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

Name: Zhang Xinyu
Student ID:1155203383

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:

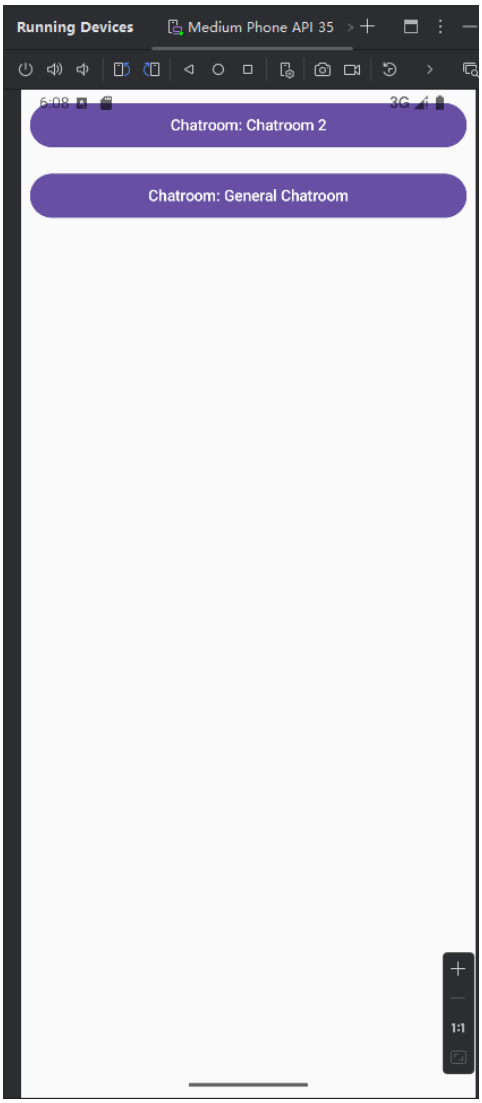
1. Do your own work. You are welcome to discuss the problems with your fellow classmates. Sharing ideas is great, and do write your own explanations/comments.
2. If you use help from the AI tools, e.g. ChatGPT, write clearly how much you obtain help from the AI tools. No marks will be taken away for 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.

Basic information:
My phone model is:
I use a virtual / physical phone.
Virtual phone

Tasks:

1. Able to retrieve and list Chatrooms automatically. (20%)

Screenshot(s) for emulator on the MainActivity:

