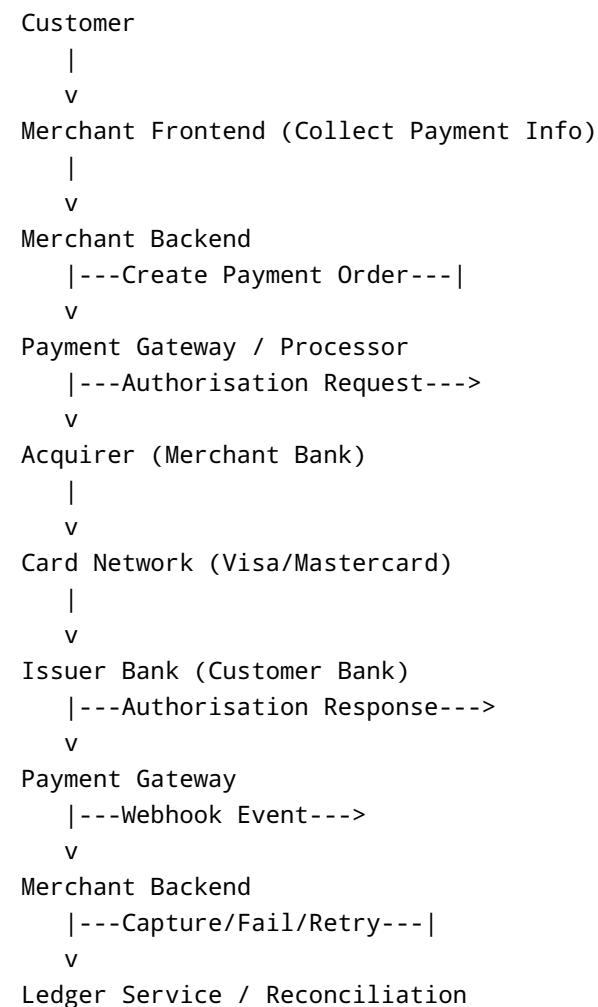


Visual Payment Flow & Architecture for SaaS/Fintech (Markdown + Pseudocode)

This document provides a visual, step-by-step flow of payment processing with pseudocode annotations. It is designed for beginners to advanced engineers preparing for production systems or interviews.

1 . High-Level Visual Flow



2 . Authorisation vs Capture Flow

```
function authorisePayment(payment):
    response = gateway.authorise(payment)
    if response.requiresSCA:
        redirectToBankApp(response.authUrl)
    elif response.declined:
        markPaymentFailed(payment.id)
    else:
        markPaymentAuthorised(payment.id)

function capturePayment(paymentId):
    gateway.capture(paymentId)
    markPaymentCaptured(paymentId)
```

3 . SCA / 3 DS 2 Flow

```
Customer enters card -> Gateway -> Issuer checks SCA
    if SCA required -> Redirect to Bank App / OTP / Biometrics
        Customer authenticates
Issuer returns auth result -> Gateway -> Webhook to Merchant -> Update Payment Status
```

Pseudocode:

```
if payment.requiresSCA:
    redirectToBankApp(payment.authUrl)
await webhook.payment_authorised
updateOrder(payment.id)
```

4 . Multi-Provider Routing Diagram

```
if payment.currency in ['EUR', 'GBP']:
    primary = Adyen
    fallback = Stripe
else:
    primary = Stripe
```

```

    fallback = Adyen

try:
    primary.authorise(payment)
except TemporaryFailure:
    fallback.authorise(payment)

```

Flow:

```

Payment Initiated
|
v
Routing Logic -> Provider Selection -> Primary / Fallback
|
v
Gateway Authorisation -> Webhook Event -> Merchant Backend

```

5 . Ledger & Double-Entry Flow

```

PaymentCaptured:
    debit: MerchantReceivable
    credit: Revenue
RefundIssued:
    debit: Revenue
    credit: MerchantReceivable
ChargebackReceived:
    debit: ChargebackLoss
    credit: MerchantReceivable

```

Flow Diagram:

```

Payment Captured --> Ledger Update
Refund Issued --> Ledger Update
Chargeback Received --> Ledger Update
Reconciliation Job --> Compare Ledger vs Gateway vs Acquirer Reports

```

6 . Chargeback & Refund Flow

```
Customer Disputes -> Issuer -> Network -> Acquirer -> Merchant  
Merchant submits evidence -> Network decides  
Ledger updated based on result
```

Refund Flow:

```
Merchant Initiates Refund -> Gateway -> Issuer -> Customer  
Ledger Updated Independently  
Webhook Updates Merchant State
```

7 . Reconciliation & Idempotency

```
function authorise(request):  
    if request.idempotencyKey in db:  
        return db[request.idempotencyKey]  
    response = gateway.authorise(request)  
    db[request.idempotencyKey] = response  
    return response  
  
function reconcile():  
    for settlement in acquirerReport:  
        match = findInternalPayment(settlement.reference)  
        if match.status != 'Captured':  
            raiseAlert()
```

8 . End-to-End Event Flow

```
PaymentInitiated -> PaymentAuthorised -> PaymentCaptured -> PaymentFailed ->  
RefundIssued -> ChargebackReceived
```

- Webhooks are critical - Outbox pattern ensures events are not lost - Each event triggers ledger & state update

9 . Compliance Overlay

```
PCI DSS -> Gateway handles card, tokenization, encryption  
PSD2/SCA -> 2FA enforced via 3DS2 / bank app  
Open Banking -> Bank login & approval for A2A payments
```

Flow Overlay Diagram:

```
Customer -> Gateway -> Compliance Check (SCA/Tokenization) -> Authorisation ->  
Webhook -> Merchant -> Ledger / Reconciliation
```

Summary

This Markdown diagram guide combines:
- High-level flows for payment initiation to settlement
- 3 DS 2 simulation
- Multi-provider routing
- Double-entry ledger
- Webhook handling & idempotency
- Chargeback/refund flows
- Compliance overlay

It is designed to ~~single reference for beginners to advanced engineers~~ for both implementation and interview prep.