Detailed Messaging API Documentation

Table of Contents

Authentication

All API endpoints require authentication using a bearer token. Include the token in the Authorization header of your requests:

```
Authorization: Bearer <your_auth_token>
```

Login

Authenticate and receive a bearer token.

Endpoint: `POST /api/auth/login`

Request Body:

```
{
  "email": "user@example.com",
  "password": "yourpassword"
}
```

Response:

```
{
  "token": "your_auth_token",
  "user": {
    "body": {
      "data": {
        "id": 123
      }
    }
  }
}
```

Status Codes:

- 200: Successful login
- 401: Invalid credentials

Conversations

Create a New Conversation

Create a new one-to-one or group conversation.

Endpoint: `POST /api/conversations`

Request Body:

```
{
  "type": "ONE_TO_ONE" | "GROUP",
  "participantIds": [1, 2, 3]
}
```

Response:

```
{
  "id": 456,
  "type": "ONE_TO_ONE" | "GROUP",
  "participants": [
    { "id": 1 },
    { "id": 2 },
    { "id": 3 }
  ]
}
```

Status Codes:

- 201: Conversation created successfully
- 400: Invalid request (e.g., non-existent participants)

Notes:

- For ONE_TO_ONE conversations, include only one other participant ID.
- For GROUP conversations, you can include multiple participant IDs.

Get All Conversations

Retrieve all conversations for the authenticated user.

Endpoint: `GET /api/conversations`

Response:

```
[
  {
    "id": 456,
    "participants": [
      { "id": 1 },
      { "id": 2 }
    ],
    "last_message_at": "2023-09-22T12:00:00Z"
  }
]
```

Status Codes:

- 200: Successful retrieval

Leave a Group Conversation

Leave a group conversation.

Endpoint: `POST /api/conversations/:conversationId/leave`

Response:

```
{
  "message": "Left the conversation successfully"
}
```

Status Codes:

- 200: Successfully left the conversation
- 403: Not a member of the conversation
- 404: Conversation not found

Notes:

- This endpoint is only applicable for group conversations.
- Users cannot leave one-to-one conversations.

Messages

Send a Message

Send a new message in a conversation.

Endpoint: `POST /api/conversations/:conversationId/messages`

Request Body:

```
{
  "content": "Hello, this is a test message!",
  "parent_message_id": 789 // Optional, for threaded replies
}
```

Response:

```
{
  "id": 101,
  "content": "Hello, this is a test message!",
  "sender_id": 1,
  "parent_message_id": 789, // If applicable
  "created_at": "2023-09-22T12:00:00Z"
}
```

Status Codes:

- 201: Message sent successfully
- 403: Not a member of the conversation
- 404: Conversation not found

Get Messages in a Conversation

Retrieve messages from a conversation with optional pagination.

Endpoint: `GET /api/conversations/:conversationId/messages`

Query Parameters:

- `limit` : Number of messages to retrieve (default: 50)
- `before` : ISO timestamp to retrieve messages before a certain date

Response:

```
[
  {
    "id": 101,
    "content": "Hello, this is a test message!",
    "sender_id": 1,
    "created_at": "2023-09-22T12:00:00Z"
  }
]
```

Status Codes:

- 200: Successful retrieval
- 403: Not a member of the conversation
- 404: Conversation not found

Edit a Message

Edit an existing message.

Endpoint: `PUT /api/conversations/:conversationId/messages/:messageId`

Request Body:

```
{
  "content": "Updated message content"
}
```

Response:

```
{
  "id": 101,
  "content": "Updated message content",
  "updated_at": "2023-09-22T12:05:00Z"
}
```

Status Codes:

- 200: Message updated successfully
- 400: Invalid content (e.g., empty message)
- 403: Not authorized to edit the message
- 404: Message not found

Notes:

- Users can only edit their own messages.
- There may be a time limit for editing messages (e.g., 5 minutes after sending).

Delete a Message

Delete an existing message.

Endpoint: `DELETE /api/conversations/:conversationId/messages/:messageId`

Response:

```
{
  "message": "Message deleted successfully"
}
```

Status Codes:

- 200: Message deleted successfully
- 403: Not authorized to delete the message
- 404: Message not found

Notes:

- Users can only delete their own messages.