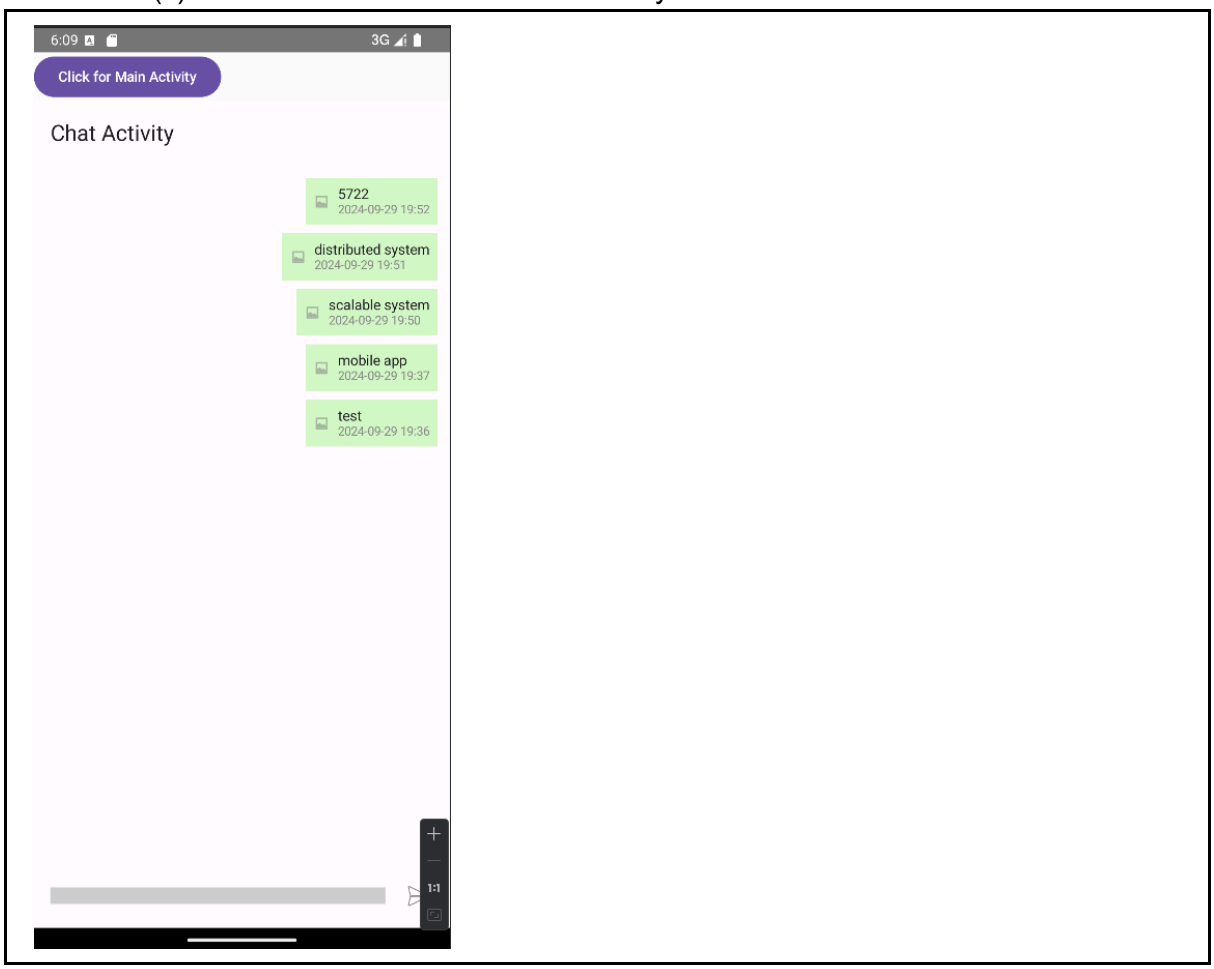


Screenshot(s) for code segment on the GET methods related to listing Chatrooms:

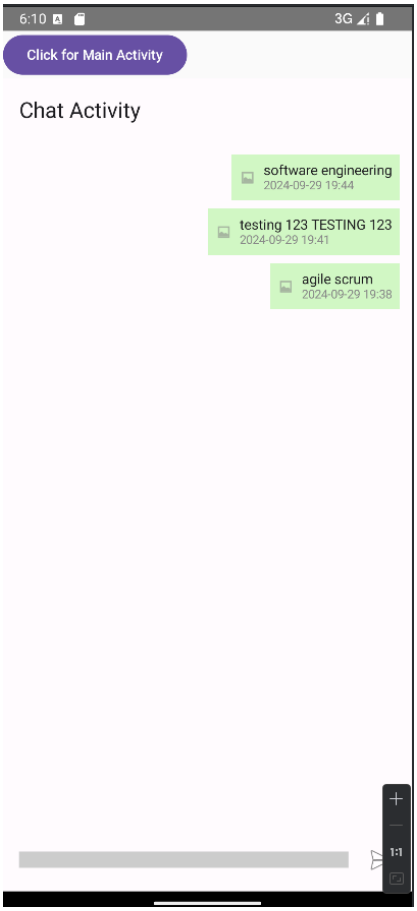
```
// State to hold chatrooms
val chatrooms = mutableStateListOf<Pair<Int, String>>()
CoroutineScope(Dispatchers.IO).launch {
    val infJSON = GET( url: "http://10.0.2.2:8000/get_chatrooms/")
    val jsonObject = JSONObject(infJSON)
    val dataArray = jsonObject.getJSONArray( name: "data")
    for (i in 0 ≤ until < dataArray.length()) {
        val chatroom = dataArray.getJSONObject(i)
        val id = chatroom.getInt( name: "id")
        val name = chatroom.getString( name: "name")
        chatrooms.add(id to name)
    }
}
setContent {
    ChatroomButtons(chatrooms)
}
```

2. Clicking on a chatroom in MainActivity goes to the correct Chatroom. (20%)

Screenshot(s) for emulator on different ChatActivity:



Screenshot(s) for code segment on the intent from MainActivity to ChatAcitivity:



3. Able to retrieve messages and refresh messages in the Chatroom. (20%)

Screenshot(s) for code segment on GET method related to retrieving messages:

```
class ChatActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // Get the chatroomId from the Intent
        val chatroomId = intent?.getIntExtra( name: "chatroomId",  defaultValue: -1) ?: -1

        val messages = mutableStateListOf<Message>()
        CoroutineScope(Dispatchers.IO).launch {
            val infJSON = GET( url: "http://10.0.2.2:8000/get_messages/?chatroom_id=$chatroomId")
            val jsonObject = JSONObject(infJSON)
            val dataArray = jsonObject.getJSONObject( name: "data")
            val messageArray = dataArray.getJSONArray( name: "messages")
            for (i in 0 ≤ until < messageArray.length()) {
                val chatroom = messageArray.getJSONObject(i)
                val message = chatroom.getString( name: "message")
                val name = chatroom.getString( name: "name")
                val message_time = chatroom.getString( name: "message_time")
                val user_id = chatroom.getInt( name: "user_id")
                messages.add(Message(message, message_time,  isUser: user_id == 1))
            }
        }
        setContent {
            val context = LocalContext.current
            Column { Button(onClick = {
                val intent = Intent(context, MainActivity::class.java)
                context.startActivity(intent)
            }) {
                Text(text = "Click for Main Activity")
            }

                ChatScreen(messages, chatroomId,  user_id: 1) }
        }
    }
}
```

