# SnipBox API Documentation

## Authentication

### 1. Obtain JWT Token (1. Login API)

* Endpoint: POST /login/
* Request:

{

"username": "testuser",

"password": "testpassword"

}

* Response:

{

"access": "<jwt\_access\_token>",

"refresh": "<jwt\_refresh\_token>"

}

* cURL:

curl -X POST http://127.0.0.1:8000/login/ \

-H "Content-Type: application/json" \

-d '{"username": "testuser", "password": "testpassword"}'  
  
  
**2. Refresh JWT Token (2. Refresh token API)**

* Endpoint: POST /token/refresh/
* Request:

{

"refresh": "<jwt\_refresh\_token>"

}

* Response:

{

"access": "<new\_jwt\_access\_token>"

}

* cURL:

curl -X POST http://127.0.0.1:8000/token/refresh/ \

-H "Content-Type: application/json" \

-d '{"refresh": "<jwt\_refresh\_token>"}'

### **3. Create a New User – ADMIN ACCESS ONLY**

* Endpoint: POST /create-user/
* Request:

{

"username": "newuser4",

"password": "password123",

"email": "newuser@example.com",

"first\_name": "New",

"last\_name": "User"

}

* Response:

{

"username": "newuser5",

"email": "newuser@example.com",

"first\_name": "New",

"last\_name": "User"

}

* cURL:

curl -X POST http://127.0.0.1:8000/create-user/ \

-H "Content-Type: application/json" \

-d '{"username": "newuser", "email": "newuser@example.com", "password": "newpassword","first\_name": "New","last\_name": "User"}'

## **Snippets API**

### 4. Create a Snippet (Create API)

* Endpoint: POST /snippets/create/
* Request:

{

"snippet\_title": "TEST!",

"note": "This is a test",

"tags": ["TEST", "TEST2"]

}

* Response:

{

"id": 10,

"snippet\_title": "TEST!",

"note": "This is a test",

"tag\_details": [

{

"id": 1,

"tag\_title": "TEST"

},

{

"id": 13,

"tag\_title": "TEST2"

}

],

"created\_at": "2025-02-01T07:15:54.302617Z",

"updated\_at": "2025-02-01T07:15:54.302642Z" }

* cURL

curl -X POST http://127.0.0.1:8000/snippets/create/ \

-H "Content-Type: application/json" \

-H "Authorization: Bearer <jwt\_access\_token>" \

-d '{"snippet\_title": "TEST!", "note": “This is a test .", "tags": ["TEST", "TEST2"]}'

5. List All Snippets (Overview API (total count, listing))

Description: Overview API (total count, listing)

Total number of snippets and list all available snippets with a hyperlink to respective detail APIs.

* Endpoint: GET /snippets/
* Response:

{

"total\_count": 2,

"snippets": [

{

"id": 1,

"url": "http://127.0.0.1:8000/api/snippets/1/"

},

{

"id": 2,

"url": "http://127.0.0.1:8000/api/snippets/2/"

}

]

}

* cURL

curl -X GET http://127.0.0.1:8000/api/snippets/ \

-H "Authorization: Bearer <jwt\_access\_token>"

## Tags API

### 6. Create a Tag

A. Endpoint: POST /api/tags/

* Request:

{

"tag\_title": "tag26"

}

* Response:  
   {

"id": 15,

"tag\_title": "tag26" }

* cURL:

curl -X POST http://127.0.0.1:8000/api/tags/ \

-H "Content-Type: application/json" \

-H "Authorization: Bearer <jwt\_access\_token>" \

-d '{"tag\_title": "tag26"}'

B. Endpoint: GET /api/tags/

Tag list API

API to list tags

* Response:

[{

"id": 1,

"tag\_title": "TEST"

},

{

"id": 2,

"tag\_title": "tag1"

},

{

"id": 3,

"tag\_title": "tag4"

}]

* cURL

curl -X GET http://127.0.0.1:8000/api/tags/ \

-H "Authorization: Bearer <jwt\_access\_token>"

### 7. Reterive , Delete , Patch a Tag

A.Endpoint: GET /api/tags/{id}/

*1. GET: Retrieve a tag by its ID*

* Description: This method retrieves a tag's details based on the provided id.:
* Request

{

"id": 1,

"tag\_title": "TEST"

}

{id}: The unique identifier of the tag.

* Response:

{

"id": 1,

"tag\_title": "TEST"

}

* **cURL**

curl -X GET http://127.0.0.1:8000/api/tags/1/ \

-H "Authorization: Bearer <jwt\_access\_token>"

#### 2. **DELETE**: Delete a tag by its ID

* Description: This method deletes a tag from the database.

Endpoint: DELETE /api/tags/{id}/

Response:204 No Content: The tag was successfully deleted.

