Endpoint Being Tested: http://127.0.0.1:5000/clarification

Case: *Successful clarification generation*

Request Method: POST

Inputs:

```
{
  "user_id": 125,
  "quest": "What are variables and literals in Python?"
}
```

Expected Output:

```
HTTP Status Code: 200 and JSON with 'response'
```

Actual Output:

```
HTTP Status Code: 200
JSON: {"query": "What are variables and literals in Python?", "response": "In
Python, a variable is a named storage location that holds a value.  The value can
be changed during the program's execution.  A literal, on the other hand, is a
fixed value directly written into the code.  For example, `name = \"Alice\"`
declares a variable named `name` and assigns it the literal string value
\"Alice\".  The variable `name` can later be assigned a different literal value,
such as `\"Bob\"`.  Literals are the actual values stored in variables.  Variables
can store different literal values, and these values can be modified as needed.
Literals are typically used on the right-hand side of an assignment, whereas
variables can be used on either side.  The choice between using a variable or a
literal depends on whether the value is expected to change during the program's
execution. If the value is constant, a literal is appropriate; if the value might
change, a variable is preferred."}
```

Result: Success

Pytest Code:

```
def test_clarification_success(client):
    payload = {
        "user_id": 125,
        "quest": "What are variables and literals in Python?"
    }
    response = client.post("/clarification", json=payload)
```

```
    data = response.get_json()

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

    write_test_doc(
        title="***Case:*** *Successful clarification generation*",
        endpoint="http://127.0.0.1:5000/clarification",
        method="POST",
        inputs=json.dumps(payload, indent=2),
        expected="HTTP Status Code: 200 and JSON with 'response'",
        actual=f"HTTP Status Code: {response.status_code}\nJSON:
{json.dumps(data)}",
        result=result
    )

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

Case: *Missing query in request payload*

Request Method: POST

Inputs:

```
{
  "user_id": 125,
  "quest": ""
}
```

Expected Output:

```
HTTP Status Code: 400 and error message
```

Actual Output:

```
HTTP Status Code: 400
JSON: {"Error": "A query is required"}
```

Result: Success

Pytest Code:

```
def test_missing_query(client):
    payload = {
        "user_id": 125,
        "quest": ""
    }
    response = client.post("/clarification", json=payload)
    data = response.get_json()

    expected_status = 400
    result = "Success" if response.status_code == expected_status and "Error" in
data else "Failed"

    write_test_doc(
        title="***Case:*** *Missing query in request payload*",
        endpoint="http://127.0.0.1:5000/clarification",
        method="POST",
        inputs=json.dumps(payload, indent=2),
        expected="HTTP Status Code: 400 and error message",
        actual=f"HTTP Status Code: {response.status_code}\nJSON:
{json.dumps(data)}",
        result=result
    )

    assert response.status_code == 400
    assert data and "Error" in data
```

Case: *Internal server error during clarification generation*

Request Method: POST

Inputs:

```
{
  "user_id": -999,
  "quest": "This should trigger a server-side failure"
}
```

Expected Output:

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

Actual Output:

```
HTTP Status Code: 200
JSON: {"query": "This should trigger a server-side failure", "response": "I can
```

```
only answer syllabus-related questions. Please ask something relevant to the
syllabus."}
```

Result: Failed

Pytest Code:

```
def test_clarification_server_error(client):
    # This should be an ID or input that forces some kind of internal failure
during agent processing.
    payload = {
        "user_id": -999,  # Invalid or corrupted user session to simulate internal
error
        "quest": "This should trigger a server-side failure"
    }
    response = client.post("/clarification", json=payload)
    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 during clarification
generation*",
        endpoint="http://127.0.0.1:5000/clarification",
        method="POST",
        inputs=json.dumps(payload, indent=2),
        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
```