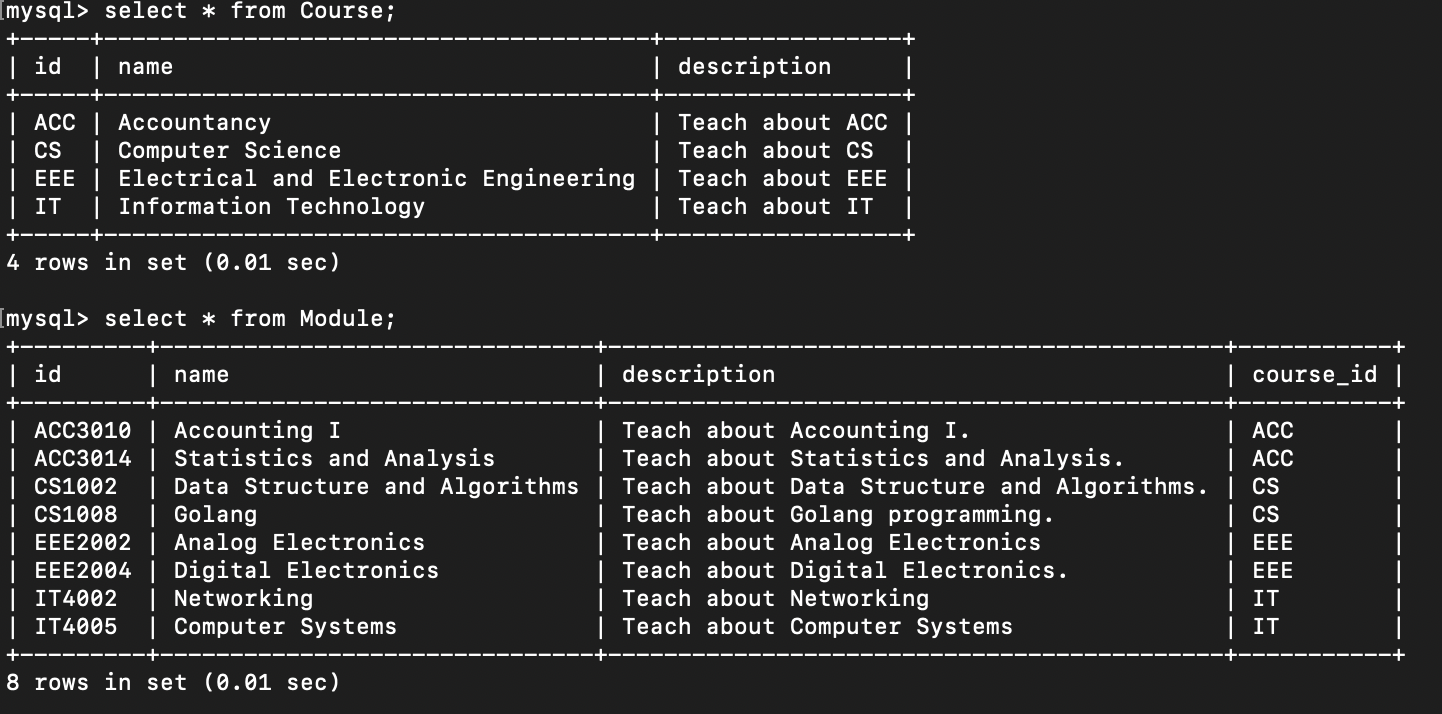


* 1. HTTP PUT Method

For HTTP PUT Method, it will prompt you for Course and Module information to update. If you want to add any Course/Module, the code and name of Course/Module must be present if not it will return an error message.

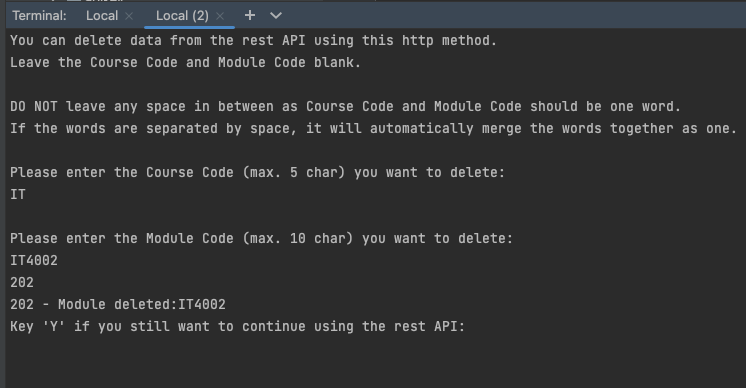


After updating the course/module, you will see that the database will be updated with the new Module in the Module Table as shown below.

