```
from typing import Dict, Any, Union

import requests

from loguru import logger



class AlphaVantageClient:
    """
    Client to fetch commodities and economic indicators data from Alpha Vantage API.
    """

    BASE_URL = "https://www.alphavantage.co/query"

    def __init__(self, api_key: str) -> None:
        """
        Initialize the AlphaVantageClient with an API key.

        :param api_key: Your Alpha Vantage API key.
        """
        self.api_key = api_key

    def fetch_data(
        self, function: str, symbol: str = None
    ) -> Union[str, Dict[str, Any]]:
        """
        Fetches data from Alpha Vantage API and returns it as both string and dictionary.
```

```
:param function: Alpha Vantage function type (e.g., 'TIME_SERIES_DAILY', 'REAL_GDP').
:param symbol: Optional. The commodity/economic indicator symbol.
:return: The data as both a string and a dictionary.
"""
params = {
    "apikey": self.api_key,
    "function": function,
    "symbol": symbol,
}

logger.info(
    f"Fetching data for function '{function}' with symbol '{symbol}'"
)

try:
    response = requests.get(self.BASE_URL, params=params)
    response.raise_for_status()
    data = response.json()
    data_as_string = response.text
    logger.success(
        f"Successfully fetched data for {symbol if symbol else function}"
    )
    return data_as_string, data
except requests.RequestException as e:
    logger.error(
        f"Error while fetching data from Alpha Vantage: {e}"
```

```
            )
            return str(e), {}

    def get_commodities_data(
        self,
    ) -> Dict[str, Union[str, Dict[str, Any]]]:
        """
        Fetches data for trending commodities such as Crude Oil, Natural Gas, and others.

        :return: Dictionary with commodity names as keys and a tuple of (string data, dictionary data) as values.
        """
        commodities = {
            "Crude Oil (WTI)": "OIL_WTI",
            "Crude Oil (Brent)": "OIL_BRENT",
            "Natural Gas": "NATURAL_GAS",
            "Copper": "COPPER",
            "Aluminum": "ALUMINUM",
            "Wheat": "WHEAT",
            "Corn": "CORN",
            "Cotton": "COTTON",
            "Sugar": "SUGAR",
            "Coffee": "COFFEE",
            "Global Commodities Index": "COMMODITIES",
        }
```

```
        commodity_data = {}
        for name, symbol in commodities.items():
            data_str, data_dict = self.fetch_data(
                function="TIME_SERIES_DAILY", symbol=symbol
            )
            commodity_data[name] = (data_str, data_dict)

        return commodity_data

    def get_economic_indicators(
        self,
    ) -> Dict[str, Union[str, Dict[str, Any]]]:
        """
        Fetches data for economic indicators such as Real GDP, Unemployment Rate, etc.

        :return: Dictionary with indicator names as keys and a tuple of (string data, dictionary data) as values.
        """
        indicators = {
            "Real GDP": "REAL_GDP",
            "Real GDP per Capita": "REAL_GDP_PER_CAPITA",
            "Treasury Yield": "TREASURY_YIELD",
            "Federal Funds Rate": "FEDERAL_FUNDS_RATE",
            "CPI": "CPI",
            "Inflation": "INFLATION",
            "Retail Sales": "RETAIL_SALES",
```

```
            "Durable Goods Orders": "DURABLE_GOODS",
            "Unemployment Rate": "UNEMPLOYMENT",
            "Nonfarm Payroll": "NONFARM_PAYROLL",
        }

        indicator_data = {}
        for name, function in indicators.items():
            data_str, data_dict = self.fetch_data(function=function)
            indicator_data[name] = (data_str, data_dict)

        return indicator_data


if __name__ == "__main__":
    # Replace with your actual API key
    API_KEY = "your_alpha_vantage_api_key"

    av_client = AlphaVantageClient(api_key=API_KEY)

    logger.info("Fetching commodities data...")
    commodities_data = av_client.get_commodities_data()

    logger.info("Fetching economic indicators data...")
    economic_indicators_data = av_client.get_economic_indicators()

    # Example of accessing the data
```

```
for name, (data_str, data_dict) in commodities_data.items():
    logger.info(
        f"{name}: {data_str}..."
    )  # Truncate the string for display


for name, (
    data_str,
    data_dict,
) in economic_indicators_data.items():
    logger.info(
        f"{name}: {data_str}..."
    )  # Truncate the string for display
```