Endpoint Being Tested:
http://127.0.0.1:5000/faq_suggestions/<lecture_id>

Case: *Successful FAQ suggestion retrieval*

Request Method: GET

Inputs:

```
lecture_id: 22
```

Expected Output:

```
HTTP Status Code: 200 and field 'faq_suggestions' in JSON
```

Actual Output:


```
HTTP Status Code: 200
JSON: {"lecture_id": 22, "faq_suggestions": "The students' questions reveal a
significant focus on one key concept: nesting `if-else` statements within loops.
Since \"Can the if-else statements be used inside of loops?\" is asked repeatedly
(five times), it should be the primary focus of your proactive
addressing.\n\nHere's a comprehensive strategy to address these student questions
proactively in future sessions:\n\n**1.  Addressing the Frequent Question (\"Can
if-else statements be used inside loops?\"):**\n\n* **Begin with a clear and
concise \"yes.\"**  Don't beat around the bush.  This establishes the core answer
immediately.\n* **Provide a simple, illustrative example:** Show a code snippet
demonstrating a `for` loop (or `while` loop) containing an `if-else` statement.
Keep it short, focused, and easily understandable.  For example:\n\n```python\nfor
i in range(10):\n    if i % 2 == 0:\n        print(f\"{i} is even\")\n    else:\n
print(f\"{i} is odd\")\n```\n\n* **Explain the logic and flow:**  Step through the
example line by line, explaining how the loop iterates and how the `if-else`
statement modifies the behavior within each iteration.  Visual aids (like a
flowchart) can be incredibly helpful here.\n* **Expand on use cases:** Discuss
practical scenarios where this nesting is useful.  Examples might include:\n    *
Processing elements in a list conditionally.\n    * Implementing search
algorithms.\n    * Controlling program flow based on data within a loop.\n*
**Address potential misconceptions:**  Some students might struggle to visualize
the nested execution.  Explicitly address this by highlighting that the `if-else`
block executes *for each iteration* of the loop.\n\n**2. Addressing the Less
Frequent Question (\"Do we actually need to use if-else statements in all codes?
\"):**\n\n* **Answer with a definitive \"no.\"**  Many programs don't require
conditional logic.\n* **Provide examples:** Show simple programs that don't use
`if-else` statements\u2014perhaps a program that just calculates a sum or prints a
sequence of numbers.\n* **Explain the purpose of `if-else`:** Emphasize that `if-
else` statements are tools for *controlling program flow* based on conditions.
```

```
They are used when the program's behavior needs to change based on different
inputs or situations.  They are not mandatory for all programs.\n\n**3. Proactive
Teaching Strategies:**\n\n* **Preemptive introduction:** In your lecture *before*
introducing loops, briefly mention the possibility of nesting control structures
like `if-else` inside loops. This creates a mental framework for students.\n*
**Incorporate examples throughout:** Don't just present the concept in isolation.
Integrate nested `if-else` statements into many different examples throughout your
lectures on loops and conditional statements.\n* **Encourage questions:** Create a
safe and welcoming classroom environment where students feel comfortable asking
questions, even if they seem basic.\n* **Use diverse examples:**  Show examples
with different types of loops (for, while) and different types of conditions.
This will help students grasp the broader applicability of the concept.\n\n\nBy
focusing on the most frequent question and incorporating these proactive teaching
strategies, you can significantly reduce future confusion and enhance student
understanding."}
```


Result: Success

Pytest Code:

```
def test_faq_suggestions_success(client):
    lecture_id = 22
    endpoint = f"/faq_suggestions/{lecture_id}"

    response = client.get(endpoint)
    data = response.get_json()

    expected_status = 200
    result = "Success" if response.status_code == expected_status and
"faq_suggestions" in data else "Failed"

    write_test_doc(
        title="***Case:*** *Successful FAQ suggestion retrieval*",
        endpoint=endpoint,
        method="GET",
        inputs=f"lecture_id: {lecture_id}",
        expected="HTTP Status Code: 200 and field 'faq_suggestions' in JSON",
        actual=f"HTTP Status Code: {response.status_code}\nJSON:
{json.dumps(data)}",
        result=result
    )

    assert response.status_code == 200
    assert "faq_suggestions" in data
```

Case: *Lecture ID with no questions in Firestore*

Request Method: GET

Inputs:

```
lecture_id: 99999
```

Expected Output:

```
HTTP Status Code: 200 and message: 'No questions/doubts asked by the students for
this lecture.'
```

Actual Output:

```
HTTP Status Code: 200
JSON: {"lecture_id": 99999, "faq_suggestions": "No questions/doubts asked by the
students for this lecture."}
```

Result: Success

Pytest Code:

```
def test_faq_suggestions_no_questions(client):
    lecture_id = 99999  # Use an ID with no questions in Firestore
    endpoint = f"/faq_suggestions/{lecture_id}"

    response = client.get(endpoint)
    data = response.get_json()

    expected_status = 200
    expected_message = "No questions/doubts asked by the students for this
lecture."
    result = "Success" if response.status_code == expected_status and
expected_message in data.get("faq_suggestions", "") else "Failed"

    write_test_doc(
        title="***Case:*** *Lecture ID with no questions in Firestore*",
        endpoint=endpoint,
        method="GET",
        inputs=f"lecture_id: {lecture_id}",
        expected=f"HTTP Status Code: 200 and message: '{expected_message}'",
        actual=f"HTTP Status Code: {response.status_code}\nJSON:
{json.dumps(data)}",
        result=result
    )

    assert response.status_code == 200
    assert expected_message in data.get("faq_suggestions", "")
```

Case: *Internal Server Error due to invalid lecture_id type*

Request Method: GET

Inputs:

```
lecture_id: invalid
```

Expected Output:

```
HTTP Status Code: 500 and error message
```

Actual Output:

```
HTTP Status Code: 404
JSON: null
```

Result: Failed

Pytest Code:

```
def test_faq_suggestions_server_error(client):
    lecture_id = "invalid"  # Forcefully pass a string to trigger error
    endpoint = f"/faq_suggestions/{lecture_id}"

    response = client.get(f"/faq_suggestions/{lecture_id}")
    try:
        data = response.get_json()
    except Exception:
        data = None

    expected_status = 500
    result = "Success" if response.status_code == expected_status else "Failed"

    write_test_doc(
        title="***Case:*** *Internal Server Error due to invalid lecture_id
type*",
        endpoint=endpoint,
        method="GET",
        inputs=f"lecture_id: {lecture_id}",
        expected="HTTP Status Code: 500 and error message",
        actual=f"HTTP Status Code: {response.status_code}\nJSON:
{json.dumps(data)}",
        result=result
    )
```

```
    assert response.status_code == 500
    assert data and "Error" in data if isinstance(data, dict) else True
```
