End-to-End Payment Flow (Connect Research)

1. Client Payment Flow (Frontend + Backend)

Step 1: Project Setup

- Client posts a project.
- Estimated project fee is determined (consultant fee + platform fee + tax).

Step 2: Checkout Page

- Built with **Stripe Elements** (embedded form).
- User sees:
    - **Consultant fee** (goes to consultant).
    - **Platform fee** (goes to Connect Research).
    - **Taxes** (calculated via Stripe Tax based on client's location).
    - **Promo code input** (applies discount **only to platform fee**).
- Total amount = `(Consultant fee + Platform fee – Promo discount) + Taxes`

Step 3: Payment Confirmation

- Payment goes to **Connect Research Stripe account (platform)**.
- Funds are held in **escrow** (via Stripe Connect Custom/Express account).
- Stripe webhook listens for `payment_intent.succeeded` or `payment_intent.payment_failed`.

2. Escrow Logic

After Payment:

- Funds are **not immediately transferred** to the consultant.
- Stored as **available balance** in platform account, or in **separate balance** per Stripe Connect architecture.

Work Submitted by Consultant

- Consultant clicks “Submit Work.”
- Timestamp is recorded.
- System notifies the client and sets a **5-day timer**.

Client Options

- Client can:
    - Click “Mark as Delivered” → Funds released to consultant.
    - Click “Raise Dispute” → Funds held.
- “Mark as Delivered” should Trigger fund **release** via Stripe Transfer or PaymentIntent to **consultant’s connected account**.
- Notify both parties.

Client Does Not Confirm in X Days (e.g., 5):

- Auto-release funds to consultant via backend logic.

If Dispute Is Raised Before Release:

- Funds are **held**.
- No release until admin resolves.
- Resolution:
    - Partial or full refund to client.
    - Release to consultant.
    - Split.

3. Dispute Workflow

Trigger:

- Client clicks “Dispute” before delivery confirmation.
- System flags transaction as **"Under Dispute"**.

Actions:

- Notifications sent to:
    - Client (acknowledgment)
    - Consultant (alert)
    - Admin (with project details)

Admin Interface:

- Review evidence.
- Choose to:
    - Refund client.
    - Release full/partial funds to consultant.

Stripe Webhooks Monitored:

- `charge.dispute.created`
- `charge.dispute.closed`

4. Payout Flow (Consultant)

- Once funds are cleared (via delivery confirmation or dispute resolution), initiate:
    - **Stripe Transfer** to the **consultant's connected account**.
- Stripe webhook: `transfer.paid` (for confirmation)

5. Promo Code Management

- Applies to **platform fee only**.
- Built into frontend and backend logic.
- Admin can:
    - Create/edit/delete codes via admin dashboard.
    - Set: code, % or flat amount, expiration date, usage limit.

- Promo applied **before taxes**.

6. Tax Management

- Use **Stripe Tax**:
    - Enable for consulting services.
    - Auto-detect location from billing info.
    - Include tax on:
        - Consultant fee
        - Platform fee (optional depending on structure).
- Displayed clearly during checkout.
- Remittance: Enable Stripe's automatic tax filing or export via dashboard for manual remittance.

7. Email Notifications (Dynamic)

Clients receive:

- Payment confirmation
- Project delivery confirmation
- Dispute acknowledgment

Consultants receive:

- Payout confirmation
- Dispute raised alert

Admins receive:

- Dispute raised
- Payment issue (e.g., failed webhook, refund needed)

Dynamic Email Content:

- Project ID & name

- Amount paid/held
- Dates
- Dispute reason (if applicable)
- Status updates

Can be built using:

- SendGrid
- MailerSend
- SMTP from backend

8. Admin Dashboard Features

- Create/Edit/Delete Promo Codes.
- View:
    - Payment logs
    - Platform fee breakdowns
    - Consultant payouts
    - Tax data (pull via Stripe API)
- Manually resolve disputes
- Release or hold payments
- View project + payment timelines

Stripe Webhooks to Configure

Webhook Event	Purpose
`payment_intent.succeeded`	Payment successful
`payment_intent.payment_failed`	Payment failed
`transfer.paid`	Consultant received payout

`charge.dispute.created`	Dispute raised
`charge.dispute.closed`	Dispute resolved
`checkout.session.completed`	Optional, for Stripe Checkout flows