Swagger
Supported by SMARTBEAR

/static/swagger.json Explore

DocServer API

[Base URL: localhost:8080/]
/static/swagger.json

DocServer API

Purpose: This is a simple API server designed for **educational purposes only**. It demonstrates basic concepts of user authentication, document storage (as JSON), document sharing, and content-based querying. **It is NOT intended for production use.**

High-Level Overview:
DocServer allows users to:

- Register and log in to manage their accounts.
- Create, retrieve, update, and delete documents. Document content can be any valid JSON structure.
- Share their documents with other registered users.
- Search for documents they have access to, including powerful filtering based on the document's JSON content.

Content Querying (`content_query` parameter):
The `GET /documents` endpoint supports filtering documents based on their content using the `content_query` parameter. This allows you to search for documents where specific fields within the JSON content match certain criteria.

Query Syntax:
Each `content_query` parameter string follows the format: `path operator value`

- `path` : A dot-separated path to navigate the JSON structure (e.g., `user.name` , `details.metadata.version`). Use numeric indices for arrays (e.g., `items.0.id` , `tags.1`).
- `operator` : The comparison operator. Supported operators include:
- `equals` : Equal to (strings, numbers, booleans, null)
- `notequals` : Not equal to
- `greaterthan` : Greater than (numbers)
- `greaterthanorequals` : Greater than or equal to (numbers)
- `lessthan` : Less than (numbers)
- `lessthanorequals` : Less than or equal to (numbers)
- `contains` : String contains substring, or array contains element (case-sensitive by default).
- `startswith` : String starts with prefix (case-sensitive by default).
- `endswith` : String ends with suffix (case-sensitive by default).
- `value` : The value to compare against.
- Strings MUST be enclosed in double quotes (e.g., `\"John Doe\"`). Remember to URL-encode the query parameter string. Add `-insensitive` suffix to string operators (e.g., `equals-insensitive` , `contains-insensitive`) for case-insensitive matching.
- Numbers (e.g., `123` , `45.6`), booleans (`true` / `false`), and `null` should be used directly.

Logical Operators (Combining Queries):
You combine multiple conditions by providing `content_query` parameters for conditions interleaved with explicit logical operators (`and` or `or`).

- `and` **(Explicit):** To link two conditions with AND, place `content_query=and` between them. The document must match *both* conditions.
- `or` **(Explicit):** To link two conditions with OR, place `content_query=or` between them. The document must match *either* condition.

Examples:

Assume document content like:

```
{
"project": "Alpha",
"status": "active",
"priority": 5,
"assignee": { "name": "Alice", "email": "alice@example.com" },
"tags": ["urgent", "backend"],
"metadata": { "version": 1.2, "reviewed": true }
}
```

1. **Simple Equality:** Find documents where `status` is `active` .
   `?content_query=status equals \"active\"`
2. **Numeric Comparison:** Find documents where `priority` is greater than or equal to `5` .
   `?content_query=priority greaterthanorequals 5`
3. **Nested Field:** Find documents assigned to `Alice` .
   `?content_query=assignee.name equals \"Alice\"`

4. **Array Element:** Find documents where the first tag is `urgent`.
   `?content_query=tags.0 equals \"urgent\"`
5. **Explicit `AND`:** Find documents for project `Alpha` **AND** status `active`.
   `?content_query=project equals \"Alpha\"&content_query=and&content_query=status equals \"active\"`
6. **Explicit `OR`:** Find documents where status is `active` **OR** priority is less than `3`.
   `?content_query=status equals \"active\"&content_query=or&content_query=priority lessthan 3`
7. **Combined `AND` and `OR`:** Find documents where (project is `Alpha` **AND** status is `active`) **OR** (priority is `10`). Evaluation is strictly left-to-right.
   `?content_query=project equals \"Alpha\"&content_query=and&content_query=status equals \"active\"&content_query=or&content_query=priority equals 10`
   (Explanation: `project equals "Alpha"` AND `status equals "active"` is evaluated first, then the result is OR'd with `priority equals 10`.)
8. **Nested Field with `AND`:** Find documents where `assignee.name` is `Alice` **AND** `metadata.reviewed` is `true`.
   `?content_query=assignee.name equals \"Alice\"&content_query=and&content_query=metadata.reviewed equals true`
   Type "Bearer" followed by a space and JWT token.

