Unit Tests - Miscellaneous DB Functions

Test Case 1: Create Document

Objective:

Ensure the system can handle the creation of a document in the `miscellaneous` collection and return an HTTP 201 Created status.

Method:

`test_create`

Test Steps:

1. Define the data to be inserted into the `miscellaneous` collection.
2. Send a POST request to the `miscellaneous` endpoint with the defined data.
3. Assert that the HTTP response status is 201 Created.
4. Count the number of documents in the `miscellaneous` collection.
5. Assert that the document count is 1.

Expected Result:

The system should successfully create a document in the `miscellaneous` collection, return an HTTP 201 Created status, and the collection should contain exactly one document.

Test Case 2: Retrieve All Documents

Objective:

Ensure the system can retrieve all documents from the `miscellaneous` collection and return an HTTP 200 OK status.

Method:

`test_get_all`

Test Steps:

1. Send a GET request to the `miscellaneous` endpoint.
2. Assert that the HTTP response status is 200 OK.

Expected Result:

The system should successfully retrieve all documents from the `miscellaneous` collection, and the response should be HTTP 200 OK.

Test Case 3: Update Document - Success

Objective:

Ensure the system can handle the update of a document in the `miscellaneous` collection and return an HTTP 200 OK status.

Method:

`test_put`

Test Steps:

1. Define the data to update the document in the `miscellaneous` collection.
2. Send a PUT request to the `miscellaneous` endpoint with the defined data.
3. Assert that the HTTP response status is 200 OK.

Expected Result:

The system should successfully update the document in the `miscellaneous` collection, and the response should be HTTP 200 OK.

Test Case 3: Update Document - Invalid Data

Objective:

Test the system's response when attempting to update a document with incomplete or invalid data.

Method:

`test_put_not_valid`

Test Steps:

1. Define the invalid data to update the document in the `miscellaneous` collection.
2. Send a PUT request to the `miscellaneous` endpoint with the invalid data.
3. Assert that the HTTP response status is 400 Bad Request.

Expected Result:

The system should return an HTTP 400 Bad Request status due to the incomplete or invalid data provided for the update.