**Product Requirements Document (PRD)**

**PayMatch – Multi-tenant Payments Rails Aggregator for SMEs in East Africa**

**1. Introduction & Executive Summary**

**PayMatch** is a multi-tenant SaaS **payments rails aggregator**, designed to give SMEs in East Africa a single API for processing both **incoming (C2B collections, STK push, bank transfers)** and **outgoing (B2C disbursements, supplier/vendor payouts)** payments.

Unlike existing PSPs that focus only on collections, PayMatch positions itself as a **neutral, API-first middleware** that abstracts multiple providers (M-Pesa Daraja, Pesapal, iPay, bank APIs) under one developer-friendly layer.

* **Problem**: SMEs are forced to integrate separately with multiple payment providers (mobile money, PSPs, banks). This creates complexity, increases costs, and slows down innovation. Manual reconciliation adds another layer of inefficiency.
* **Solution**: PayMatch provides a **single, unified API** for collections and payouts, tenant-isolated transaction storage, automated reconciliation, and real-time insights. Businesses can plug in once, and PayMatch handles routing, security, and compliance.
* **Positioning**: A **payments rails aggregator** for East Africa. Start with M-Pesa, expand to banks and regional wallets. Reconciliation is a **value-added service**, not the sole product.

**Target Audience**: SMEs, fintech startups, and platforms (e-commerce, lending, ERP/PoS systems) that need to manage both incoming and outgoing payments at scale.

**2. Goals & Objectives**

* **MVP**: Launch with inbound (C2B, STK Push) + outbound (B2C disbursement) on M-Pesa.
* **Expansion**: Add bank APIs (KCB, Equity, Co-op) and PSPs (Pesapal, iPay) within 12 months.
* **Adoption**: Onboard 10 SMEs/fintechs with live transactions in 6 months.
* **Reliability**: 99.9% uptime for transaction processing.
* **Efficiency**: Reconciliation reduces SME workload by 90%.
* **Revenue**: SaaS + transaction-based pricing.

**3. User Stories (MVP)**

* **SME Owner**: “I want one system to collect from customers and pay out to suppliers.”
* **Developer**: “I want a unified API for collections and payouts, instead of integrating with multiple PSPs.”
* **Finance Manager**: “I want daily reconciled reports of both inflows and outflows.”
* **Customer Support**: “I want to confirm in seconds if a customer payment was received or a supplier was paid.”
* **Platform Operator**: “I want to control API keys, rate limits, and provider credentials for multiple tenants.”

**4. Product Features (MVP)**

**4.1 Core API (multi-rail)**

* **Inbound (Collections)**:
  + C2B payments (M-Pesa Paybill/Till, Pesapal, iPay).
  + STK Push requests (customer-initiated).
* **Outbound (Disbursements)**:
  + B2C M-Pesa payouts (vendors, refunds, commissions).
  + Bank transfers (planned in Phase 2).
* **Webhook Listener**: Unified ingestion for multi-rail events.
* **Reconciliation Engine**: Automatic matching of inflows/outflows with business records.

**4.2 Web Dashboard**

* **Transactions**: Unified ledger of collections & payouts.
* **Search & Filters**: By invoice, M-Pesa code, MSISDN, vendor, payout batch.
* **Analytics**: Cash in vs. cash out, daily/weekly/monthly net flows.
* **Credential Management**: Manage PSP & bank credentials, webhook tokens.

**4.3 Developer Experience**

* **Unified REST API**: /v1/payments/stk, /v1/payments/c2b, /v1/payouts/b2c, /hooks/{provider}.
* **Interactive Documentation**: Swagger/OpenAPI, sample SDKs (Node.js, PHP, Python).
* **Multi-tenant Security**: API key auth with per-tenant rate limits, encrypted credentials.

**5. Technical Architecture**

* **Core Platform**: Go 1.22 multi-tenant skeleton (already bootstrapped).
* **Database**: PostgreSQL with tenant-aware schemas; Redis for rate limiting and job queues.
* **Event Processing**: Idempotent event storage (payment\_events) → async reconciliation worker.
* **Security**: AES-256 for provider credentials; hashed API keys; HTTPS.
* **Extensibility**: Provider abstraction layer allows plugging new rails (banks, wallets).

**6. Non-Functional Requirements**

* **Scalability**: Handle thousands of tenants, millions of transactions/month.
* **Reliability**: Retry & backoff for failed transactions.
* **Performance**: API latency ≤500ms, real-time ingestion of PSP events.
* **Observability**: Structured logs, health endpoints, usage counters per tenant.

**7. Out of Scope (MVP)**

* Advanced orchestration (split settlements, multi-rail routing).
* Complex workflow automation (escrow, lending flows).
* Native ERP/Accounting integrations (QuickBooks, Zoho).
* Multi-currency (initially KES only).

**8. Success Metrics**

* **Coverage**: # of inbound & outbound rails supported.
* **Adoption**: # of tenants actively processing both collections & payouts.
* **Throughput**: Avg. transactions per tenant per day/week.
* **Accuracy**: >95% reconciliation success rate.
* **Revenue**: Transaction-based + subscription MRR.