MIT

Authentication

POST `/auth/forgot-password` Request Password Reset Code (OTP)

Initiates the password reset process by requesting a One-Time Password (OTP) to be sent (conceptually) to the user's registered email address.

Provide the `email` address associated with the account you want to reset the password for.
Security Note: To prevent attackers from figuring out which emails are registered ("email enumeration"), this endpoint will *always* return a `202 Accepted` response, regardless of whether the email exists in the system or not.
If the email *does* exist, the server generates an OTP, stores it temporarily, and (in a real system) would send it via email. The OTP is needed for the `/auth/reset-password` step.

Parameters Try it out

Name	Description
forgotPassword * required `object` *(body)*	The email address for the account needing a password reset. **Example Value** | Model

```
{
  "email": "string"
}
```

Parameter content type
application/json

Responses Response content type application/json

Code	Description
202	Request Accepted. If the email address is registered, an OTP has been generated (and would typically be emailed). Check your email for the code.
400	Bad Request: The request body is invalid (e.g., missing email or invalid format). **Example Value** | Model

Code **Description**

```
{
  "error": "string"
}
```

500

Internal Server Error: Something went wrong on the server while processing the request (e.g., OTP generation failed).

Example Value | Model

```
{
  "error": "string"
}
```

POST `/auth/login` Log In to Your Account

Authenticates a user using their registered email and password.

If the credentials are correct, the server generates a JSON Web Token (JWT). This token acts like a temporary key or session ID.
You need to include this JWT in the `Authorization` header (as a Bearer token) for subsequent requests to protected endpoints (like accessing your profile or documents).
Example Header: `Authorization: Bearer <your_token_here>`

Parameters

Try it out

Name	Description
login * required `object` *(body)*	Your email and password.

Example Value | Model

```
{
  "email": "string",
  "password": "string"
}
```

Parameter content type

application/json

Responses

Response content type application/json

Code **Description**

200

Login Successful. The response body contains the JWT access token.

Example Value | Model

```
{
  "token": "string"
}
```

400

Bad Request: The data you sent is invalid (e.g., missing email or password, incorrect JSON format).

Example Value | Model

```
{
  "error": "string"
}
```

Code	Description
401	Unauthorized: The email or password you provided is incorrect. Please check your credentials.

Example Value | Model

```
{
  "error": "string"
}
```

500	Internal Server Error: Something went wrong on the server during login (e.g., database issue, error generating the JWT).

Example Value | Model

```
{
  "error": "string"
}
```

POST `/auth/logout` Log Out (Client-Side Action)

Indicates the intention to log out. Since JWTs are stateless (the server doesn't keep track of active tokens), true logout happens on the client-side.

Action Required by Client: To effectively log out, the client application (e.g., your web browser or mobile app) MUST delete or discard the stored JWT access token.
Calling this endpoint doesn't invalidate the token on the server, but it serves as a conventional way to signal the end of a session in API design.

Parameters Try it out

No parameters

Responses Response content type application/json

Code	Description
204	Logout Signaled. No content is returned. Remember to discard the JWT on the client.
401	Unauthorized: Although logout is client-side, this endpoint might still require a valid token to be called as per API design consistency.

Example Value | Model

```
{
  "error": "string"
}
```

POST `/auth/reset-password` Set New Password Using Reset Code (OTP)

Completes the password reset process by setting a new password for the account.

You must provide:

- The `email` address of the account.
- The `otp` (One-Time Password) received after calling `/auth/forgot-password`.
- The desired `new_password` (must meet minimum length requirements, e.g., 8 characters).

The server will first verify if the provided OTP is correct and hasn't expired for the given email. If valid, it will hash the `new_password` and update the user's account.

Parameters

Try it out

Name	Description
resetPassword * required `object` *(body)*	Email, OTP, and the new password.

Example Value | Model

```
{
  "email": "string",
  "new_password": "stringst",
  "otp": "string"
}
```

Parameter content type

application/json

Responses

Response content type application/json

Code	Description
204	Password Reset Successful. Your new password is now active. You can log in using it. No content is returned in the response body.
400	Bad Request: The request body is invalid (e.g., missing fields, new password too short).

Example Value | Model

```
{
  "error": "string"
}
```

401 Unauthorized: The provided OTP is incorrect, expired, or does not match the email address.

Example Value | Model

```
{
  "error": "string"
}
```

404 Not Found: The profile associated with the email address could not be found (e.g., it might have been deleted after the OTP was requested).

