IEMS5722 Mobile Network Programming and Distributed Server Architecture (Fall 2024)
Assignment 5
Expected time: 9 hours

Learning outcomes:
1. To learn how to implement push messages in Android using Google Cloud Messaging 2. To learn how to launch a Microsoft Azure web service with a public link 3. To set up a private GitHub repository

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_Assignment5.pdf: The short report for your work. We will grade your work based on the short report.
    - 1155xxxxxx_Assignment5.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 5.
7. Late submissions will receive a 30% mark penalty.
8. This assignment accounts for 9% of your final grade.
9. Due date: 21st November, 2024 (Thursday) 23:59.

Instructions:

In this assignment, you will further develop your mobile app and your server application, such that the server application will be able to push new messages to the Android app using **Firebase Cloud Messaging (FCM)**. Specifically, you will have to perform these tasks in this assignment:

- Register for a Google account, create a project, and set up Firebase Cloud Messaging API
- Extend your Android app to enable FCM, including getting a registration token from FCM
- Extend your server application such that when a new message is submitted to a chat room, all other users (devices) should be notified by using FCM messages

You should register for a Google account, create a project, and enable the Google Cloud Messaging API. You can follow what we discussed in Lecture 7. A detailed guideline can be found at this website: https://firebase.google.com/docs/cloud-messaging/android/client. In summary, you will have to complete these steps:

- Add Firebase Cloud Messaging to your Android project.
- Add code to check for Google Play Services in the main activity's `onCreate` and `onResume` methods.
- Create a class that extends `FirebaseMessagingService` that will obtain a token for FCM.
- Add code to report the received token to your own server application.
- Use the above class that extends `FirebaseMessagingService` to handle FCM messages pushed from the server.
- Generate a **notification** when a push message is received, when the app is in background. Use the **chat room name** as the title of the notification, and put the **message** in the short description field.

You can refer to the sample app at the following URL:
https://github.com/firebase/quickstart-android/tree/master/messaging

Extending the Server Application:

Next, you should extend your server such that it can send out push notifications by sending requests to Google's FCM server when necessary.

First, you should create a new collection in the database, which will be used to store the tokens submitted by the app. You can create a simple collection with the following design, and insert a new record whenever a client has submitted a token.

```
_id: Int, not NULL, and unique
user_id: Int and not NULL
token: String and not NULL
```

Next, you should create a new API to allow the client to report the token. Create an API as specified below.

API	**POST** http://iems5722-chatroomserverapp-1155xxxxxx.azurewebsites.net/submit_push_token
Descriptions	For submitting a token for a specific user
Input Parameters	user_id (the unique ID of the user, use your student ID for now) token (the push token generated for FCM)
Example	POST the following to the API: user_id=1234567890&token=iamadummytokeniems5722
Sample Output	`{ "status": "OK" }`

Setting up a private GitHub repository for your server application:

Next, you should create a GitHub account and create an empty repository with name format: **iems5722-chatroomserverapp-1155xxxxxx**, where 1155xxxxxx is your student ID. Then, you can push your server application to the GItHub.

Meanwhile, you also need to push the **requirements.txt**, specifying all packages used in your application, and **service_account_file.json**, used in the FCM, onto your repository.

Setting up a Microsoft Azure web app:

Next, you should create a Microsoft Azure account using a **student free account**: https://azure.microsoft.com/en-us/free/students. Following the instruction in lecture 9, the Microsoft Azure web app is deployed based on your private GitHub repository.

You need to name your web service with the format: **https://iems5722-chatroomserverapp-1155xxxxxx.azurewebsites.net**, where 1155xxxxxx is your student ID. If the link is unavailable, you may add some more tokens onto the path.

Including the demo path in your deployment:

For testing purposes, include the demo path in your server application.

```
@app.get("/demo/")
async def get_demo(a: int = 0, b: int = 0, status_code=200):
  sum = a+b
  data = {"sum": sum, "date": date.today()}
  return JSONResponse(content=jsonable_encoder(data))
```

Tasks:

1. Setup the FCM on the client side. (15%)
2. Setup the FCM on the server side, with the additional API and collection. (25%)
3. Setup the private GitHub repository with all necessary files. (15%)
4. Setup the Microsoft Azure web app based on your GitHub repository. (15%)
5. Test your web service with multiple phones. (30%)

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. Remember to add the link of your Microsoft Azure server to the MongoDB Atlas, otherwise, your server application cannot access your database system.