Search Messages

Search for messages within a conversation.

Endpoint: `GET /api/conversations/:conversationId/messages/search`

Query Parameters:

- `query`: Search term

Response:

```
[
  {
    "id": 101,
    "content": "Message containing search term",
    "sender_id": 1
  }
]
```

Status Codes:

- 200: Successful search (returns an empty array if no results)
- 403: Not a member of the conversation
- 404: Conversation not found

Get Thread Messages

Retrieve messages in a thread.

Endpoint: `GET /api/conversations/:conversationId/messages/:parentMessageId/thread`

Response:

```
[
  {
    "id": 102,
    "content": "This is a reply",
    "parent_message_id": 101,
    "sender_id": 2,
    "created_at": "2023-09-22T12:05:00Z"
  }
]
```

Status Codes:

- 200: Successful retrieval
- 403: Not a member of the conversation
- 404: Parent message or conversation not found

Reactions

Add/Update Reaction

Add or update a reaction to a message.

Endpoint: `POST /api/conversations/:conversationId/messages/:messageId/reactions`

Request Body:

```
{
  "reaction": "□"
}
```

Response:

```
{
  "reactions": "[{\"userId\":1,\"reaction\":\"□\"}]"
}
```

Status Codes:

- 200: Reaction added/updated successfully
- 403: Not a member of the conversation
- 404: Message or conversation not found

Notes:

- If a user has already reacted to the message, this will update their existing reaction.

Remove Reaction

Remove a reaction from a message.

Endpoint: `DELETE /api/conversations/:conversationId/messages/:messageId/reactions`

Request Body:

```
{
  "reaction": "□"
}
```

Response:

```
{
  "message": "Reaction removed successfully"
}
```

Status Codes:

- 200: Reaction removed successfully
- 403: Not a member of the conversation
- 404: Reaction, message, or conversation not found

File Uploads

Files can be uploaded as part of a message. Use multipart/form-data to send both the message content and the file.

Endpoint: `POST /api/conversations/:conversationId/messages`

Request:

- Use `multipart/form-data`
- Include `content` field for message text
- Include `file` field for file upload

Response:

```
{
  "id": 103,
  "content": "Here is a test file attachment",
  "sender_id": 1,
  "created_at": "2023-09-22T12:10:00Z",
  "file_info": [
    {
      "id": 1,
      "file_url": "https://example.com/files/1",
      "download_url": "https://example.com/files/1/download",
      "original_name": "example.png"
    }
  ]
}
```

Status Codes:

- 201: File uploaded successfully
- 400: Invalid file type or size
- 403: Not a member of the conversation
- 404: Conversation not found

Notes:

- There are restrictions on file types and sizes. Consult your system administrator for specific limits.
- Common image formats (e.g., .png, .jpg) are typically allowed.
- Empty files or files exceeding the size limit will be rejected.

Read Status

Mark Messages as Read

Mark all messages in a conversation as read.

Endpoint: `POST /api/conversations/:conversationId/read`

Response:

```
{
  "message": "Messages marked as read"
}
```

Status Codes:

- 200: Messages marked as read successfully
- 403: Not a member of the conversation
- 404: Conversation not found

Get Unread Message Count

Get the total count of unread messages across all conversations.

Endpoint: `GET /api/conversations/unread-count`

Response:

```
{
  "unreadCount": 5
}
```

Status Codes:

- 200: Unread count retrieved successfully

Notes:

- This endpoint returns the total unread count across all conversations for the authenticated user.
- If a user has no conversations or all messages are read, the count will be 0.

Error Handling

The API uses conventional HTTP response codes to indicate the success or failure of requests. In general:

- 2xx range indicate success
- 4xx range indicate an error that failed given the information provided (e.g., bad request, unauthorized)
- 5xx range indicate an error with the server

All error responses will include a JSON object with an `error` field providing a description of the error:

```
{
  "error": "Detailed error message"
}
```

Pagination

For endpoints that return lists of items (e.g., messages, conversations), the API uses cursor-based pagination. Use the `before` parameter with the `created_at` timestamp of the oldest item in the current set to get the next page of results.

Testing

It's recommended to use the provided test suite to ensure your implementation correctly interacts with the API. The test suite covers a wide range of scenarios including edge cases and error conditions.