# Complete Beginner’s Guide to Payment Processing

Purpose: This document explains the full payment ecosystem from Merchant, Fintech (Gateway/Processor), Acquirer, Card Networks, and Issuer Banks. It is written for complete beginners and includes architecture diagrams and pseudocode.

---

# 1. High-Level Mental Model

A card payment is not "money moving instantly". It is a structured conversation between:

- Customer
- Merchant
- Payment Gateway / Processor
- Acquirer (merchant’s bank)
- Card Network
- Issuer (customer’s bank)

The process happens in stages:

1. Authorisation (permission + hold funds)
2. Capture (confirm transaction)
3. Clearing
4. Settlement
5. Reconciliation
6. Possible Chargeback

---

# 2. Responsibilities by Participant

---

## 2.1 Customer

### Responsibility

- Provides payment details
- Completes authentication (if required)
- Can dispute transactions

### What customer never sees

- Interchange fees
- Acquirer settlement batching
- Fraud scoring

---

## 2.2 Merchant (Your SaaS / Ecommerce Business)

### Responsibilities

- Collect payment intent from customer
- Send payment request to gateway
- Store internal order state
- Handle asynchronous webhooks
- Fulfil product/service
- Handle refunds
- Reconcile settlements

### Merchant Internal State Machine

Example:

```
OrderStatus = {
  Pending,
  PaymentAuthorised,
  PaymentFailed,
  PaymentCaptured,
  Refunded,
  Chargeback
}
```

Merchant must NEVER assume payment success until webhook confirms it.

---

## 2.3 Payment Gateway / Processor

### Responsibilities

- Securely collect card details (PCI compliance)

- Tokenise card data
- Send authorisation request to acquirer
- Handle fraud checks
- Orchestrate 3 D Secure authentication
- Return transaction status
- Send webhooks to merchant

### Gateway Does NOT:

- Hold merchant funds permanently
- Issue final settlement to business bank (acquirer does)

---

## 2.4 Acquirer (Merchant's Bank)

### Responsibilities

- Provides merchant account
- Receives authorised transactions
- Sends transaction to card network
- Handles settlement
- Manages chargebacks
- Transfers funds to merchant bank account

---

## 2.5 Card Network (Visa/Mastercard etc.)

### Responsibilities

- Routes transaction between acquirer and issuer
- Defines dispute rules
- Calculates interchange fees

They do NOT: - Hold money long-term - Talk directly to merchant

---

## 2.6 Issuer Bank (Customer's Bank)

### Responsibilities

- Approves or declines transaction
- Applies fraud detection

- Reserves customer funds
- Handles customer disputes

---

# 3 . Full Payment Lifecycle (Card Payment)

---

## Stage 1 : Payment Initiation

Customer submits card details.

Merchant Backend:

```
function createPayment(orderId, amount):
    payment = new Payment(orderId, amount)
    save(payment)
    return gateway.authorise(payment)
```

---

## Stage 2 : Authorisation

Flow:

Customer → Gateway → Acquirer → Network → Issuer

Issuer checks: - Card validity - Funds - Fraud - SCA requirement

Possible responses:

- Approved
- Soft Decline (authentication required)
- Hard Decline

---

## Stage 3 : Strong Customer Authentication (if required)

If SCA required:

```
if response.requiresAuthentication:
    return redirectToBank(response.authUrl)
```

Customer approves in banking app.

Issuer sends result back to gateway.

Gateway sends webhook:

```
POST /webhook
{
  "event": "payment.authorised",
  "paymentId": "123"
}
```

Merchant updates state:

```
function handleWebhook(event):
    if event.type == "payment.authorised":
        markPaymentAuthorised(event.paymentId)
```

---

# Stage 4 : Capture

Capture can be automatic or manual.

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

This triggers clearing and settlement process.

---

# Stage 5 : Clearing & Settlement

Clearing: - Transaction details exchanged

Settlement: - Funds move issuer → acquirer → merchant

Usually batched daily.

Merchant receives payout report.

---

# Stage 6 : Reconciliation

Merchant compares:

- Internal database transactions
- Gateway report
- Acquirer settlement report

Pseudocode:

```
for each settlement in acquirerReport:
    match = findInternalPayment(settlement.reference)
    if not match:
        raiseAlert()
```

---

# Stage 7 : Refunds

Refund flow:

Merchant → Gateway → Acquirer → Issuer → Customer

Refund does NOT cancel original transaction. It creates a new reversing transaction.

---

# Stage 8 : Chargeback Lifecycle

1. Customer disputes transaction
2. Issuer creates chargeback
3. Acquirer removes funds from merchant
4. Merchant receives notification
5. Merchant submits evidence
6. Network arbitration decision

Merchant must maintain: - Logs - Delivery confirmation - Authentication proof

---

# 4. Authorisation vs Capture

Authorisation: - Reserve funds - Temporary hold

Capture: - Confirm transaction - Triggers money movement

---

# 5. Soft Decline vs Hard Decline

Soft Decline: - Retry possible - Often SCA required

Hard Decline: - Do not retry - Card invalid / blocked

---

# 6. PSD2 & SCA (Europe/UK)

## PSD2 Purpose

Reduce online fraud by requiring customer authentication.

## SCA Rule

Two of three factors required:

- Knowledge (password/PIN)
- Possession (phone/app)
- Inherence (biometrics)

---

# 7. 3D Secure 2 Flow

```
Customer submits card
↓
Issuer decides SCA required
↓
Customer redirected to bank app
↓
Customer approves
↓
Issuer returns authentication result
```

```
↓
Payment authorised
```

---

# 8. Open Banking Payments

Instead of card:

```
Redirect to bank login
↓
Customer approves transfer
↓
Bank sends confirmation
↓
Merchant marks paid
```

Benefits: - Lower fees - No chargebacks

---

# 9. Idempotency (Critical for Engineers)

Prevent duplicate charge on retry.

```
function authorise(request):
    if exists(request.idempotencyKey):
        return storedResponse

    response = gateway.authorise(request)
    store(request.idempotencyKey, response)
    return response
```

---

# 10. Production-Grade Architecture

Components:

- API Layer
- Payment Service
- Webhook Processor
- Ledger Service

- Reconciliation Job

Event-driven model:

```
PaymentInitiated
PaymentAuthorised
PaymentCaptured
PaymentFailed
RefundIssued
ChargebackReceived
```

Ledger should use double-entry accounting.

---

# 11. Common Beginner Mistakes

- Assuming synchronous payment success
- Not verifying webhook signatures
- Not storing gateway event IDs
- Ignoring reconciliation
- Mixing payment logic with order logic

---

# 12. Final Mental Model

Payments are:

- Distributed
- Asynchronous
- Regulated
- Dispute-driven
- Eventually consistent

Your job as a fintech engineer is to build systems that:

- Track state correctly
- Handle retries safely
- Remain auditable
- Survive failures

---

# End of Guide

This document provides a complete structural overview for beginners entering payment processing systems.