BRD Analysis - 2025-08-08

Okay, here's the Python code to execute the APIs presented in the BRD. This code is designed to simulate sending POST requests to the local host API endpoint. **Important Considerations:** * **Localhost:** This code assumes the API is running on `http://localhost:9080`. Adjust this if your API is running on a different address. * **Dependencies:** You'll need to use the `requests` library. If you don't have it, install it using: `pip install requests` * **File Uploads:** This code *simulates* file uploads. In a real application, you'd need to handle file uploads correctly – using libraries like `multipart/form-data` or a file upload component in your application. This example just constructs the correct URL and data. * **Error Handling:** This code includes basic error handling but should be expanded in a production environment. * **No Actual Execution:** This code *does not* execute the API. It constructs the requests and prints the results, simulating how the API would respond.

```
import requests
import json

# --- Utility Function for Sending Requests ---
def send_request(url, data=None, files=None):
    """Sends a POST request to the given URL with optional data and files."""
    try:
        if files:
            response = requests.post(url, files=files)
        else:
            response = requests.post(url, data=data)  # Send data as JSON
        response.raise_for_status()  # Raise HTTPError for bad responses (4xx or 5xx)
        return response.json()  # Return JSON response
    except requests.exceptions.RequestException as e:
        print(f"Request Error: {e}")
        return None

# --- Example Usage (Simulating API Calls) ---

# 1. Basic Text Query
print("--- Sending Basic Text Query ---")
result = send_request(
    "http://localhost:9080/file",
    data={"content": "What banking services does IDFC offer?"}
)
print(json.dumps(result, indent=4))  # Pretty print the JSON

# 2. PDF Analysis
print("\n--- Sending PDF Analysis ---")
result = send_request(
    "http://localhost:9080/file",
    data={"content": "Analyze this bank statement"},
    files={"pdf": open("statement.pdf", "rb")}  # Assume 'statement.pdf' exists
)
print(json.dumps(result, indent=4))

# 3. RAG Question Answering
```

```
        print("\n--- Sending RAG Question Answering ---")
        result = send_request(
            "http://localhost:9080/file",
            data={"content": "What is the invoice total amount?"},
            files={"rag": open("invoice.pdf", "rb")}  # Assume 'invoice.pdf' exists
        )
        print(json.dumps(result, indent=4))

        # 4. Document Classification
        print("\n--- Sending Document Classification ---")
        result = send_request(
            "http://localhost:9080/file",
            data={"content": "What documents are present in this file?"},
            files={"pdf": open("documents.pdf", "rb")}  # Assume 'documents.pdf' exists
        )
        print(json.dumps(result, indent=4))


        print("\n--- End of Example ---")
```

Explanation: 1. **`send_request()` function:** * Takes the URL, data (as a dictionary), and files (as file objects) as arguments. * Uses `requests.post()` to send a POST request. * If files are provided, the `files` parameter of `requests.post()` is used. * `response.raise_for_status()`: This line is *crucial*. It raises an HTTPError exception if the API returns a 4xx or 5xx status code (indicating an error). This makes your code much more robust. * `response.json()`: Parses the JSON response from the API. * Error handling: Catches `requests.exceptions.RequestException` to handle network problems or other request-related errors. 2. **Example Usage:** * The code then calls `send_request()` for each of the example API calls described in the BRD, demonstrating how to construct the requests and print the JSON responses. **To use this code:** 1. **Install `requests`:** `pip install requests` 2. **Create Dummy PDFs:** Make sure you have `statement.pdf`, `invoice.pdf` and `documents.pdf` files in the same directory as your Python script (or adjust the file paths in the code). You can create simple PDF files with text. 3. **Run the Script:** Execute the Python script. The output will simulate the responses you'd get from the API calls. This revised code provides a much more practical and robust starting point for interacting with your API. Remember to adapt it to your specific needs and environment. The detailed explanations and error handling will help you debug and maintain your code.