Endpoint Being Tested: http://127.0.0.1:5000/api/grading/query

Case: *Successful grading document query*

Request Method: POST

Inputs:

```
{
  "query": "What is the grading criteria for software engineering?",
  "k": 1,
  "score_threshold": 0.6
}
```

Expected Output:

```
HTTP Status Code: 200 and JSON with 'answer' and 'documents'
```

Actual Output:


```
HTTP Status Code: 200
JSON: {"answer": "The final grade (T) is calculated as:  T = 0.05GAA + 0.2Qz2 +
0.4F + 0.1GP1 + 0.1GP2 + 0.1PP + 0.05CP, where GAA is the average of the first 10
weekly assignments, Qz2 is the Quiz 2 score, F is the end-term exam score, GP1 and
GP2 are group project milestone scores, PP is the project presentation score, and
CP is the course participation score.  Eligibility requirements include a minimum
average on the first 7 weekly assignments and submission of the group project.",
"documents": [{"content": "2. Software Engineering\n\nQuiz 1: No Quiz 1 Quiz 2:
March 16 2025 End term: April 13 2025\n\nAbove to be attended in person at
designated centres.\n\n\nEligibility to write end term exam:\n\nAverage of the
best 5 out of the first 7 weekly assignment scores >= 40/100 AND submission of
Group project Milestone [1-3]\n\n\nEligibility to get final course grade:
Attending the End term exam AND Submission of group project (All milestones)  is
mandatory for course grade AND score in group project > 0\n\n\nOverall score for
eligible students:\n\nGAA = Average score in First 10 weekly graded
assignments\n\nQz1 = NOT THERE IN THIS COURSE\n\nQz2 = score in Quiz II (0, if not
attempted)\n\nGroup Project- Milestone 1-3 (After week 6) - GP1\n\nGroup project -
Milestone 4-6 (After week 12) - GP2\n\nProject Presentation - PP\n\nCourse
participation activity - CP\n\nF - score in End Term exam\n\nT = 0.05GAA + 0.2Qz2
+ 0.4F+ 0.1GP1 + 0.1GP2 + 0.1PP + 0.05CP\n\n(More details about the Group project
will be given in the course).", "source": "JAN 2025 TERM GRADING DOCUMENT"}]}
```


Result: Success

Pytest Code:

```
def test_query_success(client):
    payload = {
        "query": "What is the grading criteria for software engineering?",
        "k": 1,
        "score_threshold": 0.6
    }

    response = client.post("/api/grading/query", json=payload)
    data = response.get_json()

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

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

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

Case: *Missing required 'query' parameter*

Request Method: POST

Inputs:

```
{
  "k": 1,
  "score_threshold": 0.6
}
```

Expected Output:

```
HTTP Status Code: 400 and error message indicating 'Query is required'
```

Actual Output:

```
HTTP Status Code: 400
JSON: {"error": "Query is required"}
```

Result: Success

Pytest Code:

```
def test_query_missing_param(client):
    payload = {
        # query is missing
        "k": 1,
        "score_threshold": 0.6
    }

    response = client.post("/api/grading/query", 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 required 'query' parameter*",
        endpoint="http://127.0.0.1:5000/api/grading/query",
        method="POST",
        inputs=json.dumps(payload, indent=2),
        expected="HTTP Status Code: 400 and error message indicating 'Query is
required'",
        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 while processing query*

Request Method: POST

Inputs:

```
{
  "query": null
}
```

Expected Output:

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

Actual Output:

```
HTTP Status Code: 500
JSON: {"error": "Error processing query: 'NoneType' object has no attribute
'replace'"}
```

Result: Success

Pytest Code:

```
def test_query_server_error(client):
    payload = {
        "query": None  # Simulating malformed input
    }

    response = client.post("/api/grading/query", 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 while processing query*",
        endpoint="http://127.0.0.1:5000/api/grading/query",
        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
```
