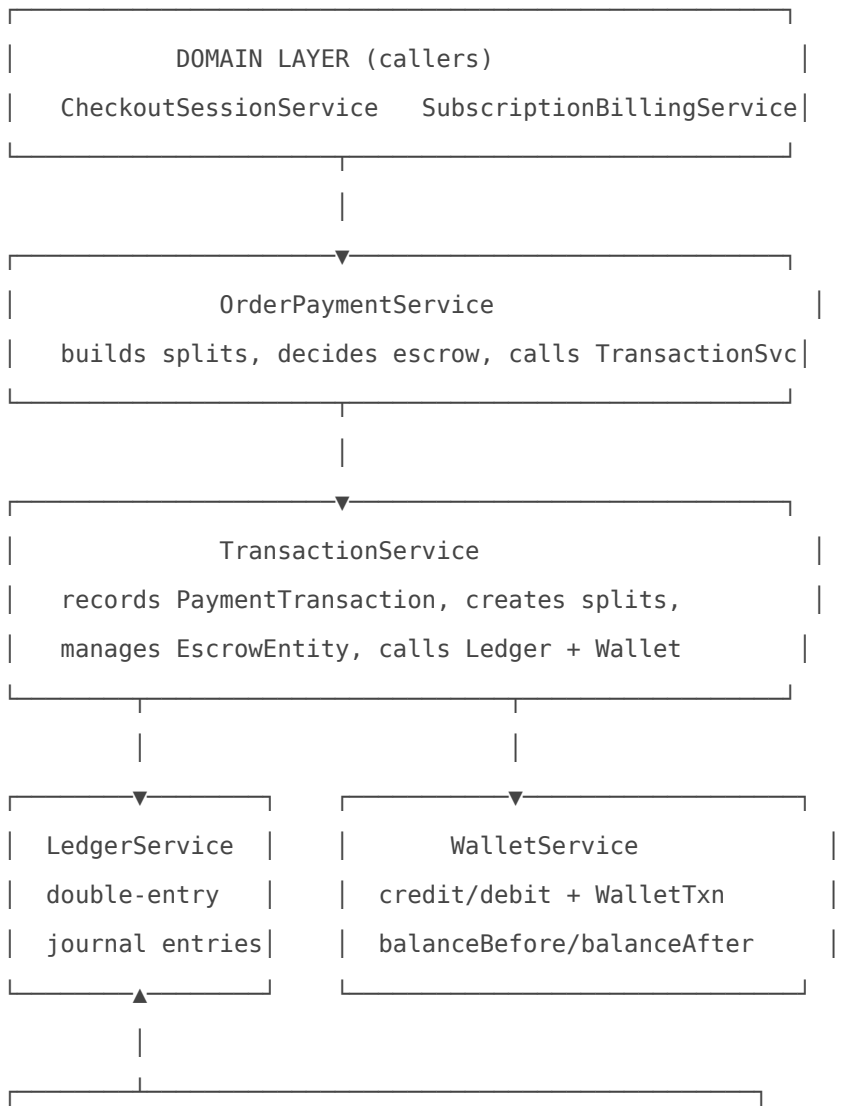
JikoXpress Pro — Financial Engine Flow

> QBIT SPARK CO LIMITED | April 2026 | Internal Architecture Reference

---

1. Architecture Overview

The financial engine is a layered stack. Each layer only talks to the layer directly below it. No layer skips levels.



```
│              PSP Layer                              │
│  PspGatewayResolver → SnippeGateway                 │
│  PspCollectionService   DisbursementService         │
│  SnippeWebhookService                               │
└─────────────────────────────────────────────────────┘
```

---

2. The Two Financial Worlds

World	Channels	Treasury Involved	Platform Earns
World 1 — JikoXpress Pool	App, WhatsApp	Yes — full ledger	Commission + delivery margin
World 2 — Kitchen's Own	POS cash, kitchen custom, kitchen wallet	No — zero treasury	Zero per order (covered by subscription)

Rule: If money never touches Snippe, it never touches the ledger.

