# API Documentation: Document Ingestion

## Base URL

http://your-domain.com/documents/api/ingestion/

## Authentication

This API requires authentication via **Token Authentication**. Include the token in the Authorization header:

Authorization: Token <your-token>

## 1. Retrieve Paginated List of Documents

### ****GET**** /documents/api/ingestion/

Retrieves a paginated list of documents for the authenticated user.

### Query Parameters

| **Parameter** | **Type** | **Required** | **Description** |
| --- | --- | --- | --- |
| perPage | int | No | Number of results per page (default: 10) |
| page | int | No | Page number to retrieve |

### Response

#### ****Success (200)****

{

"count": 100,

"total\_pages": 10,

"current\_page": 1,

"next\_page": 2,

"previous\_page": null,

"results": [

{

"documentUUID": "123e4567-e89b-12d3-a456-426614174000",

"documentTitle": "Sample Document",

"processStatus\_\_status": "Success",

"createdAt": "2025-02-17T12:34:56Z"

}

]

}

#### ****Error Responses****

* **404** - No more pages available
* **500** - Internal server error

## 2. Document Ingestion

### ****POST**** /documents/api/ingestion/

Ingests a new document by saving metadata and storing content in Weaviate.

### Request Body (JSON)

{

"name": "Sample Document",

"text": "This is the content of the document."

}

### Response

#### ****Success (201)****

{

"message": {

"IngestionStatus": "Successfully stored 5 chunks in Weaviate!",

"DocumentUUID": "123e4567-e89b-12d3-a456-426614174000"

}

}

#### ****Error Responses****

* **500** - Internal server error or "Scheduled status not found."

## OpenAPI Specification (Swagger)

openapi: 3.0.0

info:

title: Document Ingestion API

version: 1.0.0

description: API for document ingestion and retrieval.

servers:

- url: http://your-domain.com

description: Production server

paths:

/documents/api/ingestion:

get:

summary: Retrieve paginated list of documents

security:

- TokenAuth: []

parameters:

- name: perPage

in: query

schema:

type: integer

description: Number of results per page (default: 10)

- name: page

in: query

schema:

type: integer

description: Page number

responses:

'200':

description: Successful response

content:

application/json:

schema:

type: object

properties:

count:

type: integer

total\_pages:

type: integer

current\_page:

type: integer

next\_page:

type: integer

nullable: true

previous\_page:

type: integer

nullable: true

results:

type: array

items:

type: object

properties:

documentUUID:

type: string

format: uuid

documentTitle:

type: string

processStatus\_\_status:

type: string

createdAt:

type: string

format: date-time

'404':

description: No more pages

'500':

description: Internal server error

post:

summary: Ingest a new document

security:

- TokenAuth: []

requestBody:

required: true

content:

application/json:

schema:

type: object

properties:

name:

type: string

text:

type: string

responses:

'201':

description: Document successfully ingested

content:

application/json:

schema:

type: object

properties:

message:

type: object

properties:

IngestionStatus:

type: string

DocumentUUID:

type: string

format: uuid

'500':

description: Internal server error

components:

securitySchemes:

TokenAuth:

type: apiKey

in: header

name: Authorization

**Logical Steps Behind the API**

### 1. Authentication

* The API uses **Token Authentication** (Token <your-token>).
* Each request must include the token in the **Authorization** header.
* The server validates the token before processing any request.

### 2. Retrieving Paginated Documents (GET /documents/api/ingestion/)

#### Steps:

1. **User sends a GET request** to /documents/api/ingestion/ with optional query parameters:
   * perPage: Number of documents per page (default: 10).
   * page: The page number requested.
2. **Server validates the request:**
   * Checks if the provided token is valid.
   * Identifies the authenticated user.
3. **Fetch paginated documents:**
   * The database retrieves documents related to the authenticated user.
   * Applies pagination logic (LIMIT and OFFSET) based on perPage and page.
4. **Format and send response:**
   * Constructs a JSON response containing:
     + count: Total number of documents.
     + total\_pages: Number of pages.
     + current\_page: Current page number.
     + next\_page / previous\_page: Navigation links.
     + results: A list of documents with UUID, title, status, and createdAt.
   * Returns **HTTP 200 (OK)** if successful.
   * Returns **HTTP 404 (Not Found)** if no more pages exist.
   * Returns **HTTP 500 (Internal Server Error)** for server issues.

### 3. Document Ingestion (POST /documents/api/ingestion/)

#### Steps:

1. **User sends a POST request** with a JSON payload:

{

"name": "Sample Document",

"text": "This is the content of the document."

}

1. **Server validates the request:**
   * Checks the **token authentication**.
   * Ensures that name and text fields are present in the request body.
2. **Process the document:**
   * Creates a **unique UUID** for the document.
   * Breaks the document **text** into smaller chunks (e.g., sentences or paragraphs).
   * Stores the **document metadata** (title, UUID, status) in the database.
   * Sends the **chunked content** to **Weaviate** (a vector database) for storage.
3. **Return response:**

If successful, returns **HTTP 201 (Created)** with:

{

"message": {

"IngestionStatus": "Successfully stored X chunks in Weaviate!",

"DocumentUUID": "123e4567-e89b-12d3-a456-426614174000"

}

}

If an error occurs (e.g., database failure or missing fields), returns **HTTP 500 (Internal Server Error)**.

### 4. Security Implementation

* **Authentication:** Users must send a valid token for access.
* **Authorization:** Only authenticated users can access or ingest documents.
* **Data Validation:**
  + Ensures required fields are present.
  + Enforces proper data types (e.g., int for page).
* **Pagination:** Limits data retrieval for better performance.
* **Weaviate Integration:** Stores vectorized document data for fast search and retrieval.

### ****Summary of API Workflow****

1. **User sends a request** (GET or POST).
2. **API authenticates the user** using the token.
3. **Performs the requested operation:**
   * Fetch paginated documents (GET).
   * Ingest document content (POST).
4. **Returns structured JSON response** with relevant data or error messages.