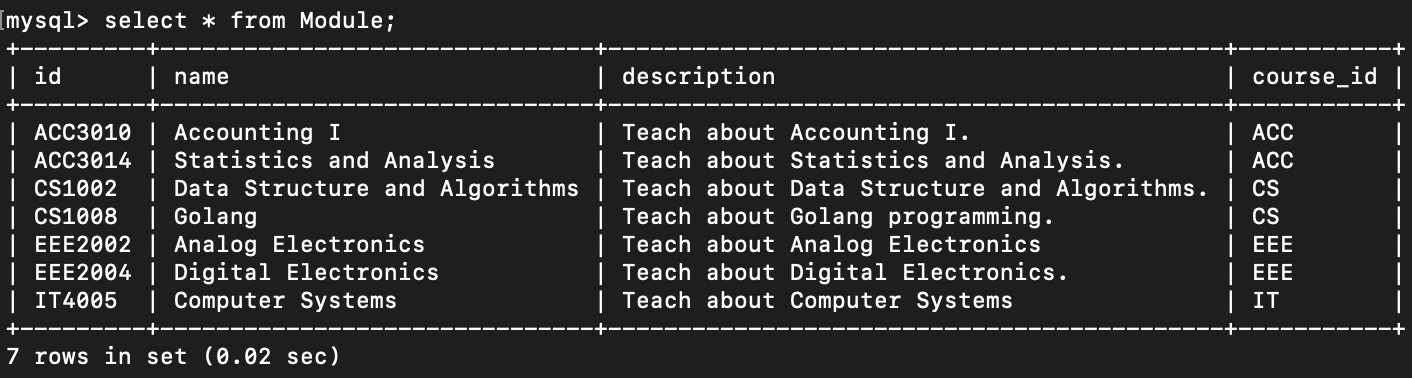


* 1. HTTP DELETE Method

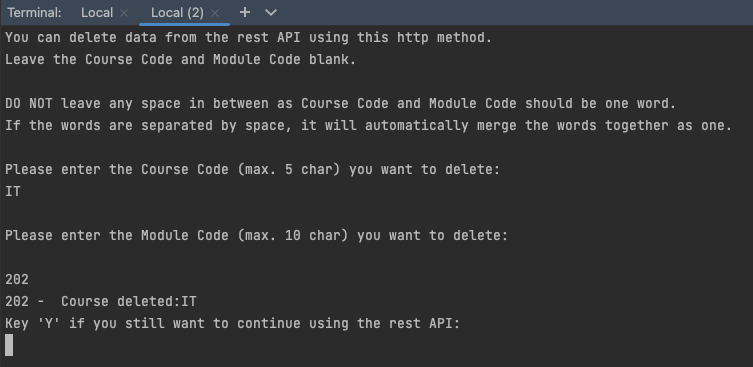
For HTTP DELETE Method, it will prompt you for Course and Module Code to delete. If you include both the Course and Module code, it will delete the module only as shown in below.

