# DSCD Assignment 1 Part 2
# README
# Low-level group messaging application using ZeroMQ

This project implements a low-level group messaging application using ZeroMQ. The application consists of a central message server, multiple groups, and users interacting with each other in real time. Users can join or leave groups and exchange messages within those groups.

## Architecture Overview

The architecture comprises a central server, multiple groups, and users. The server maintains the list of groups along with their IP addresses, while users can join or leave groups and exchange messages within those groups.

## Types of Nodes

Message Server:
- Maintains a list of groups and their IP addresses.
- Handles requests from groups to register and from users to fetch the list of available groups.
- Acts as an intermediary between users and groups.

Group:
- Manages users who are part of the group.
- Stores and handles messages sent by users.
- Accepts requests from users to join or leave the group and to fetch or send messages.

User:
- Interacts with groups by joining or leaving them, fetching messages, and sending messages.

## Detailed Explanation

MessageServer ↔ Group

1. MessageServer Registration:
   - A group server requests the message server to register itself.
   - The message server registers the group server and sends a SUCCESS response.

User ↔ MessageServer

1. GetGroupList:
   - A user requests the message server to provide the list of live groups.
   - The message server returns a list of live servers.

User ↔ Group

1. joinGroup:
   - A user requests to join a group by sending its UUID.
   - The group accepts the user and adds it to its list of users, sending a SUCCESS response.

2. LeaveGroup:
   - A user requests to leave the group.
   - The group removes the user from its list of users and sends a SUCCESS response.

3. GetMessage:
   - A user requests all messages from the group after a specified timestamp.
   - The group sends all corresponding messages to the user.

4. SendMessage:
   - A user sends a message to the group.
   - The group accepts the message and updates it with the current timestamp, sending a SUCCESS response.

## Evaluation

Each project submission will be evaluated by running the files from the deliverables submitted. The TAs will run the files in the following order:

1. `message.py`
2. `group.py`
3. `user.py`

Ensure that all functionalities are correctly implemented and print statements match the

expected outputs.

## Setup and Running Instructions

1. Clone this repository to your local machine.
2. Ensure ZeroMQ is installed on your system.
3. Run the files mentioned above in the specified order to test the application.