Lab 03. Integrate with MongoDB

In this lab, we will integrate our webservice with MongoDB

Objectives:

- Learn how to connect to MongoDB
- Learn how to find documents from MongoDB
- Get familiar with API routes, path parameters and query strings

A. Create a New Project

1. Create a folder `lab03` with a new python file `main.py` in it.
2. Implement a home route in `main.py`.
3. Execute `main.py` file to run the web server.

Setup MongoDB

4. Download JSON file `books.json` file from https://github.com/qinjie/sample-data/blob/master/books.json.
5. In MongoDB Cloud, import `books.json` into a collection `books` in a database `demo`.

B. Get from MongoDB

1. Create a file `database.py` with following code
    - Use your own connection string with correct username, password and database

```
from pymongo import MongoClient

# MongoDB attributes
mongodb_url =
'mongodb+srv://root:qwer1234@cluster0.hlixs.mongodb.net__/demo?
retryWrites=true&w=majority'

try:
  client = MongoClient(mongodb_url)
  db = client['demo']
  print('Connected to MongoDB')
except Exception as e:
  print(repr(e))
  raise Exception('Error in MongoDB connection.')
```

2. Import `database.py` file in `main.py`.

```
import database
```

List all Books

1. Add a route to return all books in the database.
   - Use `collection.find()` function to get a cursor. Convert cursor to lists to get all items.
   - The `_id` value is an `ObjectId` type which needs to be converted to String before sending to client.

```
@app.get('/books')
def list_books():
  cursor = database.db.books.find({})
  result = []
  for item in cursor:
    item['_id'] = str(item['_id'])
    result.append(item)
  cursor.close()
  return result
```

Exercise

4. Add query strings `skip` and `limit` to the `GET /books` route and use them in `collection.find()` function.

```
def list_books(skip: int=0, limit: int=0):
  cursor = database.db.books.find().skip(skip).limit(limit)
  ...
```

Get an Item

5. Implement the `GET /books/{book_id}` route to return a book by its ID.
   - We have to convert `_id` from string back to `ObjectId` type before pass it into `find_one()` function.

```
from bson import ObjectId

@app.get('/books/{book_id}')
def get_book_by_id(book_id: str):
  book = database.db.books.find_one({'_id': ObjectId(book_id)})
  if book:
    book['_id'] = str(book['_id'])
    return book
  else:
    return JSONResponse(f'Book not found: {book_id}', 404)
```