Example Value | Model

```
{
  "error": "string"
}
```

500 Internal Server Error: Something went wrong on the server (e.g., hashing the new password failed, database update failed).

Example Value | Model

```
{
  "error": "string"
}
```

POST `/auth/signup` Register a New User Account

Creates a new user profile in the system. This is the first step for a new user to start using the service.

You need to provide your desired `email`, a secure `password` (minimum 8 characters), your `first_name`, and `last_name`.
The server will securely hash the password before storing it (meaning the original password is never saved directly).
If the email address is already registered, the request will fail.

Parameters

Try it out

Name	Description
signup * required `object` *(body)*	User registration details. All fields except 'extra' are required.

Example Value | Model

```
{
  "email": "string",
  "extra": "string",
  "first_name": "string",
  "last_name": "string",
  "password": "stringst"
}
```

Parameter content type

application/json

Responses

Response content type application/json

Code	Description
201	Account Created Successfully. The response body contains the details of the newly created profile (excluding the password hash).

Example Value | Model

```
{
  "creation_date": "string",
  "email": "string",
  "extra": "string",
  "first_name": "string",
  "id": "string",
  "last_modified_date": "string",
  "last_name": "string",
  "password_hash": "string"
}
```

400 Bad Request: The data you sent is invalid (e.g., missing required fields, invalid email format, password too short) OR the email address is already in use by another account.

Example Value | Model

```
{
  "error": "string"
}
```

500 Internal Server Error: Something went wrong on the server while creating the account (e.g., password hashing failed, database connection issue).

Example Value | Model

```
{
  "error": "string"
}
```

Code	Description

Documents

GET `/documents` List and Search Your Documents

Retrieves a list of documents that the currently logged-in user has access to (either owned or shared with them).

This endpoint supports powerful filtering, sorting, and pagination using query parameters:

- `scope` : Control which documents to see:
- `owned` : Only documents you created.
- `shared` : Only documents shared with you by others.
- `all` (default): Both owned and shared documents.
- `content_query` : Filter documents based on their JSON content using a specific query language (details likely in separate documentation or examples). This allows searching within the document data itself. Example: `?content_query=metadata.status eq "published"`
- `sort_by` : Choose the field to sort results by: `creation_date` (default) or `last_modified_date` .
- `order` : Set the sort direction: `asc` (ascending) or `desc` (descending, default).
- `page` : For pagination, specify the page number (starts at 1, default is 1).
- `limit` : For pagination, specify the number of documents per page (default is 20, max is 100).

Example: `/documents?scope=owned&sort_by=last_modified_date&order=asc&page=1&limit=10` (Get the first 10 oldest modified documents owned by the user).

Parameters

Try it out

Name	Description
scope `string` *(query)*	Filter by ownership: 'owned', 'shared', or 'all'. *Available values* : owned, shared, all *Default value* : all *Example* : owned all
content_query `array[string]` *(query)*	Advanced filter based on document content (specific syntax applies). *Example* : user.name eq "John Doe"
sort_by `string` *(query)*	Field to sort results by. *Available values* : creation_date, last_modified_date *Default value* : creation_date *Example* : last_modified_date creation_date
order `string` *(query)*	Sorting direction. *Available values* : asc, desc *Default value* : desc *Example* : asc desc
page `integer` *(query)*	Page number for pagination (starts at 1). *Default value* : 1 *Example* : 2 1
limit `integer` *(query)*	Number of documents per page. *Default value* : 20 *Example* : 50 20

Responses **Response content type** application/json

Code	Description
200	A list of documents matching the criteria, along with pagination details (total count, current page, limit).

Example Value | Model

```
{
  "data": [
    {
      "content": "string",
      "creation_date": "string",
      "id": "string",
      "last_modified_date": "string",
      "owner_id": "string"
    }
  ],
  "limit": 0,
  "page": 0,
  "total": 0
}
```

Code	Description
400	Bad Request: One or more query parameters are invalid (e.g., invalid 'scope', incorrect 'content_query' syntax, non-integer 'page'/'limit').

Example Value | Model

```
{
```

Code	Description
	`"error": "string"` `}`
401	Unauthorized: Your access token is missing, invalid, or expired.
500	Internal Server Error: Something went wrong on the server while retrieving documents.

```
  "error": "string"
}
```

401 Unauthorized: Your access token is missing, invalid, or expired.

Example Value | Model

```
{
  "error": "string"
}
```

500 Internal Server Error: Something went wrong on the server while retrieving documents.

Example Value | Model

```
{
  "error": "string"
}
```

POST `/documents` Create a New Document

Allows a logged-in user to create and store a new document.