**cURL:**

curl -X DELETE http://127.0.0.1:8000/api/tags/1/ \

-H "Authorization: Bearer <jwt\_access\_token>"

##### 3. **PATCH**: Update a tag by its ID

* Description: This method updates the details of an existing tag.

Request:

{

"tag\_title": "NEW\_TITLE"

}

Response:

{

"id": 1,

"tag\_title": "NEW\_TITLE"

}

cURL:

curl -X PATCH http://127.0.0.1:8000/api/tags/1/ \

-H "Authorization: Bearer <jwt\_access\_token>" \

-H "Content-Type: application/json" \

-d '{"tag\_title": "NEW\_TITLE"}'

### 8. Tag Detail API Documentation

#### Endpoint: /api/tags-detail/{tag\_id}/

This endpoint returns all snippets that are linked to the selected tag, identified by the tag\_id. It supports the GET method.

* Request:

GET /api/tags-detail/{tag\_id}/

{tag\_id}: The unique identifier of the tag whose linked snippets are being retrieved.

* Response:

{

"id": 1,

"tag\_title": "TEST",

"snippets": [

{

"id": 1,

"snippet\_title": "Test 1",

"note": "Test",

"tag\_details": [

{

"id": 1,

"tag\_title": "TEST"

}

],

"created\_at": "2025-01- 31T14:41:24.669319Z",

"updated\_at": "2025-01- 31T14:41:24.669343Z"

},

{

"id": 2,

"snippet\_title": "Sample Snippet",

"note": "daa",

"tag\_details": [

{

"id": 1,

"tag\_title": "TEST"

}

],

"created\_at": "2025-01- 31T16:27:45.019472Z",

"updated\_at": "2025-01- 31T18:18:24.267358Z"

},

{

"id": 4,

"snippet\_title": "Sample Snippet",

"note": "test",

"tag\_details": [

{

"id": 1,

"tag\_title": "TEST"

}

],

"created\_at": "2025-01- 31T17:19:56.512504Z",

"updated\_at": "2025-01- 31T18:18:29.397920Z"

}

]

}

* cURL:

curl -X GET http://127.0.0.1:8000/api/tags-detail/1/ \

-H "Authorization: Bearer <jwt\_access\_token>"

9.Snippet Detail API

GET: Retrieve all snippets created by the current user

API should display the snippet title, note, and timestamp information. Only show the snippet created by the current user.

* Request

GET /api/snippets\_details/

* **Response**

[

{

"id": 8,

"snippet\_title": "My First Note",

"note": "This is an important note.",

"created\_at": "2025-02-01T06:43:53.278284Z",

"updated\_at": "2025-02-01T06:43:53.278310Z"

},

{

"id": 9,

"snippet\_title": "My Second Note",

"note": "This is an important note.",

"created\_at": "2025-02-01T06:44:18.687487Z",

"updated\_at": "2025-02-01T06:44:18.687506Z"

},

{

"id": 10,

"snippet\_title": "TEST!",

"note": "This is a test",

"created\_at": "2025-02-01T07:15:54.302617Z",

"updated\_at": "2025-02-01T07:15:54.302642Z"

}

]

* **cURL**

curl -X GET http://127.0.0.1:8000/api/snippets\_details/ \

-H "Authorization: Bearer <jwt\_access\_token>"

**10. Snippet Detail API Documentation (PATCH & DELETE Methods)**

* **Endpoint**: /api/snippets\_details/{snippet\_id}/

{snippet\_id}: The unique identifier of the snippet to be updated.

1. PATCH: Update a snippet by its ID

* Request

{

"snippet\_title": "TEST!0011",

"note": "This is a test",

"created\_at": "2025-02-01T07:15:54.302617Z",

"updated\_at": "2025-02-01T07:15:54.302642Z"

}

* Response

200 OK: Returns the updated snippet details.

{

"id": 10,

"snippet\_title": "TEST!0011",

"note": "This is a test",

"created\_at": "2025-02-01T07:15:54.302617Z",

"updated\_at": "2025-02-01T07:15:54.302642Z"

}

* cURL

curl -X PATCH http://127.0.0.1:8000/api/snippets\_details/10/ \

-H "Authorization: Bearer <jwt\_access\_token>" \

-H "Content-Type: application/json" \

-d '{"snippet\_title": "TEST!0011", "note": "This is a test"}'

**2. DELETE: Delete a snippet by its ID**

* Description: This method deletes a specific snippet by its ID.
* Request:

DELETE /api/snippets\_details/{snippet\_id}/

* Response:

204 No Content: The snippet was successfully deleted.

404 Not Found: If the snippet with the given ID does not exist.

* CURL

curl -X DELETE http://127.0.0.1:8000/api/snippets\_details/10/ \

-H "Authorization: Bearer <jwt\_access\_token>"