Design Specification

Content Module (Under Infrastructure Team)

-Prepared by Talha Yaseen (111701019)

Introduction

In this project, there is a requirement of a chat window where any client can type messages and that will be displayed over all other active clients machines along with a profile image of clients who are sending messages and the time at which clients are sending messages. And these things will be done by the Content Module which is going to cooperate with the UI module and the Networking module.

When any new user joins, the UI module receives a username and profile image of that user. And the UI module passes that username and corresponding profile image (in the form of string) to the Content Module and then the Content module sends that user details to the server through the Networking module. And then the server sends back these details to the content module through networking and the content module updates their hashmaps accordingly.

When clients try to send messages then the UI module transfers that message to the Content module and the Content module converts that message to JSON format and then it transfers that message to the server through the Networking module. And then the server sends back the messages to the Networking module of each client's machine and the Networking module then sends formatted messages to its Content module and then the Content module decodes it and sends decoded messages to its UI module to display the messages in the chat window.

Interfaces and My Work Agenda

1. Between Content Module and Networking Module

```
public interface ICommunicator {
    #Start communication
    void start();
    #Stop the Communication
    void stop();
    #Send the data to the destination IP
    void send(String destination, String message, String identifier);
    #Subscribe the module for receiving the data from the network module
    void subscribeForNotifications(String identifier, INotificationHandler
handler);
}
```

```
public interface INotificationHandler {
    void onMessageReceived(String message);
}
```

- For the communication between Content and Networking module, `subscribeForNotifications(String identifier, INotificationHandler handler)` method of `ICommunicator` interface will be called. Then the content module will provide an identifier and a handler to the Networking module and that will be stored in the hashmap. And for sending messages to the Content Module the Networking module will use this handler.
- And then I will implement a class (ContentManager) of `INotificationHandler` interface type which will include a method `onMessageReceived(String message)` that will be called by the Networking Module to send a message to the Content Module.

- The Networking Module will send the messages in JSON format to the Content Module by calling `onMessageReceived(String message)` method.
- Whenever a new user will join, the server will notify the Networking module and the Networking module will send a JSON format message whose first field has a string **newUser,** second field will have **username,** third field will have **IP address** and fourth field will have string of **image** (profile picture of new user) of a new user to the Content Module. And then the Content Module will update their hashmaps.
- Whenever a user wants to exit, the server will notify the Networking module and the Networking module will send a JSON format message whose first field has a string **userExit** and second field will have **username** of user who wants to exit to the Content Module. And then the Content Module will delete that user data from their hashmaps.
- Whenever the Networking module sends message which is sent by user to the Content Module in JSON format, then first field will contain **Messages in JSON format**, second field will contain **Time** at which message is actually being sent, third field will contain **username** of user who sent this message and fourth field will contain the **actual message** which previously sent by UI Module to Content Module.

2. Between UI Module and Content Module

```
public interface contentICommunicator{
    //Methods
    void notifyUserExit();
    void initialiseUser(String username, String ip, String image);
    void sendMessageToContent(String message);
    void subscribeForNotifications(String identifier, contentNotificationHandler
handler);
}
```

```
public interface contentNotificationHandler {
```

```
    //Methods
    void onMessageReceived(String username, String message, String image);
    void onNewUserJoined(String username);
    void onUserExit(String username);
}
```

- For the communication between UI and Content module, `subscribeForNotifications(String identifier, contentNotificationHandler handler)` method of `contentICommunicator` interface will be called. When this method will be called then 2 Hashmaps will be created on the Content Module.
- Fields of the first hashmap would be **username and ip address** which will contain username and ip addresses of the active users and of second hashmap would be **username and image** which will contain the username and profile pictures of active users.
- And also when `subscribeForNotifications (String identifier, contentNotificationHandler handler)` method will be called, then the UI nodule will provide an identifier and a handler to the content module and then we will store both in a hashmap.
- And whenever the Content module will have to send some messages to the UI Module, the Content Module will call the respective method with the help of `handler` by using `handler.method_name(arguments).`
- Whenever the Content Module will get to know about a new user joined, the Content Module will call `handler.onNewUserJoined(String username)` method with new user username to make the UI Module know about a new user is joined.

- Whenever the Content Module will get to know about a user wanting to exit, the Content module will call `handler.onUserExit(String username)` method with users username to make the UI Module know about this user has left.
- Whenever the Content Module will get user messages, the Content Module will call `handler.onMessageReceived(String username, String message, String image)` method (where argument **username** is the username of user who sent this message, **message** is the actual message with time and **image** is the profile picture in the form of string of the user who sent this message) to send messages to UI Module.
- For this Implementation I will use one queue to keep the messages coming from the Networking module and I will design two threads, one to push the message coming from the Networking module into the queue and another to pop (or send) the message to the UI module parallely.

Class Diagram