The document's `content` can be any valid JSON structure – an object (`{}`), an array (`[]`), a string (`""`), a number, a boolean (`true` / `false`), or `null` .
The server automatically assigns a unique ID to the document and records the user who created it (the owner) and the creation/modification timestamps.
You must provide your access token for authentication. The request body needs a `content` field containing the JSON data you want to store.

Example Request Body:

```
{
"content": {
"title": "My First Document",
"body": "This is the content.",
"tags": ["example", "getting started"]
}
}
```

Parameters Try it out

Name	Description
document * required `object` *(body)*	The JSON content you want to store in the new document.

Example Value | Model

```
{
  "content": "string"
}
```

Parameter content type
application/json

Responses **Response content type** application/json

Code	Description
201	Document Created Successfully. The response body contains the details of the newly created document, including its unique ID.
400	Bad Request: The request body is invalid. It must be valid JSON and contain the required 'content' field.
401	Unauthorized: Your access token is missing, invalid, or expired. You need to be logged in to create documents.
500	Internal Server Error: Something went wrong on the server while creating the document (e.g., database error).

201 — **Example Value** | Model

```
{
  "content": "string",
  "creation_date": "string",
  "id": "string",
  "last_modified_date": "string",
  "owner_id": "string"
}
```

400 — **Example Value** | Model

```
{
  "error": "string"
}
```

401 — **Example Value** | Model

```
{
  "error": "string"
}
```

500 — **Example Value** | Model

```
{
  "error": "string"
}
```

GET `/documents/{id}` Get a Specific Document by ID

Retrieves the full details of a single document using its unique identifier (`id`).

You can only retrieve a document if:

1. You are the owner of the document.
   OR
2. The document has been explicitly shared with you by its owner.

Provide the document's `id` as part of the URL path. You also need your access token for authentication.

Parameters — Try it out

Name	Description
id * required `string` *(path)*	The unique identifier of the document you want to retrieve. *Example* : doc_abc123xyz id

Responses — **Response content type** application/json

Code **Description**

200 Successfully retrieved the document. The response body contains the document's details (ID, owner, content, timestamps).

Example Value | Model

```
{
  "content": "string",
  "creation_date": "string",
  "id": "string",
  "last_modified_date": "string",
  "owner_id": "string"
}
```

400 Bad Request: The document ID provided in the URL path is missing or invalid.

Example Value | Model

```
{
  "error": "string"
}
```

401 Unauthorized: Your access token is missing, invalid, or expired.

Example Value | Model

```
{
  "error": "string"
}
```

403 Forbidden: You do not have permission to view this document. You are neither the owner nor has it been shared with you.

Example Value | Model

```
{
  "error": "string"
}
```

404 Not Found: No document exists with the specified ID.

Example Value | Model

```
{
  "error": "string"
}
```

500 Internal Server Error: Something went wrong on the server while retrieving the document.

Example Value | Model

```
{
  "error": "string"
}
```

PUT `/documents/{id}` Update a Document's Content

Replaces the *entire* existing content of a specific document with new content.

Important: This operation overwrites the previous content completely. If you only want to modify parts of the content, you should first retrieve the document, make changes to the content in your application, and then use this endpoint to save the full, modified content.

Only the user who originally created (owns) the document is allowed to update it.
Provide the document's `id` in the URL path and the new JSON `content` in the request body. Authentication via access token is required.

Example Request Body:

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

Parameters

Try it out

Name | **Description**

id * required
`string`
(path)

The unique identifier of the document to update.
Example : doc_abc123xyz

id

document * required
`object`
(body)

The new JSON content to replace the existing document content.

Example Value | Model

```
{
  "content": "string"
}
```

Parameter content type
application/json

Responses

Response content type application/json

Code | **Description**

200

Document Updated Successfully. The response body contains the complete document with the updated content and modification timestamp.

Example Value | Model

```
{
  "content": "string",
  "creation_date": "string",
  "id": "string",
  "last_modified_date": "string",
  "owner_id": "string"
}
```

400

Bad Request: The document ID in the path is missing/invalid, or the request body is invalid (must contain 'content' field with valid JSON).

Example Value | Model

```
{
  "error": "string"
}
```

401

Unauthorized: Your access token is missing, invalid, or expired.

