**Test Cases for Hybrid Search API**

**Test Case 1: Basic Keyword Search**

**Objective:** Verify that the API can handle basic keyword searches in the title, author, and content fields.

**Request:**

* **Method:** POST
* **Endpoint:** /search
* **Request Body:**

json

{

"query": "Science"

}

**Expected Response:**

* **Status Code:** 200 OK
* **Response Body:**

json

[

{

"id": 1,

"title": "Exploring the Universe",

"author": "Jane Doe",

"publication\_date": "2023-01-15",

"category": "Science",

"content": "This article explores various aspects of the universe...",

"vector\_representation": [0.1, 0.1, ..., 0.1]

}

]

**Notes:**

* Ensure that the search results contain records where the keyword "Science" appears in the title, author, or content.

**Test Case 2: Basic Vector Search**

**Objective:** Verify that the API can handle vector-based searches based on vector similarity.

**Request:**

* **Method:** POST
* **Endpoint:** /search
* **Request Body:**

json

{

"vector": [0.1, 0.1, ..., 0.1] // 300-dimensional vector

}

**Expected Response:**

* **Status Code:** 200 OK
* **Response Body:**

json

[

{

"id": 1,

"title": "Exploring the Universe",

"author": "Jane Doe",

"publication\_date": "2023-01-15",

"category": "Science",

"content": "This article explores various aspects of the universe...",

"vector\_representation": [0.1, 0.1, ..., 0.1]

}

]

**Notes:**

* Ensure that the search results contain records with vector representations similar to the provided vector.

**Test Case 3: Hybrid Search with Keyword and Vector**

**Objective:** Verify that the API combines keyword and vector searches to return the most relevant results.

**Request:**

* **Method:** POST
* **Endpoint:** /search
* **Request Body:**

json

{

"query": "Science",

"vector": [0.1, 0.1, ..., 0.1] // 300-dimensional vector

}

**Expected Response:**

* **Status Code:** 200 OK
* **Response Body:**

json

[

{

"id": 1,

"title": "Exploring the Universe",

"author": "Jane Doe",

"publication\_date": "2023-01-15",

"category": "Science",

"content": "This article explores various aspects of the universe...",

"vector\_representation": [0.1, 0.1, ..., 0.1]

}

]

**Notes:**

* Ensure that the results combine the relevance of both the keyword "Science" and the vector similarity.

**Test Case 4: Empty Query Parameters**

**Objective:** Verify the API's behavior when no query parameters are provided.

**Request:**

* **Method:** POST
* **Endpoint:** /search
* **Request Body:**

json

{}

**Expected Response:**

* **Status Code:** 400 Bad Request
* **Response Body:**

json

{

"error": "Query parameters are missing."

}

**Notes:**

* Ensure the API validates input and returns a proper error message if no query parameters are provided.

**Test Case 5: Invalid Vector Dimensions**

**Objective:** Verify the API's behavior when provided with vectors of incorrect dimensions.

**Request:**

* **Method:** POST
* **Endpoint:** /search
* **Request Body:**

json

{

"vector": [0.1, 0.1, ..., 0.1] // 250-dimensional vector

}

**Expected Response:**

* **Status Code:** 400 Bad Request
* **Response Body:**

json

{

"error": "Invalid vector dimensions. Expected 300 dimensions."

}

**Notes:**

* Ensure the API checks the vector dimensions and returns an appropriate error if dimensions are incorrect.

**Test Case 6: Large Dataset Performance**

**Objective:** Verify the API's performance when querying a large dataset.

**Request:**

* **Method:** POST
* **Endpoint:** /search
* **Request Body:**

json

{

"query": "Technology",

"vector": [0.1, 0.1, ..., 0.1] // 300-dimensional vector

}

**Expected Response:**

* **Status Code:** 200 OK
* **Response Time:** Should be within acceptable limits (e.g., under 2 seconds).
* **Response Body:** Relevant search results based on keyword and vector.

**Notes:**

* Ensure the API performs efficiently with large datasets and provides results within acceptable response times.

**Test Case 7: Non-Existent Records**

**Objective:** Verify the API's behavior when no records match the search criteria.

**Request:**

* **Method:** POST
* **Endpoint:** /search
* **Request Body:**

json

{

"query": "NonExistentKeyword",

"vector": [0.1, 0.1, ..., 0.1] // 300-dimensional vector

}

**Expected Response:**

* **Status Code:** 200 OK
* **Response Body:**

json

[]

**Notes:**

* Ensure the API correctly handles cases where no records match the search criteria and returns an empty array.

**Test Case 8: Invalid Endpoint**

**Objective:** Verify the API's behavior when accessing an invalid endpoint.

**Request:**

* **Method:** POST
* **Endpoint:** /invalid-endpoint
* **Request Body:**

json

{

"query": "Science"

}

**Expected Response:**

* **Status Code:** 404 Not Found
* **Response Body:**

json

{

"error": "Endpoint not found."

}

**Notes:**

* Ensure that the API correctly handles requests to invalid endpoints and returns a proper 404 error.