---

3. Chart of Accounts

```
ASSETS  — what platform physically controls
  ASSET_PSP_SNIPPE              money sitting at Snippe
  ASSET_ESCROW                  held money, belongs to nobody yet

LIABILITIES  — what platform owes, never touch without permission
  LIABILITY_WALLETS             all user wallets combined
  LIABILITY_SETTLEMENTS         payouts in flight, on the way out

REVENUE  — platform's own earnings
  REVENUE_SUBSCRIPTION_FEES         kitchen plan payments
  REVENUE_MARKETPLACE_COMMISSION    10% on App/WhatsApp orders
  REVENUE_DELIVERY_MARGIN           30% of delivery fee
  REVENUE_PROCESSING_MARGIN         0.5% on PSP transactions

EXPENSES
  EXPENSE_REFUNDS               money returned to customers

EQUITY
```

```
  EQUITY_CAPITAL               admin investments
  EQUITY_RETAINED_EARNINGS     accumulated profits — admin withdraws from here only
```

Golden Rule (checked continuously):

```
ASSET_PSP_SNIPPE + ASSET_ESCROW >= LIABILITY_WALLETS + LIABILITY_SETTLEMENTS
```

If this breaks → crisis alert immediately.

---

4. Money Flow — All Scenarios

4.1 Wallet Topup (Customer adds money)

Trigger: Customer initiates topup via mobile money

```
Customer phone
    │
    ▼
[SnippeGateway.initiateCollection()]
    │  POST /v1/payments  →  Snippe
    │
    ▼
PspCollectionEntity created (status: PROCESSING)
    │
    │  ... customer enters PIN ...
    │
    ▼
Snippe fires webhook: payment.completed
    │
    ▼
[SnippeWebhookController]
    │  verifies HMAC signature
    │  idempotency guard via PspCollectionLogEntity
    │
    ▼
[PspCollectionService.onPaymentCompleted()]
    │
```

```
    ├── LedgerService.writeEntry()
    │     DEBIT  ASSET_PSP_SNIPPE     50,000
    │     CREDIT LIABILITY_WALLETS    50,000
    │
    ├── WalletService.credit()
    │     wallet.balance += 50,000
    │     WalletTransactionEntity (type: TOPUP, balanceBefore, balanceAfter)
    │
    └── PspCollectionEntity (status: COMPLETED)
```

End state: Money at Snippe. Customer wallet credited. Ledger balanced.

4.2 App Order — Delivery — Mobile Money (Full escrow flow)

Order total: TZS 18,000

- Menu + packaging: 13,000 → kitchen
- Delivery fee: 4,000 (rider 2,800 + platform 1,200)
- Commission: 1,000 → platform

```
Customer pays via USSD
    │
    ▼
[PspCollectionService.initiateCollection()]
    │  PspCollectionEntity (status: PROCESSING)
    │
    ▼
Snippe webhook: payment.completed
    │
    ▼
[PspCollectionService.onPaymentCompleted()]
    │  LedgerService.writeEntry()
    │    DEBIT  ASSET_PSP_SNIPPE    18,000
    │    CREDIT LIABILITY_WALLETS   18,000   ← temporary, will move to escrow
    │
    ▼  (collection type = ORDER_PAYMENT, wallet NOT credited here)
    │
    ▼
```

```
[OrderPaymentService.pay()]
    │  builds splits:
    │    KITCHEN_REVENUE  → kitchen     13,000  (PENDING)
    │    DELIVERY_FEE     → rider        2,800  (PENDING)
    │    DELIVERY_FEE     → platform     1,200  (PENDING)
    │    SERVICE_FEE      → platform     1,000  (PENDING)
    │
    ▼
[TransactionService.record()]
    │
    ├── LedgerService.writeEntry()
    │     DEBIT  ASSET_PSP_SNIPPE    18,000
    │     CREDIT ASSET_ESCROW        18,000   ← money held
    │
    ├── PaymentTransactionEntity (holdInEscrow: true)
    ├── TransactionSplitEntity × 4  (status: PENDING)
    └── EscrowEntity (status: HELD, condition: DELIVERY_CONFIRMED)

    ... kitchen prepares, rider picks up, delivers ...

    ▼
Rider confirms delivery
    │
    ▼
[OrderPaymentService.confirmDelivery()]
    │
    ▼
[TransactionService.releaseEscrow()]
    │
    ├── LedgerService.writeEntry()
    │     DEBIT  ASSET_ESCROW                   18,000
    │     CREDIT LIABILITY_WALLETS              13,000  (kitchen)
    │     CREDIT LIABILITY_WALLETS               2,800  (rider)
    │     CREDIT REVENUE_DELIVERY_MARGIN         1,200
    │     CREDIT REVENUE_MARKETPLACE_COMMISSION   1,000
    │
    ├── WalletService.credit(kitchenId, 13,000, ORDER_EARNING)
    ├── WalletService.credit(riderId,    2,800, DELIVERY_EARNING)
    │
    ├── TransactionSplitEntity × 4  (status: CREDITED)
```

```
    └── EscrowEntity (status: RELEASED)
```