Example Value | Model

```
{
  "error": "string"
}
```

403

Forbidden: You are not the owner of this document, so you cannot update it.

Example Value | Model

Code	Description
	```{ "error": "string" }```
404	Not Found: No document exists with the specified ID.
500	Internal Server Error: Something went wrong on the server while updating the document.

```
{
  "error": "string"
}
```

404 Not Found: No document exists with the specified ID.

Example Value | Model

```
{
  "error": "string"
}
```

500 Internal Server Error: Something went wrong on the server while updating the document.

Example Value | Model

```
{
  "error": "string"
}
```

DELETE `/documents/{id}` Delete a Document

Permanently deletes a specific document from the system.

WARNING: This action is irreversible! Once deleted, the document cannot be recovered.
Any records indicating this document was shared with others will also be removed.

Only the user who originally created (owns) the document is allowed to delete it.
Provide the document's `id` in the URL path. Authentication via access token is required.

Parameters Try it out

Name	Description
id * required `string` *(path)*	The unique identifier of the document to delete. *Example* : doc_abc123xyz id

Responses **Response content type** application/json

Code	Description
204	Document Deleted Successfully. No content is returned in the response body because the resource no longer exists.
400	Bad Request: The document ID provided in the URL path is missing or invalid.

Example Value | Model

```
{
  "error": "string"
}
```

401 Unauthorized: Your access token is missing, invalid, or expired.

Example Value | Model

Code	Description
	`{ "error": "string" }`
403	Forbidden: You are not the owner of this document, so you cannot delete it. **Example Value** | Model `{ "error": "string" }`
404	Not Found: No document exists with the specified ID. (Note: The API might return 204 even if not found, treating deletion of a non-existent item as success). **Example Value** | Model `{ "error": "string" }`
500	Internal Server Error: Something went wrong on the server while deleting the document. **Example Value** | Model `{ "error": "string" }`

Sharing

GET `/documents/{id}/shares` See Who a Document is Shared With

Retrieves a list of user profile IDs that a specific document has been shared with.

Only the user who originally created (owns) the document can use this endpoint to see who they've shared it with.
Provide the document's `id` in the URL path. Authentication via access token is required.
If the document hasn't been shared with anyone, it returns an empty list.

Parameters Try it out

Name	Description
id * required `string` *(path)*	The unique identifier of the document whose share list you want to view. *Example* : doc_abc123xyz id

Responses **Response content type** application/json

Code	Description
200	Successfully retrieved the list of profile IDs the document is shared with. The 'shared_with' array contains the IDs.

Code	Description
	Example Value | Model

```
{
  "shared_with": [
    "string"
  ]
}
```

400 Bad Request: The document ID provided in the URL path is missing or invalid.

Example Value | Model

```
{
  "error": "string"
}
```

401 Unauthorized: Your access token is missing, invalid, or expired.

Example Value | Model

```
{
  "error": "string"
}
```

403 Forbidden: You are not the owner of this document, so you cannot view its share list.

Example Value | Model

```
{
  "error": "string"
}
```

404 Not Found: No document exists with the specified ID.

Example Value | Model

```
{
  "error": "string"
}
```

500 Internal Server Error: Something went wrong on the server while retrieving the share list.

Example Value | Model

```
{
  "error": "string"
}
```

PUT `/documents/{id}/shares` Set/Replace Who a Document is Shared With

Completely replaces the list of users a specific document is shared with.

Provide a JSON array named `shared_with` in the request body, containing the profile IDs of the users you want to share the document with.
Important: Any users previously shared with, but *not* included in the new list, will lose access.
To remove *all* shares for a document, send an empty array: `{"shared_with": []}`.

Only the document owner can perform this operation. You cannot share a document with yourself (the owner).
Provide the document's `id` in the URL path. Authentication via access token is required.

Example Request Body (Share with user 'user_123' and 'user_456'):

```
{
"shared_with": ["user_123", "user_456"]
}
```

Parameters Try it out

Name	Description
id * required `string` *(path)*	The unique identifier of the document whose share list you want to set/replace. *Example* : doc_abc123xyz id
shareRequest * required `object` *(body)*	A JSON object containing the 'shared_with' key, whose value is an array of profile IDs. **Example Value** | Model