Screenshot(s) for code segment on the refresh button:

4. Able to use asynchronous tasks (e.g. coroutine) to handle both GET and POST requests. (20%)

Screenshot(s) for code segment on the coroutine on getting chatroom:

```
val messages = mutableStateListOf<Message>()
CoroutineScope(Dispatchers.IO).launch {
    val infJSON = GET( url: "http://10.0.2.2:8000/get_messages/?chatroom_id=$chatroomId")
    val jsonObject = JSONObject(infJSON)
    val dataArray = jsonObject.getJSONObject( name: "data")
    val messageArray = dataArray.getJSONArray( name: "messages")
    for (i in 0 ≤ until < messageArray.length()) {
        val chatroom = messageArray.getJSONObject(i)
        val message = chatroom.getString( name: "message")
        val name = chatroom.getString( name: "name")
        val message_time = chatroom.getString( name: "message_time")
        val user_id = chatroom.getInt( name: "user_id")
        messages.add(Message(message, message_time, isUser: user_id == 1))
    }
}
```

Screenshot(s) for code segment on the coroutine on getting messages:

```
// State to hold chatrooms
val chatrooms = mutableStateListOf<Pair<Int, String>>()
CoroutineScope(Dispatchers.IO).launch {
    val infJSON = GET( url: "http://10.0.2.2:8000/get_chatrooms/")
    val jsonObject = JSONObject(infJSON)
    val dataArray = jsonObject.getJSONArray( name: "data")
    for (i in 0 ≤ until < dataArray.length()) {
        val chatroom = dataArray.getJSONObject(i)
        val id = chatroom.getInt( name: "id")
        val name = chatroom.getString( name: "name")
        chatrooms.add(id to name)
    }
}
setContent {
    ChatroomButtons(chatrooms)
}
```

Screenshot(s) for code segment on the coroutine on sending messages:

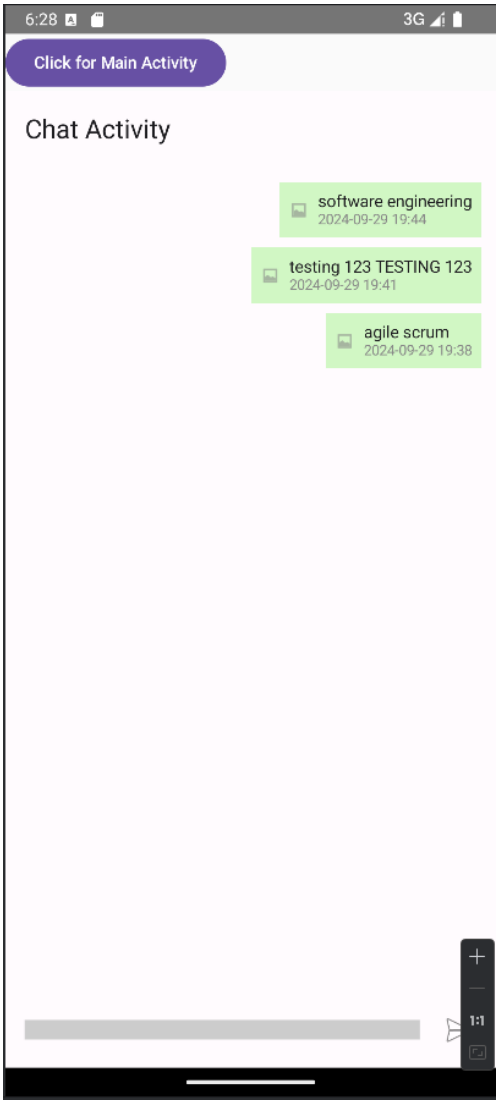
```
val messages = mutableStateListOf<Message>()
CoroutineScope(Dispatchers.IO).launch {
    val infJSON = GET( url: "http://10.0.2.2:8000/get_messages/?chatroom_id=$chatroomId")
    val jsonObject = JSONObject(infJSON)
    val dataArray = jsonObject.getJSONObject( name: "data")
    val messageArray = dataArray.getJSONArray( name: "messages")
    for (i in 0 ≤ until < messageArray.length()) {
        val chatroom = messageArray.getJSONObject(i)
        val message = chatroom.getString( name: "message")
        val name = chatroom.getString( name: "name")
        val message_time = chatroom.getString( name: "message_time")
        val user_id = chatroom.getInt( name: "user_id")
        messages.add(Message(message, message_time, isUser: user_id == 1))
    }
}
```

