IEMS5722 Mobile Network Programming and Distributed Server Architecture (Fall 2024)
Assignment 3
Expected time: 6 hours

Learning outcomes:
1. To learn how to use asynchronous tasks to handle network operations 2. To learn how to send and receive HTTP messages in Android to and from APIs 3. To learn how to handle data encoded in JSON format\

Instructions:

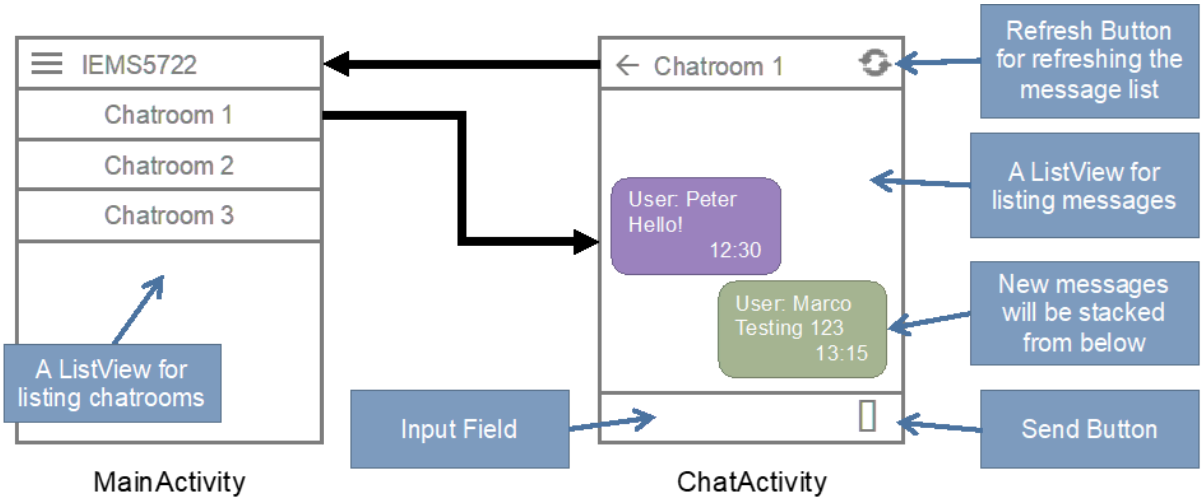
1. Do your own work. You are welcome to discuss the problems with your fellow classmates. Sharing ideas is great, and do write your own explanations/comments.
2. If you use help from the AI tools, e.g. ChatGPT, write clearly how much you obtain help from the AI tools. No marks will be taken away for 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_Assignment3.pdf: The short report for your work. We will grade your work based on the short report.
    - 1155xxxxxx_Assignment3.zip: The zip file containing your Android Studio project. 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 3.
7. Late submissions will receive a 30% mark penalty.
8. This assignment accounts for 6% of your final grade.
9. Due date: 24th October, 2024 (Thursday) 23:59.

Instructions:

In this assignment, you are going to extend your previous assignment by adding networking functionality to the mobile app, such that it will become a working instant messaging app. We will provide you with several APIs (application programming interfaces), which will allow you to 1) get a list of chatrooms, 2) get a list of messages in a chatroom, and 3) send a message to a chatroom.

The server will store any message sent to it through valid API requests, and therefore the user will be able to see the messages when he or she visits the chatroom again.

First of all, you will update the user interface of the messaging app as follows.



As in the previous assignment, the app has two activities. In the first activity, you should now have a list view for listing all available chatrooms. The list of chatrooms should be retrieved by sending a request to an API (details below). When the user clicks on one of the chat rooms, the app should transit to the Chat Activity, which will list the messages that have been sent to the selected chat room so far. The list of messages should also be retrieved by sending a request to an API.

In addition, you should also further develop the app such that the user can send messages to the chatroom. This is done by submitting the message to the server via a specific API.

Finally, you will also add a "**Refresh**" button. When clicked by the user, the button will cause the ListView of messages to refresh and fetch the latest messages from the server. (You will learn how to push any new message to the app in another assignment.)

APIs:

You will need to use the following three APIs when implementing the app in this assignment.

API	**GET** http://DOMAINNAMETBC/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://DOMAINNAMETBC/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	{
	"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://DOMAINNAMETBC/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 "status": "ERROR". Therefore, you should check whether "status": "OK" before you continue to use the data returned by the APIs.

Tasks:

1. Able to retrieve and list Chatrooms automatically. (20%)
2. Clicking on a chatroom in MainActivity goes to the correct Chatroom. (20%)
3. Able to retrieve messages and refresh messages in the Chatroom. (20%)
4. Able to use asynchronous tasks (e.g. coroutine) to handle both GET and POST requests. (20%)
5. Able to send messages to a specific Chatroom and update UI correctly. (20%)

Notes:

1. If you decide to use Java or other Kotlin approaches to design your chatroom, you will need to adjust the description of the screenshots on your own.
2. Check the IP of the test server (DOMAINNAMETBC) on the course Blackboard. You can access the test server via IE VPN only.
3. The test server script is provided on the Blackboard as well, so that you can use localhost to test on your own.
4. For the /send_message API used in this assignment, the maximum length of the displayed name (“name”) is 20 characters, and maximum length of the message (“message”) is 200 characters.
5. Messages sent to the server via the /send_message API will not be stored in the server, to avoid an excessive number of messages to be stored in the server.
6. For simplicity sake, only minimum error checking is performed on the server. Do not deliberately hack, attack or break the system (e.g., via SQL injection). Any attempt to do so will result in zero mark in the assignment and/or other disciplinary actions.