```
{
  "shared_with": [
    "string"
  ]
}
```

Parameter content type
application/json

Responses **Response content type** application/json

Code	Description
204	Share List Updated Successfully. No content is returned in the response body.
400	Bad Request: The request body is invalid (e.g., missing 'shared_with' array, invalid JSON) OR you tried to include the owner's ID in the 'shared_with' list.

Example Value | Model

```
{
  "error": "string"
}
```

401 Unauthorized: Your access token is missing, invalid, or expired.

Example Value | Model

```
{
  "error": "string"
}
```

403 Forbidden: You are not the owner of this document, so you cannot modify its share list.

Example Value | Model

```
{
  "error": "string"
}
```

404 Not Found: No document exists with the specified ID.

Example Value | Model

```
{
  "error": "string"
}
```

Code	Description
500	Internal Server Error: Something went wrong on the server while updating the share list.

Example Value | Model

```
{
  "error": "string"
}
```

PUT `/documents/{id}/shares/{profile_id}` Share a Document with One User

Adds a single specified user (by their `profile_id`) to the list of users who can access a specific document.

This operation is *additive* – it doesn't affect other users the document might already be shared with.
It's also *idempotent*, meaning if you try to add a user who already has access, the operation succeeds without making any changes.

Only the document owner can perform this operation. You cannot share a document with yourself (the owner).
Provide the document's `id` and the target user's `profile_id` in the URL path. Authentication via access token is required.

Parameters Try it out

Name	Description
id * required `string` *(path)*	The unique identifier of the document you want to share. *Example* : doc_abc123xyz id
profile_id * required `string` *(path)*	The unique identifier of the user profile you want to grant access to. *Example* : user_123 profile_id

Responses **Response content type** application/json

Code	Description
204	User Added to Share List Successfully (or was already shared with). No content is returned.
400	Bad Request: You tried to share the document with its owner (yourself).

Example Value | Model

```
{
  "error": "string"
}
```

Code	Description
401	Unauthorized: Your access token is missing, invalid, or expired.

Example Value | Model

```
{
  "error": "string"
}
```

Code	Description
403	Forbidden: You are not the owner of this document, so you cannot share it.

Example Value | Model

```
{
  "error": "string"
}
```

Code	Description
404	Not Found: The specified Document ID or Profile ID does not exist, or the IDs were missing from the URL path.

Example Value | Model

```
{
  "error": "string"
}
```

Code	Description
500	Internal Server Error: Something went wrong on the server while adding the user to the share list.

Example Value | Model

```
{
  "error": "string"
}
```

DELETE `/documents/{id}/shares/{profile_id}` Stop Sharing a Document with One User

Removes a single specified user (by their `profile_id`) from the list of users who can access a specific document.

This operation only affects the specified user; other users the document is shared with remain unaffected.
It's *idempotent*, meaning if you try to remove a user who doesn't currently have access (or never did), the operation succeeds without error.

Only the document owner can perform this operation.
Provide the document's `id` and the target user's `profile_id` (the one to remove) in the URL path. Authentication via access token is required.

Parameters Try it out

Name	Description
id * required string *(path)*	The unique identifier of the document you want to modify shares for. *Example* : doc_abc123xyz id
profile_id * required string *(path)*	The unique identifier of the user profile whose access you want to revoke. *Example* : user_123 profile_id

Responses Response content type application/json

Code	Description
204	User Removed from Share List Successfully (or was not shared with). No content is returned.

Code	Description
401	Unauthorized: Your access token is missing, invalid, or expired.
403	Forbidden: You are not the owner of this document, so you cannot modify its shares.
404	Not Found: The specified Document ID or Profile ID does not exist, or the IDs were missing from the URL path.
500	Internal Server Error: Something went wrong on the server while removing the user from the share list.

401 — **Example Value** | Model

