Task 1 — Picking the domain for your project (0 marks)

Our project is to develop a de-centralized chat room system like WhatsApp (but de-centralized).

Task 2 — Writing the specification (10 marks)

Our project is to develop a de-centralized chat room system.

In the system, every user can create her/his own chatroom and manage the enrollment of the other users.

In a chatroom, everyone can chat with each other with messages like text, image and actions(like).

Task 3 — CRC model (25 marks)

Framework & Drivers	[Phase 2] Android UI	
	[Phase 0&1] Keyboard Input & Console output	
	[Phase 0&1] Browser (e.g. Chrome)	
Gateway / Controllers	**SystemInOut**	
	UserRepo	
	Server	**UserSrever** **ChatroomServer**
Use Cases	**UserProfile**	
	ChatroomManager	
Entities	**User**	
	Chatroom	
	Message	**TextMessage** **ActionMessage** (likes, read status) **ImageMessage**

TextMessage	Message	**ActionMesage**	Message	**ImageMessage**	Message
TextMessage(String msg, User sender, TimeStamp t)	**ChatroomManager**	**ActionMesage**(**ActionMesage.Action** a, User sender, TimeStamp t)	**ChatroomManager**	**ImageMessage**(String img_path, User sender, TimeStamp t)	**ChatroomManager**

User	
getId getNickname	**UserProfile**

Chatroom	
getId getRoomName setRoomName addMessage getMessages	**ChatroomManager**

UserProfile	
getOwner	**UserServer** **User**

ChatroomManager	
addChatRoom getChatRooms sendTextMessage getMessage(roomID) addUser(roomID)	Chatroom UserProfile

Server	
Listen queryToMap (UI in Phase 0 and Phase 1)	

UserServer	
Listen List owner	**UserProfile**

ChatServer	
Listen List Messages given chatroom id Post messages addUser given user id and chatroom id	**ChatroomManager**

SystemInOut	
Get single line string Interact mode (UI for Phase 0 and Phase 1)	**ChatroomManager**

- Is there enough in your model to satisfy most of your specification?
    - at least three entity classes: yes
    - at least two use case classes: yes

    - at least one controller: yes
    - and at least a basic command line interface: yes
- **Are your cards clearly organized so your TA can easily assess what you have done?**
    - Does each card clearly belong to a layer of Clean Architecture and is that clearly indicated?
    - Did your group find a good balance between too much detail and too little detail when describing each card's responsibilities?
- Are there any places where your CRC model seems to be clearly inconsistent with any of the SOLID principles?

Task 4 — Scenario Walk-Through (15 marks)

1. The Main program launches
2. Try to load user profile
3. If not found, prompt the user for a new one
4. Start the user sever fore reading profile
5. Start the message server for polling and sending messages
6. In the interactive mode of the command line, we can add a new chatroom
7. Each new chatroom produces a chatroom id, with the id, we can send messages in the interactive mode
8. All those messages and user profiles can be retrieved from the browser with a specific port
9. (More details to be added in Phase 1)

Task 5 — Skeleton Program (20 marks)

- Can the code compile and run? If not, you automatically get 0 marks for this part!
    - Your TA should be able to clone your repo and run your code. Make sure you provide enough information that they can do this easily!
- Does the code contain at least one unittest? And does that unittest pass?
- Does the code demonstrate an honest effort at implementing enough code such that your scenario walk-through can be run?
    - We need to see that your program can take in some kind of input, do something with it, and produce some kind of output.
- Does the code have any style warnings in IntelliJ?

Task 6 — Progress Report (30 marks)

Please kindly see above sections for our progress.

Q: what each group member has been working on and plans to work on next?

Junhao is working on the server side of things.

Lilian is working on the repo.

Peter is working on the message manager.

Kruzer is working on the messages.

Jackson is working on the command-line.

Varun is working on the unit test cases.

Q: What has worked well so far with your design as you have started implementing the code?

The server APIs are working well and we plan to add more APIs. We should involve more access control in the ChatroomManager and possibly split out a MessageManger from it, so that the ChatroomManger is mainly responsible for enrollments.

Question for TA:

Should we separate the task on the Chatroom manager to other use cases? How can we achieve that?