GroupChatManager Module Documentation

1. Introduction

The **GroupChatManager** module is responsible for handling group chat messages in the messaging application. Its main functionalities include:

- Sending messages to a group
- Retrieving group messages
- Editing messages
- Deleting messages
- Pinning/unpinning messages
- Retrieving pinned messages
- Logging message-related actions

This module interacts with the **Database** and **GroupManager** modules to ensure secure and consistent message management.

---

2. Dependencies

- **Database**: Executes SQL queries for message storage and retrieval.
- **GroupManager**: Validates group existence and user membership.
- **Message**: Data structure representing a chat message (ID, sender, content, timestamps, status).
- **Boost.UUID**: Generates unique identifiers for log entries.
- **ctime**: Handles timestamps for message creation and editing.

---

3. Public Interface

```
bool sendGroupMessage(const std::string& group_id, const Message& message)
```

- **Input:** Group ID, message object (id, sender_id, content, timestamps).
- **Process:**
    1. Verify that the group exists.
    2. Verify that the sender is a group member.
    3. Insert the message into the database.
    4. Log the `"send"` action.
- **Output:** `true` if successful, otherwise `false`.

```
std::vector<Message> getGroupMessages(const std::string& group_id, int limit = 100)
```

- **Input:** Group ID, maximum number of messages (default = 100).
- **Process:** Fetches the most recent non-deleted messages for the specified group.
- **Output:** A list of `Message` objects ordered by timestamp (descending).

`bool deleteGroupMessage(const std::string& message_id, const std::string& requester_id)`

- **Input:** Message ID, requester's user ID.
- **Process:**
  - Only the message sender or the group creator can delete a message.
  - Updates the database to mark the message as `deleted=1`.
  - Logs the `"delete"` action.
- **Output:** `true` if successful, otherwise `false`.

`bool editGroupMessage(const std::string& message_id, const std::string& new_content, const std::string& requester_id)`

- **Input:** Message ID, new message content, requester's user ID.
- **Process:**
  - Only the original sender can edit their message.
  - Updates the message content and `edited_timestamp` in the database.
  - Logs the `"edit"` action.
- **Output:** `true` if successful, otherwise `false`.

`Message getGroupMessageById(const std::string& message_id)`

- **Input:** Message ID.
- **Process:** Retrieves the message record from the database.
- **Output:** Returns a `Message` object, or an empty object if not found.

`bool pinGroupMessage(const std::string& message_id, const std::string& requester_id, bool pin = true)`

- **Input:** Message ID, requester's user ID, pin flag (`true` = pin, `false` = unpin).
- **Process:**
  - Only the group creator can pin or unpin messages.
  - Updates the `pinned` field in the database.
  - Logs `"pin"` or `"unpin"` action.
- **Output:** `true` if successful, otherwise `false`.

`std::vector<Message> getPinnedMessages(const std::string& group_id)`

- **Input:** Group ID.
- **Process:** Retrieves all pinned, non-deleted messages for the given group.
- **Output:** A list of pinned `Message` objects.

---

4. Private Methods

`void logAction(const std::string& message_id, const std::string& group_id, const std::string& user_id, const std::string& action)`

- **Process:** Records actions (`send`, `edit`, `delete`, `pin`, `unpin`) in the `group_message_logs` table with a unique ID and timestamp.

---

5. Rules and Constraints

- A user must be a group member to send messages.
- Only the sender can edit their own messages.
- Messages can only be deleted by the sender or the group creator.
- Only the group creator can pin/unpin messages.
- Deleted messages are not returned by `getGroupMessages` or `getPinnedMessages`.