```
{
  "error": "string"
}
```

403 — **Example Value** | Model

```
{
  "error": "string"
}
```

404 — **Example Value** | Model

```
{
  "error": "string"
}
```

500 — **Example Value** | Model

```
{
  "error": "string"
}
```

Profiles

GET `/profiles` Search User Profiles

Allows authenticated users to search for other user profiles within the system.

You can filter the search using query parameters in the URL:

- `email` : Find profiles where the email address contains the provided text (case-insensitive). Example: `?email=test.com`
- `first_name` : Find profiles where the first name contains the provided text (case-insensitive). Example: `?first_name=jo`
- `last_name` : Find profiles where the last name contains the provided text (case-insensitive). Example: `?last_name=smi`
  You can combine multiple filters. The search returns profiles that match *all* provided filters.

Results are paginated to handle potentially large numbers of users:

- `page` : Specifies which page of results to retrieve (starts at 1). Default is 1. Example: `?page=2`
- `limit` : Specifies how many profiles to return per page. Default is 20, maximum is 100. Example: `?limit=50`

Example combining filters and pagination: `/profiles?first_name=a&page=1&limit=10` (Find profiles with 'a' in the first name, show the first 10 results).

Parameters Try it out

Name	Description
email string *(query)*	Filter profiles where email contains this text (case-insensitive). *Example* : user@example.com email

Name	Description
first_name string *(query)*	Filter profiles where first name contains this text (case-insensitive). *Example* : John first_name
last_name string *(query)*	Filter profiles where last name contains this text (case-insensitive). *Example* : Doe last_name
page integer *(query)*	Page number for results (starts at 1). *Default value* : 1 *Example* : 1 1
limit integer *(query)*	Number of profiles per page. *Default value* : 20 *Example* : 20 20

Responses

Response content type application/json

Code **Description**

200 A list of profiles matching the search criteria, along with pagination details (total count, current page, limit).

Example Value | Model

```
{
  "data": [
    {
      "creation_date": "string",
      "email": "string",
      "extra": "string",
      "first_name": "string",
      "id": "string",
      "last_modified_date": "string",
      "last_name": "string"
    }
  ],
  "limit": 0,
  "page": 0,
  "total": 0
}
```

400 Bad Request: Invalid query parameters. 'page' and 'limit' must be positive integers. 'limit' cannot exceed 100.

Example Value | Model

```
{
  "error": "string"
}
```

401 Unauthorized: Your access token is missing, invalid, or expired. You need to be logged in to search profiles.

Example Value | Model

```
{
  "error": "string"
}
```

Code	Description
500	Internal Server Error: Something went wrong on the server while searching for profiles.

Example Value | Model

```
{
  "error": "string"
}
```

GET `/profiles/me` Get Your Own Profile

Retrieves the profile details (like first name, last name, email, creation date) for the user who is currently logged in.

Think of this as your "My Account" page data. To use this endpoint, you must first authenticate (log in) to get an access token.
The server uses the access token you provide in the request header to figure out who you are and fetch your specific profile information from the database.

Parameters Try it out

No parameters

Responses **Response content type** application/json

Code	Description
200	Your profile details were successfully retrieved. The response body contains your profile information (excluding sensitive data like the password hash).

Example Value | Model

```
{
  "creation_date": "string",
  "email": "string",
  "extra": "string",
  "first_name": "string",
  "id": "string",
  "last_modified_date": "string",
  "last_name": "string",
  "password_hash": "string"
}
```

401 Unauthorized: Your access token is missing, invalid, or expired. You might need to log in again.

Example Value | Model

```
{
  "error": "string"
}
```

404 Not Found: The server couldn't find a profile associated with your access token. This is unusual if your token is valid.

Example Value | Model

```
{
  "error": "string"
}
```

500 Internal Server Error: Something went wrong on the server side (e.g., a database connection issue or a problem reading your user ID from the token context).

Code	Description
	Example Value | Model

```
{
  "error": "string"
}
```

PUT `/profiles/me` Update Your Own Profile

Allows the currently logged-in user to update their own profile information.

You can change your `first_name` , `last_name` , and any custom `extra` data associated with your profile.
Important: You *cannot* change your email address or password using this endpoint. Password changes typically have a separate, more secure process (like a password reset flow).
You need to provide your current access token for authentication. The request body should contain the fields you want to update in JSON format.

Parameters Try it out

Name	Description
profile * required `object` *(body)*	The profile fields you want to update. 'first_name' and 'last_name' are required.

Example Value | Model

```
{
  "extra": "string",
  "first_name": "string",
  "last_name": "string"
}
```

Parameter content type
application/json

Responses Response content type application/json

Code	Description
200	Your profile was successfully updated. The response body contains the complete, updated profile.

Example Value | Model

```
{
  "creation_date": "string",
  "email": "string",
  "extra": "string",
  "first_name": "string",
  "id": "string",
  "last_modified_date": "string",
  "last_name": "string",
  "password_hash": "string"
}
```

Code	Description
400	Bad Request: The data you sent in the request body is invalid. This could be due to missing required fields ('first_name', 'last_name') or incorrect JSON formatting.

Example Value | Model