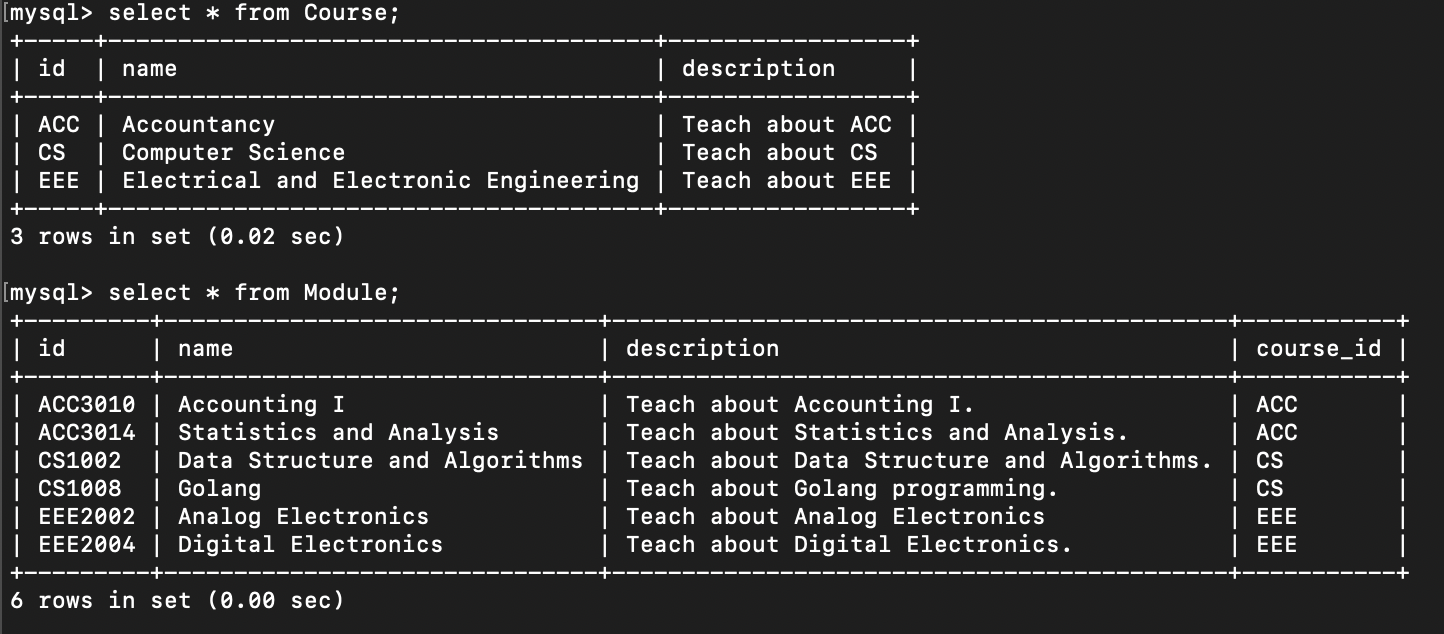


After deleting the module, you will see that the Module Table will be updated in the database with module IT4002 removed as shown below.



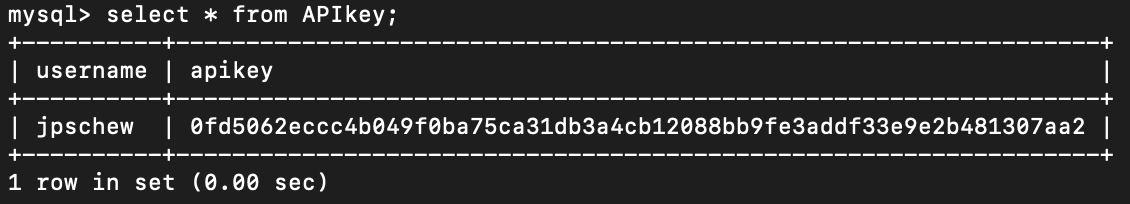
However, if you include only the Course code, it will delete the course along with all the modules associated with the course as shown below.



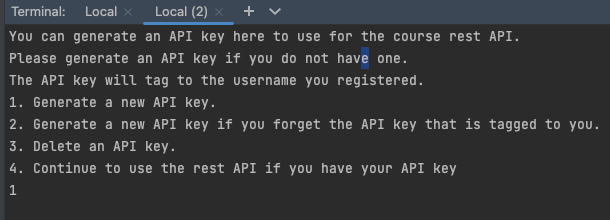


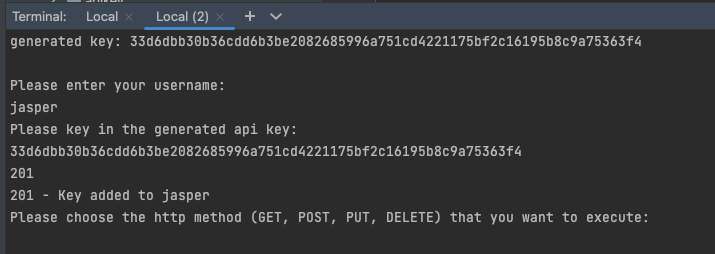
* 1. API Key

For the above testing, we are using the only user with an API key shown below.

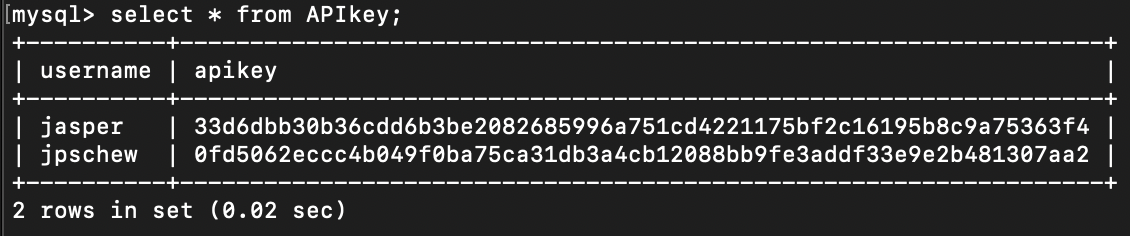


If a new user wants to generate an API key for his/her usage, this can also be done as shown below.

