Chat Service API Documentation

Developers: Omar Adel, Omar Hesham, Omar Tamer

Base URL: http://localhost:8083/

May 12, 2025

Message Attributes

- `UUID senderId`: User ID of the sender
- `UUID receiverId`: User ID of the receiver
- `String content`: Content of the message. **Cannot be an empty string and must be between 1 and 500 characters.**
- `MessageType type`: Type of the message
- `UUID id`: Unique ID of the message (auto-generated.)
- `LocalDateTime timestamp`: Timestamp of when the message was sent. (Does not change when the message is updated)
- `MessageStatus status`: Status of the message
- `ReportType reportType`: Type of report (if any) associated with the message. **null if `reported` is false**
- `boolean reported`: Indicates if the message has been reported

Enums

```
enum MessageType {
    TEXT,
    IMAGE,
    PRODUCT;
}
```

```
enum MessageStatus {
    SENT,
    DELIVERED,
    SEEN,
    FAILED;
}
```

Available Endpoints

1 GET /

Default route to verify that the server is running. Returns a 200 status code with a message indicating that the server is running.

2 GET /messages/seed/number

Route to the database seeder. Returns a 200 status code with a message indicating that the database has been seeded with the specified number of messages. Adds generated messages to the database.

3 GET /messages

Returns a list of all messages.

Sample Request

```
GET http://localhost:8083/messages
```

Sample Response

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "senderId": "1fbdc4e1-e47e-4d9a-afc6-bc3f4f975d57",
        "receiverId": "ad0aa92b-cdae-4a4b-901e-453dab81df87",
        "content": "This is my favorite dish!",
        "type": "IMAGE",
        "id": "1e38e96c-8a9e-466d-95a9-6b61ef33743c",
        "timestamp": "2025-05-08T00:39:59.383",
        "status": "FAILED",
        "reportType": null,
        "reported": false
    },
    {
        "senderId": "d9084db9-a392-4245-9eba-09e444141b4b",
        "receiverId": "b406709f-8513-445c-9155-96e0a3573eeb",
        "content": "Check out this image!",
        "type": "TEXT",
        "id": "613c5c3e-fffb-4222-9a70-17146b2d5551",
        "timestamp": "2025-05-08T00:39:59.271",
        "status": "DELIVERED",
        "reportType": null,
        "reported": false
    }
]
```

4 GET /messages/id

Returns the message with the given id.

4.1 Sample Valid Request

Sample Request

```
GET http://localhost:8083/messages/987e6575-576c-4f1b-87e0-955931a98b01
```

Sample Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "senderId": "94d013a8-cb13-473b-98eb-90c9445f8669",
    "receiverId": "1bb62666-ab0f-4759-a5df-b73968795a15",
    "content": "Hello, how are you?",
    "type": "PRODUCT",
    "id": "987e6575-576c-4f1b-87e0-955931a98b01",
    "timestamp": "2025-05-08T00:39:59.354",
    "status": "DELIVERED",
    "reportType": "OTHER",
    "reported": true
}
```

4.2 Sample Invalid Request: Message not found

Sample Request

```
GET http://localhost:8083/messages/987e6575-576c-4f1b-87e0-955931a98b01
```

Sample Response

```
HTTP/1.1 404 Not Found
Content-Type: application/json
```

5 POST /messages

Creates a new message.
Request Body:
Must be provided with a JSON body containing all of the following required fields:

- `senderId` (UUID)
- `receiverId` (UUID)
- `content` (String)
- `type` (MessageType)

Any other fields will be ignored. If one of the required fields is missing, the request will fail with a 400 error.
The message will be created with the status `SENT` and the timestamp will be set to the current time.
Note that the `senderId` and `receiverId` are allowed to be the same UUID. That is to say under the current implementation, a user can send a message to hismselef/herself.

Returns a `Location` response header with a URI to the created message, as well as the created message in the response body and a 201 status code.

5.1 Sample Valid Request

```
POST http://localhost:8083/messages
Content-Type: application/json

{
    "senderId": "e606ee69-6957-4fc6-a88f-ebadc45c5094",
    "receiverId": "613fcc40-16db-41d4-9a2a-00f517b088e3",
    "content": "This dish looks delicious!",
    "type": "TEXT"
}
```

Sample Response

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /messages/1e38e96c-8a9e-466d-95a9-6b61ef33743c

{
    "senderId": "e606ee69-6957-4fc6-a88f-ebadc45c5094",
    "receiverId": "613fcc40-16db-41d4-9a2a-00f517b088e3",
    "content": "This dish looks delicious!",
    "type": "TEXT",
    "id": "1e38e96c-8a9e-466d-95a9-6b61ef33743c",
    "timestamp": "2025-05-08T00:39:59.383",
    "status": "SENT",
    "reportType": null,
    "reported": false
}
```

5.2 Sample Bad Requests

5.2.1 Empty content

```
POST http://localhost:8083/messages
Content-Type: application/json

{
    "senderId": "e606ee69-6957-4fc6-a88f-ebadc45c5094",
    "receiverId": "613fcc40-16db-41d4-9a2a-00f517b088e3",
    "content": "",
    "type": "TEXT"
}
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain;charset=UTF-8

Validation failed for the following fields:
content: must not be blank
```

5.2.2 Missing required fields

```
POST http://localhost:8083/messages
Content-Type: application/json

{
    "senderId": "e606ee69-6957-4fc6-a88f-ebadc45c5094",
    "receiverId": "613fcc40-16db-41d4-9a2a-00f517b088e3",
    "content": "Thanks, the dish was great!"
}
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain;charset=UTF-8

