Payment Link System – Part 1: Product & Technical Strategy

1. Product Strategy

Goal

Enable merchants to charge customers in USD via payment links while delivering MXN to recipients in Mexico, with a reliable checkout UX, transparent fees, and predictable FX outcomes.

Why own the checkout UX instead of PSP redirects?

- Consistent cross-PSP experience: A hosted, product-owned checkout lets us present the same look & feel regardless of whether we route to Stripe or Adyen. This is critical when we're orchestrating multiple PSPs behind the scenes.
- Control over conversion & experimentation: Owning the frontend lets us A/B test copy, form layout, error handling, and incentives (e.g., "first payment free") without depending on PSP release cycles or hosted page constraints.
- Better fee & FX transparency: We can show a clear breakdown (amount in USD, fees, FX rate, final MXN) in real time, which is hard to do if we depend on opaque PSP-hosted flows.

Why use two PSPs instead of one?

- Reliability & redundancy: If one PSP is degraded (regional outage, risk filters, routing issues), we can fail over to the other and keep the payment link usable.
- Optimized authorization rates: Different PSPs have different strengths by card network, country, or issuing bank. Routing logic lets us optimize for higher approval rates over time.
- Pricing & negotiation leverage: With traffic going to more than one PSP, we have better leverage on fees and can optimize cost per transaction.

Biggest product risk

The biggest product risk is complexity vs trust for merchants and payers:

- We're mixing FX, multiple PSPs, and flexible fee models (fixed, variable, embedded in FX, incentives).
- If the UX is not crystal clear, users may feel the pricing is opaque or "hidden-fee-ish," hurting adoption and repeat usage.

Mitigation: prioritize a simple, opinionated initial fee model, very clear explanations in the checkout UI and receipts, and good reporting for merchants before adding too much flexibility.

2. Technical Strategy

Biggest technical risk

The biggest technical risk is correct, consistent orchestration between PSPs and FX/fee calculation under failure conditions:

- We must keep fees and FX consistent across:
    - What we show on the checkout,
    - What we send to the PSP,
    - What we record internally and settle in MXN.
- Multi-PSP orchestration introduces complex failure modes:
    - Partial charges, timeouts, duplicated attempts across PSPs,
    - Webhook races (event arrived twice or late from different PSPs),
    - Idempotency and reconciliation across systems.

A bug here affects money movement and trust, not just UI.

MVP: what's in vs out

Must-have MVP

- **Core payment link lifecycle**
    - Create payment link with basic configuration (amount in USD, target MXN account, simple fee profile).
    - Retrieve payment link and show real-time fee + FX breakdown.
- **Single primary PSP + minimal failover**
    - Start with one PSP as primary, with a **very simple failover** strategy (e.g., switch to secondary only on explicit simulated failures).
- **Basic FX + fee engine**
    - Fixed fee + simple percentage fee, plus a single FX rate with a configurable markup.
    - Rates cached for short periods; consistent rate per transaction.
- **Tokenization & charge**
    - Simulated SDK in frontend → card token → backend → PSP mock for charge.
- **Webhook handling**
    - Minimal webhook flow to update transaction status (succeeded/failed).
- **Simple merchant view**
    - Basic API or simple UI page to see list of transactions and final MXN amounts.
- **Testing**
    - Unit tests for fee/FX calculation.
    - Integration tests for core payment success + simple failover scenario.

Cut from MVP

- Complex, fully generic fee configurations (tiered pricing, per-merchant custom rules engine).

- Advanced incentive logic (e.g., “first 3 transactions at 50% discount” with complex rules); start with something like “no fee for first N transactions” hard-coded per merchant.
- Rich reporting dashboards and advanced filters.
- Sophisticated PSP routing (by BIN, card brand, country). Start with simple primary→secondary logic.
- Production-grade observability and alerting stack; in MVP, keep it to logs + minimal metrics.

3. MVP Scope & Prioritization

P0

- Hosted checkout UI owned by us (card form + breakdown of fees/FX).
- Payment link CRUD API (create, get, basic list).
- FX + fee calculator for USD→MXN (fixed + percentage + simple FX markup).
- Integration with primary PSP mock (tokenization + charge + webhooks).
- Minimal secondary PSP mock and a basic failover path.
- Transaction store with final USD charged, FX rate applied, MXN delivered, and fees.
- Unit tests for fee/FX engine and integration tests for happy path + failover.

P1 – After MVP

- More advanced fee configurations and incentive structures.
- Smarter PSP routing strategies driven by metrics.
- Better merchant dashboard & exports.
- More robust observability (dashboards, alerts) and operational tooling.