Technical Proof: NRDC Bank Integration Readiness

If the bank's technical team asks to see "what is already implemented," you can refer to the following files and logic. This proves the platform is architecturally ready for their automated reconciliation API.

---

1. Donation Initiation & Reference Generation

File: `app/api/payments/bank-transfer/route.ts`

When a donor selects "Bank Transfer", we already execute the following:

- **Unique Reference**: We generate a collision-resistant ID (e.g., `NRDC-BT-1768208...`).
- **Database Entry**: We create a `pending` record in the `Donation` table immediately.
- **Reference Tracking**: The `bankTransferReference` field is populated to ensure we have a unique key to match against the bank's narration.

---

2. Donor Flow & Instructions

File: `app/[locale]/donate/bank-transfer/page.tsx`

This page is the "Completion Screen" donors see. It is already built to:

- **Display Reference**: Shows the donor exactly what reference to use in their bank narration.
- **Copy Utility**: A one-click button for donors to copy the reference accurately.
- **Email Proof**: A `mailto` link that pre-fills the subject with the reference for manual verification (until automation is live).

---

3. Database Schema (Prisma)

File: `prisma/schema.prisma`

Our `Donation` model is already optimized for bank reconciliation with these fields:

- `reference`: Unique tracking ID.
- `bankTransferReference`: Explicit field for matching bank statements.
- `paymentStatus`: Defaulting to `pending`, ready to be toggled to `completed` via API.
- `metadata`: Flexible JSON field to store bank-specific response data (Transaction IDs, etc.).

---

4. Webhook Readiness (Proof of Concept)

File Reference: `app/api/payments/mchanga/verify/route.ts`

If they ask if we can handle automated status updates today:

- **Evidence**: Show them our M-Changa verification route.
- **Talking Point**: *"We already have a standardized verification pattern used for other gateways (like M-Changa). We simply need your API schema to implement a similar `bank-transfer/callback` route that updates our `Donation` records in real-time."*

---

5. Technical Stack Summary

- **Framework**: Next.js 15+ (App Router)
- **Database**: SQLite (Development) / PostgreSQL (Production) via Prisma ORM
- **Deployment**: Vercel (Ready for HTTPS/TLS 1.2 callbacks)

- **Security**: Ready to implement HMAC validation or OAuth2 headers for the callback endpoint.