```
{
  "error": "string"
```

Code	Description
	`}`
401	Unauthorized: Your access token is missing, invalid, or expired. You need to be logged in to update your profile.
404	Not Found: The server couldn't find your profile based on your access token.
500	Internal Server Error: Something went wrong on the server while trying to update your profile (e.g., a database error).

401 — **Example Value** | Model

```
{
  "error": "string"
}
```

404 — **Example Value** | Model

```
{
  "error": "string"
}
```

500 — **Example Value** | Model

```
{
  "error": "string"
}
```

DELETE `/profiles/me` Delete Your Own Profile

Permanently deletes the account and profile data for the currently logged-in user.

WARNING: This action is irreversible! Once you delete your account, all your data associated with it (profile, documents you own, etc.) will be removed.
(Developer Note: Full cascading delete logic, like removing shared document access, might still be under development. Currently, it primarily removes the main profile record.)
You must provide your valid access token to authorize this action.

Parameters Try it out

No parameters

Responses **Response content type** application/json

Code	Description
204	Account Successfully Deleted. No content is returned in the response body because the resource (your profile) no longer exists.
401	Unauthorized: Your access token is missing, invalid, or expired. You need to be logged in to delete your account.

401 — **Example Value** | Model

```
{
  "error": "string"
}
```

Code	Description
404	Not Found: The server couldn't find your profile based on your access token.
500	Internal Server Error: Something went wrong on the server while trying to delete your account (e.g., a database error).

Example Value | Model

```
{
  "error": "string"
}
```

Example Value | Model

```
{
  "error": "string"
}
```

Models

```
api.CreateDocumentRequest {
    content*                  {
                                description:      Content can be any valid JSON
                              }
}
```

```
api.ForgotPasswordRequest {
    email*                    string
}
```

```
api.GetDocumentsResponse {
    data                      [models.Document {
                                content               {
                                                        description:     Can be any JSON structure or simple text
                                                      }
                                creation_date         string
                                                      UTC
                                id                    string
                                                      Unique ID (UUID, dashless)
                                last_modified_date    string
                                                      UTC
                                owner_id              string
                                                      Profile ID of the owner
                              }]
    limit                     integer
    page                      integer
    total                     integer
}
```

```
api.GetSharersResponse {
    shared_with             [
                            List of Profile IDs (dashless)

                            string]
}
```

```
api.LoginRequest {
    email*                  string
    password*               string
}
```

```
api.LoginResponse {
    token                   string
}
```

```
api.ProfileResponse {
    creation_date           string
    email                   string
    extra
                            {
                            }
    first_name              string
    id                      string
    last_modified_date      string
    last_name               string
}
```

```
api.ResetPasswordRequest {
    email*                  string
    new_password*           string
                            minLength: 8
    otp*                    string
}
```

```
api.SearchProfilesResponse {
    data
                            [api.ProfileResponse {...}]
    limit                   integer
    page                    integer
    total                   integer
}
```

```
api.SetSharersRequest {
    shared_with*
                            [
                            Use pointer to distinguish between empty list and not provided?
                            No, binding:"required" means it must be present, even if empty array [] .

                            string]
}
```

```
api.SignupRequest ⌄ {
    email*                 string
    extra
                            ⌄ {
                           }
    first_name*            string
    last_name*             string
    password*              string
                           minLength: 8

                           Add basic password length validation

}
```

```
api.UpdateDocumentRequest ⌄ {
    content*
                            ⌄ {
                           }
}
```

```
api.UpdateProfileRequest ⌄ {
    extra
                            ⌄ {
                           }
    first_name*            string
    last_name*             string
}
```

```
models.Document ⌄ {
    content
                            ⌄ {
                             description:
                                                     Can be any JSON structure or simple text

                           }
    creation_date          string

                           UTC

    id                     string

                           Unique ID (UUID, dashless)

    last_modified_date     string

                           UTC

    owner_id               string

                           Profile ID of the owner

}
```

models.Profile {

```
    creation_date         string

                          UTC

    email                 string

                          Unique, used for login

    extra
                            {
                            description:
                                                      User-defined data

                          }
    first_name            string
    id                    string

                          Unique ID (UUID, dashless)

    last_modified_date    string

                          UTC

    last_name             string
    password_hash         string

                          Store hash, include in JSON persistence.

}
```

utils.APIError {

```
    error                 string
}
```