End state: Kitchen wallet +13,000. Rider wallet +2,800. Platform revenue +2,200. Ledger balanced.

4.3 App Order — Dine-in — Mobile Money (Immediate split, no escrow)

Order total: TZS 11,000

- Kitchen: 10,000
- Commission: 1,000

```
Customer pays via USSD
    │
    ▼
Snippe webhook: payment.completed
    │
    ▼
[TransactionService.record()]  holdInEscrow: false
    │
    ├── LedgerService.writeEntry()
    │     DEBIT  ASSET_PSP_SNIPPE                11,000
    │     CREDIT LIABILITY_WALLETS               10,000  (kitchen)
    │     CREDIT REVENUE_MARKETPLACE_COMMISSION   1,000
    │
    ├── WalletService.credit(kitchenId, 10,000, ORDER_EARNING)
    │
    ├── TransactionSplitEntity × 2  (status: CREDITED immediately)
    └── No EscrowEntity created
```

End state: Money flows straight through. No holding. Kitchen credited immediately.

4.4 App Order — Pickup — Platform Wallet (Internal payment)

Order total: TZS 12,000

```
Customer pays from JikoXpress wallet
    │  No PSP API call
    │
    ▼
[TransactionService.record()]  channel: PLATFORM_WALLET
    │
    ├── WalletService.debit(customerId, 12,000, ORDER_PAYMENT)
    │
    ├── LedgerService.writeEntry()
    │     DEBIT  LIABILITY_WALLETS    12,000   ← customer wallet reduced
    │     CREDIT ASSET_ESCROW         12,000   ← held for pickup
    │
    ├── EscrowEntity (status: HELD, condition: PICKUP_CODE_CONFIRMED)
    └── TransactionSplitEntity × N  (status: PENDING)

    ... customer arrives, shows pickup code ...

    ▼
[OrderPaymentService.confirmPickup()]
    │
    ▼
[TransactionService.releaseEscrow()]
    │
    └── Same release flow as delivery — splits credited, escrow released
```

End state: Pure internal redistribution. No money entered or left the Snippe pool.

4.5 POS Cash Order (World 2 — Counter order)

```
Customer pays cash at counter
    │
    ▼
[TransactionService.record()]  channel: CASH
    │
    ├── writeCollectionJournal() → returns null  ← no journal entry
    ├── PaymentTransactionEntity created (for reporting only)
    └── No splits, no escrow, no ledger movement
```

End state: Order recorded. Money physically in kitchen till. Treasury untouched. Platform earns zero per order.

4.6 Order Cancellation — Money in Escrow — PSP Payment

```
Order cancelled
    │
    ▼
[OrderPaymentService.cancelOrder()]
    │
    ▼
[TransactionService.refundEscrow()]
    │
    ├── LedgerService.writeEntry()
    │     DEBIT  ASSET_ESCROW        18,000
    │     CREDIT ASSET_PSP_SNIPPE    18,000   ← back to Snippe for external refund
    │
    ├── TransactionSplitEntity × N  (status: REVERSED)
    ├── EscrowEntity (status: REFUNDED)
    └── PaymentTransactionEntity (status: REFUNDED)


    External refund to customer Mpesa handled by PSP layer separately
```

4.7 Order Cancellation — Wallet Payment

```
Order cancelled, paid via wallet
    │
    ▼
[TransactionService.refundEscrow()]
    │
    ├── LedgerService.writeEntry()
    │     DEBIT  ASSET_ESCROW         12,000
    │     CREDIT LIABILITY_WALLETS    12,000   ← straight back to customer wallet
    │
    ├── WalletService.credit(customerId, 12,000, REFUND)
    └── Instant. No PSP call. No waiting.
```