Validation failed for the following fields:
type: must not be null
```

5.2.3 Required fields with null values

```
POST http://localhost:8083/messages
Content-Type: application/json

{
    "senderId": null,
    "receiverId": null,
    "content": "Test",
    "type": "IMAGE"
}
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain;charset=UTF-8

Validation failed for the following fields:
senderId: must not be null
receiverId: must not be null
```

5.2.4 Invalid type

```
POST http://localhost:8083/messages
Content-Type: application/json

{
    "senderId": "e606ee69-6957-4fc6-a88f-ebadc45c5094",
    "receiverId": "613fcc40-16db-41d4-9a2a-00f517b088e3",
    "content": "This dish looks delicious!",
    "type": "INVALID_TYPE"
}
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain;charset=UTF-8

Invalid value for enum type: INVALID_TYPE. Accepted values are: [TEXT,
   IMAGE, PRODUCT]
```

6 DELETE /messages/id

Deletes the message with the given ID.

6.1 Sample Valid Request

Sample Request

```
DELETE http://localhost:8083/messages/80e0bf1e-bff1-4571-b7e9-
   b2e24dbdcde8
```

Sample Response

```
HTTP/1.1 204 No Content
```

6.2 Sample Invalid Request: Message not found

Sample Request

```
DELETE http://localhost:8083/messages/ced4f1eb-52e5-4e50-9ae4-
   f95daf011404
```

Sample Response

```
HTTP/1.1 404 Not Found
Content-Type: text/plain;charset=UTF-8

No message with id ced4f1eb-52e5-4e50-9ae4-f95daf011404 found
```

7 PATCH /messages/id

Updates an existing message.

Request Body:
Must be provided with a JSON body containing any fields to be updated. Only fields for which values provided are not null will be updated.

Under the current implementation, updating a message does not change its `timestamp`. The timestamp does not change and represenets when the message was first created (sent).

The following fields can be updated:

- `content`
- `type`
- `status`

If the message was updated successfully, the response will contain the updated message with a 200 status code.

If the message was not updated, either because no valid fields to update were provided, or because the provided values are the same as the current values, a 400 status code will be returned along with an empty response body.

7.1 Sample Valid Request

```
PATCH http://localhost:8083/messages/80e0bf1e-bff1-4571-b7e9-
    b2e24dbdcde8
{
    "content": "Updated Content",
    "type": "PRODUCT"
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "senderId": "559588f3-5965-48fe-bfc7-23304bea87b2",
    "receiverId": "e7722912-538a-4663-8cc3-a11e9b63899c",
    "content": "Updated Content",
    "type": "PRODUCT",
    "id": "80e0bf1e-bff1-4571-b7e9-b2e24dbdcde8",
    "timestamp": "2025-05-08T01:51:47.521",
    "status": "SENT",
    "reportType": null,
    "reported": false
}
```

7.2 Sample Requests that do not update the message

7.2.1 No fields provided

```
PATCH http://localhost:8083/messages/80e0bf1e-bff1-4571-b7e9-
    b2e24dbdcde8
{
    "id": "80e0bf1e-bff1-4571-b7e9-b2e24dbdcde8",
    "senderId": "1fcb411f-c8fa-4999-a434-524d964a7572",
    "receiverId": "4d8e10a3-8b1a-4c66-ab14-702ef9fbd442"
}
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain;charset=UTF-8

No valid fields to update
```

7.2.2 No changes to the message

```
PATCH http://localhost:8083/messages/80e0bf1e-bff1-4571-b7e9-
    b2e24dbdcde8
{
    "content": "Updated Content",
    "type": "PRODUCT"
}
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain;charset=UTF-8

No valid fields to update
```

8 GET /messages/id/seen

Returns true if the message with the given ID has been seen, false otherwise.

Sample Requests

```
GET http://localhost:8083/messages/80e0bf1e-bff1-4571-b7e9-b2e24dbdcde8
    /seen
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

true
```

```
GET http://localhost:8083/messages/93211957-555e-4be5-8d2a-875a481f9c66
    /seen
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

false
```

9 PATCH /messages/id/seen

Marks the message with the given ID as seen.
Similar to update message endpoint in section 7. Returns a 200 status code with the updated message if the message was marked as seen successfully.
If the message was not marked as seen because it was already marked as seen, a 400 status code will be returned along with an empty response body.

9.1 Sample Valid Request

```
PATCH http://localhost:8083/messages/869a93e9-9a22-48e5-abf8-5983
    f313611f/seen
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "senderId": "f47e7de8-8b5a-4596-b21f-4270f67978f1",
    "receiverId": "00c5ac09-f93e-4cd7-8e59-b8b003e0ad05",
    "content": "Test content",
    "type": "IMAGE",
    "id": "869a93e9-9a22-48e5-abf8-5983f313611f",
    "timestamp": "2025-05-11T18:34:33.963",
    "status": "SEEN",
    "reportType": null,
    "reported": false
}
```

9.2 Sample Invalid Request: Message status already set to seen

```
PATCH http://localhost:8083/messages/869a93e9-9a22-48e5-abf8-5983
    f313611f/seen
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain;charset=UTF-8

Message with id 869a93e9-9a22-48e5-abf8-5983f313611f is already marked
    as seen
```

10 DELETE /messages

Deletes all messages from the database.

10.1 Sample Valid Request

```
DELETE http://localhost:8083/messages/clear
```

Response

```
HTTP/1.1 204 No Content
```

10.2 Sample Invalid Request

```
DELETE http://localhost:8083/messages/clear
```

Response

```
HTTP/1.1 404 Not Found
Content-Type: text/plain;charset=UTF-8

No messages to delete
```