Message Module Documentation

1. Overview

The **Message** module is responsible for handling messages in the messaging system. It manages message creation, serialization to JSON, deserialization from JSON, and provides unique identifiers (UUIDs) for each message instance.

---

2. Components

🔹 `enum class MessageType`

Defines the type of a message:

- `PRIVATE` → Private message between two users
- `GROUP` → Message inside a group
- `CHANNEL` → Message posted in a channel

Helper functions:

- `messageTypeToString(MessageType)` → Converts a message type into a string
- `stringToMessageType(std::string)` → Converts a string into a `MessageType` (throws an exception if invalid)

🔹 `struct Message`

Represents a single message with the following attributes:

- `id` → Unique identifier (UUID)
- `sender_id` → ID of the sender
- `receiver_id` → ID of the receiver
- `type` → Message type (`PRIVATE`, `GROUP`, or `CHANNEL`)
- `content` → The message body (string)
- `timestamp` → Time when the message was created
- `delivered` → Delivery status flag (default: `false`)
- `read` → Read status flag (default: `false`)

---

3. Methods

🟢 Constructor

```
Message(const std::string& sender,
        const std::string& receiver,
        MessageType msg_type,
        const std::string& msg_content);
```

- Creates a new message with all required fields.
- Automatically generates a UUID for the message ID.
- Initializes the timestamp with the current system time.

● static std::string generateUUID()

- Generates a unique identifier for each message using the Boost UUID library.

● json toJson() const

- Serializes a `Message` object into a JSON representation.
- Useful for storing messages in a database or transferring them over the network.

Example output:

```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "sender_id": "user1",
  "receiver_id": "user2",
  "type": "PRIVATE",
  "content": "Hello!",
  "timestamp": 1694529000,
  "delivered": false,
  "read": false
}
```

● static Message fromJson(const json& j)

- Deserializes a JSON object back into a `Message` instance.
- If optional fields (`delivered` or `read`) are missing, they default to `false`.

---

4. Dependencies

- **Boost UUID** → For generating unique message IDs
- **nlohmann/json** → For JSON serialization and deserialization
- **chrono / ctime** → For timestamp management

---

5. Design Notes

- Each message has a unique ID, ensuring no duplicates.
- JSON serialization makes it possible to persist messages in a database or transmit them between client and server.
- Input validation in `stringToMessageType` prevents invalid message type strings from being used.