4.8 Subscription Payment — Wallet

```
Kitchen pays monthly subscription from wallet
    │
    ▼
[SubscriptionBillingService.chargeViaWallet()]
    │
    ├── WalletService.debit(accountId, 15,000, SUBSCRIPTION_PAYMENT)
    │
    └── LedgerService.writeEntry()
          DEBIT  LIABILITY_WALLETS        15,000
          CREDIT REVENUE_SUBSCRIPTION_FEES 15,000
```

4.9 Subscription Payment — Mobile Money

```
Kitchen pays via Mpesa
    │
    ▼
Snippe webhook: payment.completed
    │
    ▼
[SubscriptionBillingService.onPaymentCompleted()]
    │
    └── LedgerService.writeEntry()
          DEBIT  ASSET_PSP_SNIPPE          15,000
          CREDIT REVENUE_SUBSCRIPTION_FEES 15,000
```

4.10 User Withdrawal (Customer, Kitchen Owner, or Rider)

```
User requests withdrawal of TZS 30,000
    │
    ▼
[DisbursementService.initiate()]
    │
```

```
    ├── hasSufficientBalance() check
    │
    ├── WalletService.debit(accountId, 30,000, WITHDRAWAL)
    │
    ├── LedgerService.writeEntry()   ← money earmarked
    │     DEBIT  LIABILITY_WALLETS      30,000
    │     CREDIT LIABILITY_SETTLEMENTS  30,000
    │
    ├── DisbursementRequestEntity (status: PENDING)
    │
    └── SnippeGateway.initiatePayout()
          POST /v1/payouts/send → Snippe

    ... Snippe processes ...

    ▼
Snippe webhook: payout.completed
    │
    ▼
[DisbursementService.onPayoutCompleted()]
    │
    ├── LedgerService.writeEntry()   ← money physically left
    │     DEBIT  LIABILITY_SETTLEMENTS  30,000
    │     CREDIT ASSET_PSP_SNIPPE       30,000
    │
    └── DisbursementRequestEntity (status: COMPLETED)
```

End state: User wallet debited. Money left Snippe. Ledger balanced.

4.11 Withdrawal Failed

```
Snippe webhook: payout.failed
    │
    ▼
[DisbursementService.onPayoutFailed()]
    │
    ├── LedgerService.writeEntry()   ← earmark reversed
    │     DEBIT  LIABILITY_SETTLEMENTS  30,000
    │     CREDIT LIABILITY_WALLETS      30,000
```

```
    │
    ├── WalletService.credit(accountId, 30,000, REVERSAL)
    │
    └── DisbursementRequestEntity (status: FAILED, walletRefunded: true)
```

End state: User wallet restored. No money lost.

4.12 Withdrawal Reversed (After Completion)

```
Snippe webhook: payout.reversed
    │  money came back to Snippe after successful payout
    │
    ▼
[DisbursementService.onPayoutReversed()]
    │
    ├── LedgerService.writeEntry()
    │     DEBIT  ASSET_PSP_SNIPPE       30,000   ← money back at Snippe
    │     CREDIT LIABILITY_WALLETS      30,000   ← returned to user
    │
    ├── WalletService.credit(accountId, 30,000, REVERSAL)
    │
    └── DisbursementRequestEntity (status: REVERSED, walletRefunded: true)
```

4.13 Platform Offer / Subsidy Applied

Customer pays TZS 8,000. Platform covers TZS 2,000 delivery fee.

```
Two TransactionEntities for same order:
    1. Customer: 8,000 via USSD → escrow
    2. Platform: 2,000 subsidy (paidBy: PLATFORM) → escrow

On delivery confirmed:
    LedgerService.writeEntry()
      DEBIT  ASSET_ESCROW                    10,000
      CREDIT LIABILITY_WALLETS                8,000  (kitchen)
      CREDIT LIABILITY_WALLETS                1,400  (rider — 70% of 2,000)
      CREDIT REVENUE_DELIVERY_MARGIN            600  (platform — 30% of 2,000)
      CREDIT REVENUE_MARKETPLACE_COMMISSION     X    (commission on food amount)
```

```
    SplitFundedBy.PLATFORM on delivery split — recorded for reporting
```