5. Able to send messages to a specific Chatroom and update UI correctly. (20%)

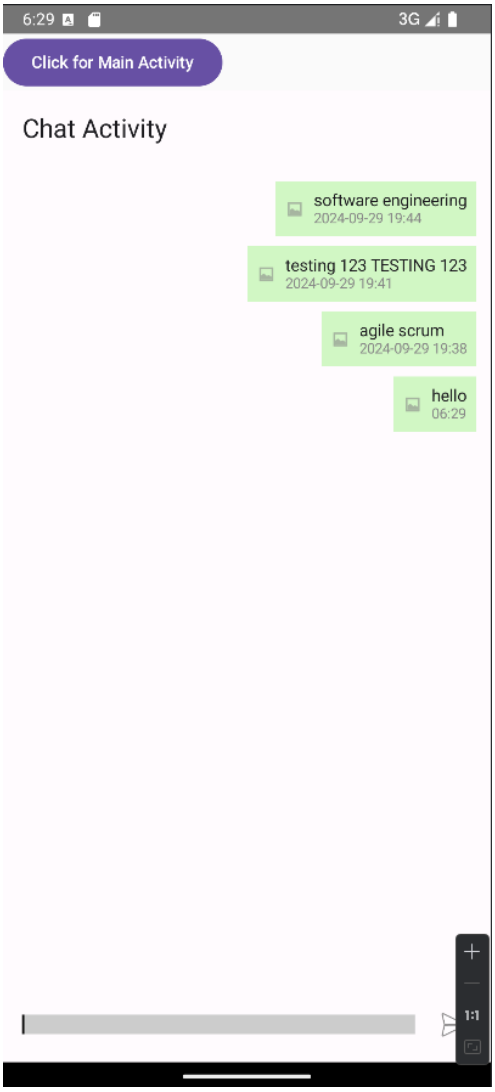
Screenshot(s) for emulator on the ChatActivity: (same as task 2)

```
fun sendMessage(inputText: TextFieldValue, messages: MutableList<Message>, chatroom_id: Int, user_id: Int, clearInput
    if (inputText.text.isNotBlank()) {
        val currentTime = getCurrentTime()
        val newMessage = Message(inputText.text, currentTime, isUser: true)
        messages.add(newMessage)

        // Create the data for the POST request
        val postData = PostRequestModel(
            chatroom_id = chatroom_id, // Replace with actual chatroom ID
            user_id = "1155203383", // Replace with actual user ID
            name = "User", // Replace with actual user name
            message = inputText.text
        )

        // Launch a coroutine to send the POST request
        CoroutineScope(Dispatchers.IO).launch {
            val response = POST( url: "http://10.0.2.2:8000/send_message/", postData)
            Log.d( tag: "POST Response", response)
        }

        clearInput(TextFieldValue())
    }
}
```

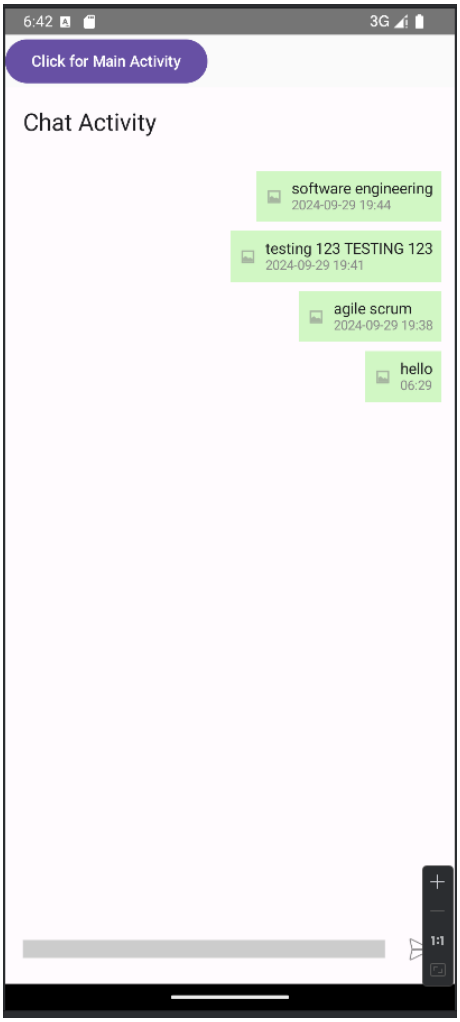
Screenshot(s) for emulator on the ChatActivity after sending a message:



Screenshot(s) for emulator on the ChatActivity after refreshing the chatroom: (same as task 2)



Screenshot(s) for code segment for updating the ChatActivity after sending a message:



Declaration (optional section):

-