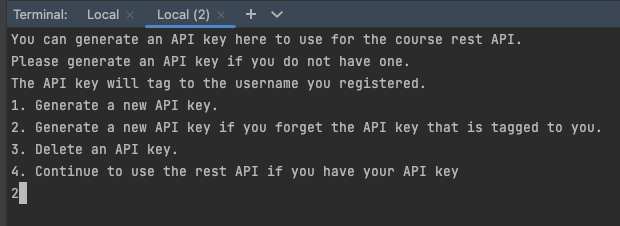


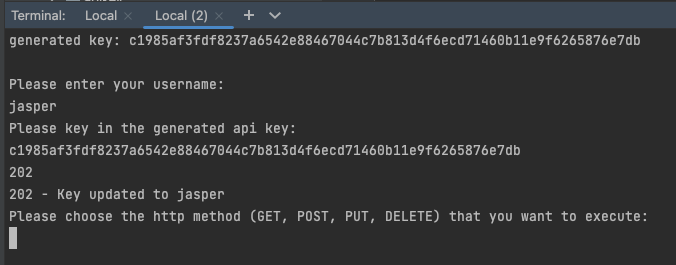


After generating a new API key to new user, he/she can now start using the REST API using the API key generated for him/her as the database has been updated.

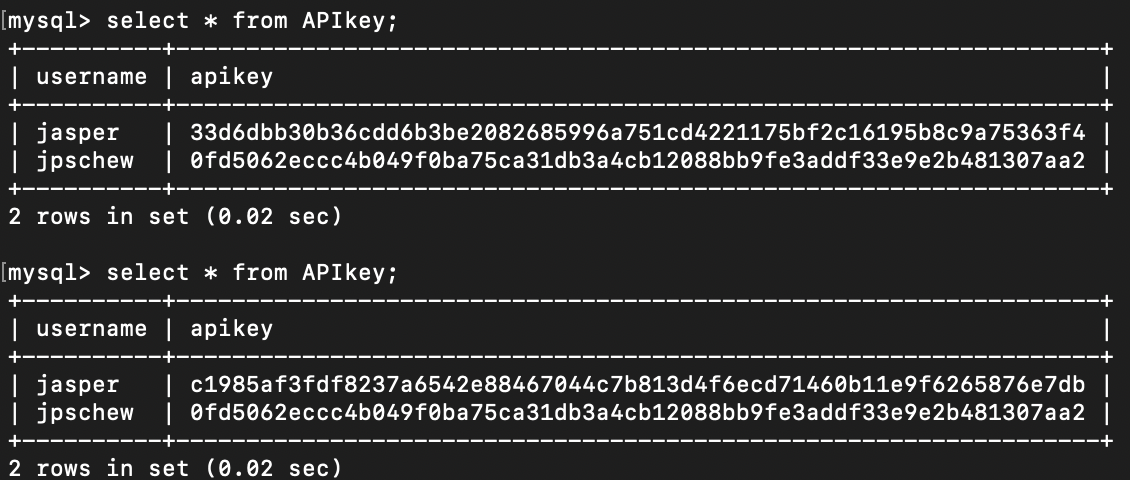


However, if the user forgot your API key, the user is able to generate a new API key to replace his/her existing one as shown below.

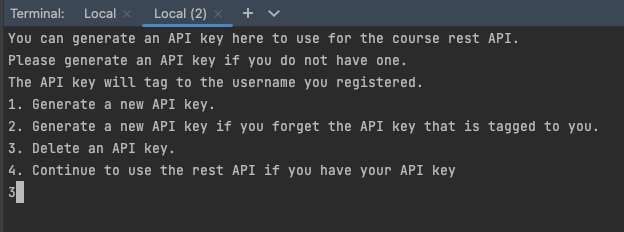


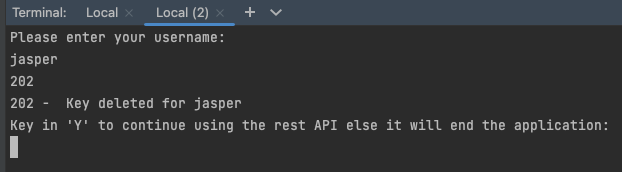


After requesting for a new API key to replace old one, he/she can now start using the REST API using the updated API key that is generated for him/her as the database has been updated.



Lastly, an API key can also be deleted as shown below.





After deleting the API key tagged to “jasper”, the API key is also deleted from the APIkey table in the database.

