Pseudo Code on how to safely connect the API’s

Instructions:

1. Bring in Tools
    - Use requests to talk to APIs, os to get secret keys, time to track time, and logging to keep track of what’s happening.
2. Set Up Logging
    - Turn on logging so we can see info and errors when the program runs.
3. Make a Rate Limiter
    - Create a tool to control how often we call the APIs so we don’t get blocked.
    - Add a timer to check if we can call again or if we need to wait.
4. Set Up API Details
    - Get secret API keys from the computer (don’t hardcode them).
    - Make a setup for each API:
        - Add the base URL (where the API lives).
        - Add the secret key for logging in.
        - Add limits on how often we can call.
5. Fetch Data Safely
    - Write a function to:
        - Check if it’s okay to call the API.
        - Wait if needed (so we don’t break the rules).
        - Get data from the API.
        - Handle errors if something goes wrong.
6. Make API Helper Functions
    - Write little functions for each API to do specific jobs, like:
        - Get transactions from Plaid.
        - Get exchange rates.
        - Get budgets.
7. Start the Program
    - Set up the APIs.
    - Call the helper functions to get data.
    - Log messages to confirm everything worked.
8. Run the Program
    - Add a part at the end to make sure the program runs when we open it.

```python
# Import necessary libraries
import requests
import os
import time
import logging

# Initialize logging
logging.basicConfig(level=logging.INFO)

# Define rate limiter class
class RateLimiter:
    def __init__(self, max_requests, interval):
        self.max_requests = max_requests
        self.interval = interval
        self.requests = []

    def allow_request(self):
        current_time = time.time()
        # Remove outdated requests
        self.requests = [req for req in self.requests if current_time - req < self.interval]
        if len(self.requests) < self.max_requests:
            self.requests.append(current_time)
            return True
        return False

    def retry_after(self):
        return max(0, self.interval - (time.time() - self.requests[0]))

# Initialize API configurations
def initialize_apis():
    API_KEYS = {
        "PLAID": os.getenv("PLAID_KEY"),
        "OPEN_EXCHANGE": os.getenv("OPEN_EXCHANGE_KEY"),
        "ALPACA": os.getenv("ALPACA_KEY"),
        "FIREFLY": os.getenv("FIREFLY_KEY"),
    }

    PLAID_CONFIG = {
        "BASE_URL": "https://api.plaid.com",
        "HEADERS": {"Authorization": f"Bearer {API_KEYS['PLAID']}"},
        "RATE_LIMIT": RateLimiter(100, 60),  # Example: 100 requests per minute
    }

    OPEN_EXCHANGE_CONFIG = {
        "BASE_URL": "https://openexchangerates.org/api",
        "HEADERS": {"Authorization": f"Bearer {API_KEYS['OPEN_EXCHANGE']}"},
        "RATE_LIMIT": RateLimiter(1, 3600),  # 1 request per hour
    }

    ALPACA_CONFIG = {
        "BASE_URL": "https://paper-api.alpaca.markets",
        "HEADERS": {"Authorization": f"Bearer {API_KEYS['ALPACA']}"},
        "RATE_LIMIT": RateLimiter(1000, 60),  # Example limit
    }

    FIREFLY_CONFIG = {
        "BASE_URL": "http://your-firefly-instance/api/v1",
        "HEADERS": {"Authorization": f"Bearer {API_KEYS['FIREFLY']}"},
        "RATE_LIMIT": RateLimiter(500, 60),  # Adjust as needed
    }

    return PLAID_CONFIG, OPEN_EXCHANGE_CONFIG, ALPACA_CONFIG,
FIREFLY_CONFIG

# Fetch data safely from API
def fetch_data(config, endpoint, params=None):
    if not config["RATE_LIMIT"].allow_request():
        logging.warning("Rate limit exceeded. Retrying later.")
        time.sleep(config["RATE_LIMIT"].retry_after())
        return fetch_data(config, endpoint, params)

    try:
        response = requests.get(f"{config['BASE_URL']}{endpoint}",
headers=config["HEADERS"], params=params)
        response.raise_for_status()  # Raise exception for HTTP errors
        return response.json()
    except requests.exceptions.RequestException as e:
        logging.error(f"Error fetching data: {e}")
        return None

# Example API-specific functions
```

```
def get_plaid_transactions(plaid_config):
    return fetch_data(plaid_config, "/transactions/get")

def get_live_exchange_rates(open_exchange_config):
    return fetch_data(open_exchange_config, "/latest.json")

def get_firefly_budgets(firefly_config):
    return fetch_data(firefly_config, "/budgets")

# Main function
def main():
    PLAID_CONFIG, OPEN_EXCHANGE_CONFIG, ALPACA_CONFIG,
FIREFLY_CONFIG = initialize_apis()

    # Example usage
    transactions = get_plaid_transactions(PLAID_CONFIG)
    if transactions:
        logging.info("Fetched Plaid transactions.")

    live_rates = get_live_exchange_rates(OPEN_EXCHANGE_CONFIG)
    if live_rates:
        logging.info("Fetched live exchange rates.")

    budgets = get_firefly_budgets(FIREFLY_CONFIG)
    if budgets:
        logging.info("Fetched Firefly budgets.")

# Run the app
if __name__ == "__main__":
    main()
```