---

5. Idempotency Guards

Every entry point is protected against duplicates:

Layer	Guard Mechanism
PSP Collection	`PspCollectionLogEntity.providerTransid` unique constraint
PSP Disbursement	`PspDisbursementLogEntity.pspEventId` unique constraint
Collection initiation	`PspCollectionEntity.idempotencyKey` unique constraint
Disbursement initiation	`DisbursementRequestEntity.idempotencyKey` unique constraint
Wallet debit	`WalletEntity @Version` optimistic lock
Ledger account update	`LedgerAccountEntity @Version` optimistic lock

If a duplicate webhook arrives → `DataIntegrityViolationException` on log save → silently skipped.

---

6. Safety Rules

Golden Rule (continuous check)

```
ASSET_PSP_SNIPPE + ASSET_ESCROW >= LIABILITY_WALLETS + LIABILITY_SETTLEMENTS
```

Escrow Integrity Check (nightly)

```
ASSET_ESCROW balance = SUM(EscrowEntity WHERE status = HELD)
```

Balance Drift Check (nightly)

```
For each LedgerAccountEntity:
  stored balance = recalculated from JournalEntryLines
  if mismatch → alert immediately
```

Wallet Refund Guard

```
DisbursementRequestEntity.walletRefunded = true
prevents double-refund on concurrent failure callbacks
```

---

7. Disbursement Channel Lifecycle

```
User adds channel (phone or bank account)
    |
    ▼
DisbursementChannelEntity
  status: ACTIVE
  nameVerified: false        ← placeholder until Snippe name lookup available
  otpVerified: true          ← no OTP for now
  isPrimary: true            ← if first channel
    |
    ▼
User initiates withdrawal
    |
    ▼
DisbursementChannelService.getChannel()
  channel.isUsable() check:
    status == ACTIVE
    otpVerified == true
    deletedAt == null
    |
    ▼
DisbursementService.initiate() proceeds
```

---

8. Data Entities — Quick Reference

Entity	Purpose
`LedgerAccountEntity`	Chart of accounts — one per `LedgerAccountCode`
`JournalEntryEntity`	One entry per money event — append only, never updated
`JournalEntryLineEntity`	Individual debit/credit lines per journal entry

Entity	Purpose
`WalletEntity`	Per-user balance + status
`WalletTransactionEntity`	Every wallet movement with balanceBefore/balanceAfter
`PspCollectionEntity`	Incoming payment request lifecycle
`PspCollectionLogEntity`	Raw webhook log — idempotency guard
`DisbursementRequestEntity`	Outgoing payout lifecycle
`DisbursementChannelEntity`	User's saved payout destinations
`PspDisbursementLogEntity`	Raw payout webhook log — idempotency guard
`PaymentTransactionEntity`	Business-level payment record per order
`TransactionSplitEntity`	Split breakdown per transaction
`EscrowEntity`	Held money with release condition

9. What Triggers What — Event Map

```
Customer pays via USSD
    └── Snippe webhook: payment.completed
            └── SnippeWebhookService.handle()
                    └── PspCollectionService.onPaymentCompleted()
                            ├── WALLET_TOPUP → WalletService.credit()
                            └── ORDER_PAYMENT → OrderPaymentService notified

Rider confirms delivery
    └── OrderPaymentService.confirmDelivery()
            └── TransactionService.releaseEscrow()
                    ├── LedgerService.writeEntry()
                    └── WalletService.credit() × N recipients

Order cancelled
    └── OrderPaymentService.cancelOrder()
            └── TransactionService.refundEscrow()
                    ├── LedgerService.writeEntry()
                    └── WalletService.credit() if wallet payment

User withdraws
    └── DisbursementService.initiate()
```

```
        ├── WalletService.debit()
        ├── LedgerService.writeEntry()
        └── SnippeGateway.initiatePayout()
                └── Snippe webhook: payout.completed / payout.failed / payout.reversed
                        └── DisbursementService.onPayout*()
                                ├── LedgerService.writeEntry()
                                └── WalletService.credit() if failed/reversed
```

Revision #1
Created 11 May 2026 01:28:28 by Admin Qbit
Updated 11 May 2026 01:28:44 by Admin Qbit