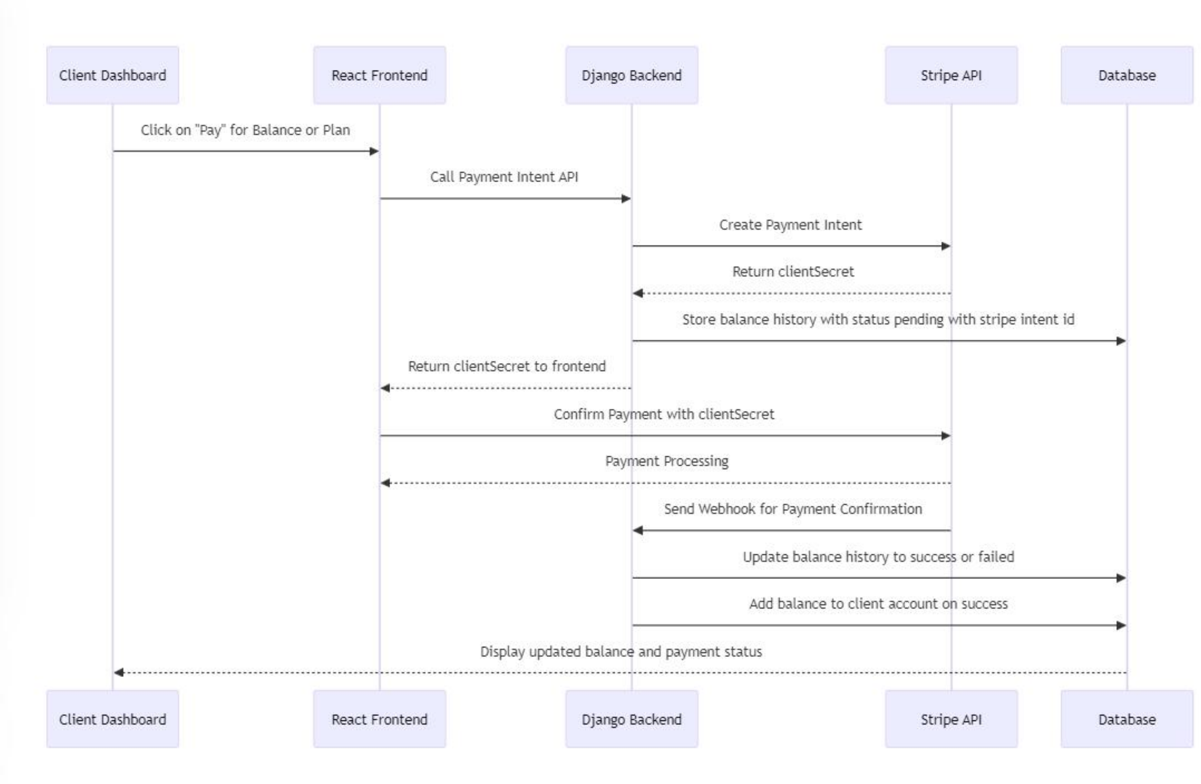
Stripe Payment Integration Workflow Documentation

Workflow Description:

1. **Client Dashboard Interaction:**
    - The user logs into the dashboard and selects a pricing plan or adds balance.
    - Once the user clicks "Pay for Add Balance" or "Purchase Plan," the frontend (React) calls the **Stripe Payment Intent API** (Django backend).
2. **Backend Process (API Call):**
    - The Django backend creates a **Payment Intent** through the **Stripe API** with the selected amount.
    - It also stores the initial transaction data in the **BalanceHistory** table with a **pending** status.
3. **Client Frontend:**
    - The frontend receives the *clientSecret* from the API and confirms the payment using **Stripe.js**.
    - The user provides card details, and the payment is processed.
4. **Stripe Webhook Confirmation:**
    - Stripe sends a webhook to the Django backend to confirm whether the payment succeeded or failed.
    - On success, the backend updates the **BalanceHistory** record to **success** and adds the credited amount to the user's balance.


Client Dashboard
React Frontend
Django Backend
Stripe API
Database
Click on "Pay" for Balance or Plan
Call Payment Intent API
Create Payment Intent
Return clientSecret
Store balance history with status pending with stripe intent id
Return clientSecret to frontend
Confirm Payment with clientSecret
Payment Processing
Send Webhook for Payment Confirmation
Update balance history to success or failed
Add balance to client account on success
Display updated balance and payment status
Client Dashboard
React Frontend
Django Backend
Stripe API
Database

Detailed Workflow Steps:

1. **Client Dashboard:**
    - Users click "Pay" for either purchasing a plan or adding balance to their account.
    - The frontend sends a request to the backend Django API to create a Stripe Payment Intent.
2. **Django Backend:**
    - Django backend creates a **Payment Intent** using the Stripe API, sending the payment amount and currency.
    - It stores a pending record in the **BalanceHistory** table.
    - The backend returns the **clientSecret** to the frontend for confirming the payment.
3. **React Frontend:**
    - The frontend uses Stripe.js to process the payment with the **clientSecret** returned by the Django API.
    - Once the user submits their card information, the frontend confirms the payment with Stripe.
4. **Stripe Webhook:**
    - Upon successful or failed payment, Stripe sends a webhook notification to the backend Django server.
    - The Django backend updates the **BalanceHistory** table to reflect the final status of the payment.
    - If the payment was successful, the user's balance is updated accordingly.
5. **Final Update:**
    - The user’s dashboard is updated to show the new balance or subscription plan.
    - The status of the payment (success or failure) is displayed to the user.

This flow ensures that the payment process is secure, using Stripe for handling sensitive information while maintaining a record of all payment activities within your system.