IEMS5722 Mobile Network Programming and Distributed Server Architecture (Fall 2024)
Assignment 4
Expected time: 6 hours

Learning outcomes:
1. To learn how to develop a server-side application using Python and FastAPI 2. To learn how to develop APIs for mobile apps 3. To learn how to use HTTP for communication between server and client 4. To learn how to connect the server-side application to MongoDB Atlas or Mongoose

Instructions:
1. Do your own work. You are welcome to discuss the problems with your fellow classmates. Sharing ideas is great, and do write your own explanations/comments.
2. If you use help from the AI tools, e.g. ChatGPT, write clearly how much you obtain help from the AI tools. No marks will be taken away for using any AI tools with a clear declaration.
3. All work should be submitted onto the blackboard before the due date.
4. You are advised to submit a .pdf file and a .zip file containing all your work.
    - 1155xxxxxx_Assignment4.pdf: The short report for your work. We will grade your work based on the short report.
    - 1155xxxxxx_Assignment4.zip: The zip file containing your Android Studio project and your Python FastAPI script. In general, we will not check this archive in grading. In case, we found some problems in your report (meaning that you already lost some points in your report), we will refer to your project source code.
    - 1155xxxxxx is your student ID.
5. Do type/write your work neatly. If we find some problems in the screenshots in your report, you will lose some points, even if those contents are in the source files.
6. If you do not put down your name, student ID in your submission, you will receive a 10% mark penalty out of the assignment 4.
7. Late submissions will receive a 30% mark penalty.
8. This assignment accounts for 7% of your final grade.
9. Due date: 7th November, 2024 (Thursday) 23:59.

Instructions:

In this assignment, you will try to develop the **server-side application** for the mobile app developed in Assignment 3. You should setup a server application using **Python** and **FastAPI**, and deploy the application using **Uvicorn**. The API you develop in this assignment should be the same as the ones provided in Assignment 3. Hence, the app developed in Assignment 3 should work with the server application developed in this assignment without any modification except the URL of the APIs.

In this assignment, you will have to use the following software applications: FastAPI, Uvicorn ASGI server, NoSQL Database (either MongoDB Atlas or Mongoose) and various Python modules.

APIs:

You will need to implement the following three APIs when implementing the app in this assignment. These APIs are highly similar to the APIs in the assignment 3 APIs. However, you can host the service in the localhost, you can choose any PORT_NUM for your service, and you need to design meaningful error messages in case of errors.

API	**GET** http://localhost:PORT_NUM/get_chatrooms
Descriptions	For retrieving a list of chatrooms from the server
Input Parameters	No input parameter is required
Example	http://DOMAINNAMETBC/get_chatrooms
Sample Output	`{ "data": [ { "id": 3, "name": "Chatroom 2" }, { "id": 2, "name": "General Chatroom" } ], "status": "OK" }`

API	**GET** http://localhost:PORT_NUM/get_messages
Descriptions	For retrieving a list of messages in a specific chatroom
Input Parameters	chatroom_id (the ID of the chatroom)
Example	http://DOMAINNAMETBC/get_messages?chatroom_id=2

Sample Output	(see below)

```
{
  "data": {
    "messages": [
      {
        "message": "5722",
        "name": "Danny",
        "message_time": "2024-09-29 19:36",
        "user_id": 1
      },
      {
        "message": "distributed system",
        "name": "Danny",
        "message_time": "2024-09-29 19:36",
        "user_id": 1
      },
      {
        "message": "scalable system",
        "name": "Danny",
        "message_time": "2024-09-29 19:36",
        "user_id": 1
      },
      {
        "message": "mobile app",
        "name": "Danny",
        "message_time": "2024-09-29 19:36",
        "user_id": 1
      },
      {
        "message_id": "test",
        "name": "Danny",
        "message_time": "2024-09-29 19:36",
        "user_id": 1
      }
    ],
  },
  "status": "OK"
}
```

API	**POST** http://localhost:PORT_NUM/send_message
Descriptions	For sending a message in a specific chatroom
Input Parameters	chatroom_id (the ID of the chatroom) user_id (the unique ID of the user, use your student ID for now) name (the displayed name of the user) message (the message input by the user)

Example	**POST** the following to the API: chatroom_id=2&user_id=1155123456&name=Peter&message=hi

Sample Output	`{`
	`  "status": "OK"`
	`}`

In case any of your input parameters is invalid, the API will return **{ "message": "ERROR_MESSAGE", "status": "ERROR"}**. Therefore, you should check whether **{"status": "OK"}** before you continue to use the data returned by the APIs. Meanwhile, you need to design meaningful error messages in this assignment, e.g. "message": "name is exceeding 20 characters".

Tasks:

1. Setup the NoSQL database, either MongoDB Atlas or Mongoose. Design the collections for the NoSQL database. (20%)
2. Design the APIs for the three methods in FastAPI. Output the HTTP request to the console for easier grading. (30%)
3. Start the Uvicorn service in the localhost. (10%)
4. Test your APIs. (40%)

Notes:

1. If you decide to use Java or other Kotlin approaches to design your chatroom, or MySQL database system to design your database, you will need to adjust the description of the screenshots on your own.
2. You may use the test server script provided in the assignment 3 on the Blackboard as the start of your work.
3. For the /send_message API used in this assignment, you need to implement the maximum length of the displayed name (“name”) is 20 characters, and maximum length of the message (“message”) is 200 characters.
4. Messages sent to the server via the /send_message API should be stored in your database system.
5. If you use an emulator, you may need to use “10.0.2.2” instead of “localhost”.