# WebSocket API Documentation - Document Chat

## Overview

The WebSocket API allows real-time interaction for document-based chat sessions. This API enables users to connect via WebSockets, send chat messages, retrieve document-based contextual responses, and receive AI-generated responses in a streaming manner.

## WebSocket Endpoint

**URL:**

/qna/api/chat/docChat/

**Protocol:** WebSocket (wss or ws)

## Authentication

The WebSocket connection requires authentication. The user must be logged in, and their session must be validated upon connection.

## Query Parameters

| **Parameter** | **Type** | **Required** | **Description** |
| --- | --- | --- | --- |
| sessionUUID | string | Yes | UUID of the chat session the user is connecting to. |

## Connection Process

1. The client establishes a WebSocket connection to /qna/api/chat/docChat/.
2. The server validates the sessionUUID to ensure the user has an active session.
3. Upon successful validation, the server retrieves indexed document UUIDs for that session.
4. The connection is accepted, and the server sends a confirmation message.

## WebSocket Messages

### Incoming Messages

| **Type** | **Expected Format** | **Description** |
| --- | --- | --- |
| Question | { "question": "Your question here" } | Sends a user query to the WebSocket server. |

### Outgoing Messages

| **Type** | **Format** | **Description** |
| --- | --- | --- |
| Connection Success | { "message": "Connected to WebSocket!" } | Sent upon successful connection. |
| Error | { "error": "Error message here" } | Sent when an issue occurs (e.g., invalid session, JSON parsing error). |
| Status | { "type": "status", "response": "Start" } or { "type": "status", "response": "end" } | Indicates when the AI response streaming starts and ends. |
| Answer | { "type": "answer", "response": "AI-generated response chunk" } | Sent in chunks as the AI generates the response. |

## Functional Logic

**Connection Handling:**

* 1. Extracts the sessionUUID from the query string.
  2. Validates the session against the authenticated user.
  3. Fetches indexed documents associated with the session.
  4. Establishes a connection and initializes the AI model.

**Receiving User Questions:**

* 1. Parses the JSON payload.
  2. Retrieves relevant document context from Weaviate vector search.
  3. Sends a status message (Start).
  4. Streams AI-generated responses chunk by chunk.
  5. Stores the conversation in the database.
  6. Sends a status message (End).

**Vector Search for Relevant Documents:**

* 1. Connects to the Weaviate database.
  2. Uses filters to match documentUUIDs associated with the session.
  3. Retrieves top-ranked document chunks related to the user’s query.

**Streaming AI Responses:**

* 1. Constructs a context including document content and user questions.
  2. Calls the AI model asynchronously to generate responses.
  3. Sends the AI response in real-time chunks to the client.

**Disconnection Handling:**

* 1. Logs the disconnection with a status code.

## Error Handling

| **Error Code** | **Description** |
| --- | --- |
| 4001 | Missing or invalid sessionUUID. |
| 500 | Internal server error (e.g., database issues, vector search failure). |

## Example WebSocket Interaction

### Client Request

{

"question": "What is the summary of the document?"

}

### Server Responses

#### Connection Confirmation

{

"message": "Connected to WebSocket!"

}

#### Streaming Status Start

{

"type": "status", "response": "Start"

}

#### AI Streaming Responses

{

"type": "answer", "response": "The document discusses..."

}

#### Streaming Status End

{

"type": "status", "response": "end"

}

## Security Considerations

* **Authentication Required:** Users must be logged in and provide a valid sessionUUID.
* **Data Validation:** The API checks input formats and prevents invalid requests.
* **Rate Limiting:** Future enhancements can include limiting concurrent WebSocket connections per user.

## Summary

This WebSocket API facilitates AI-powered document-based chat by integrating real-time communication, vector database lookups, and streaming responses. Clients can interactively ask questions related to documents, and the AI will generate contextually relevant answers in real-time.