# ****Domain Driven Design Document Digital Payment Management System (DPMS)****

## ****Introduction****

### ****Purpose/Objective****

The purpose of this DDD documentation is to model and structure the business logic of a Digital Payment Management System (DPMS) using Domain-Driven Design principles. This system enables users to manage and perform digital transactions including payments, refunds, and settlement processes. It aims to clearly define the domain, identify bounded contexts, and align business processes with technical implementations.

## ****Domain Scope****

The DPMS will cater to:

Merchants (who accept digital payments)

End-users/consumers (who initiate payments)

Bank partners and payment gateways

Administrators and compliance officers (for monitoring and audit)

## ****Domain Description****

The Digital Payment Management System is responsible for:

Initiating, processing, and tracking digital payments

Handling refunds and reversals

Transaction reconciliation and settlement

Notification and alert mechanisms

Reporting and analytics

Integrating with banks, payment gateways, and third-party fraud systems

## ****Context****

This document is focused on the core domain and its strategic boundaries. It includes only the core transaction, settlement, refund, and reconciliation domains but excludes external systems' internal logic like bank APIs or fraud systems.

## ****Strategic Design****

### ****Bounded Contexts****

#### ****1. Payment Context****

**Responsibility**: Manage the lifecycle of payment transactions.

**Ubiquitous Language**: PaymentRequest, TransactionStatus, PaymentMethod

**Key Domain Concepts**: Transaction ID, Payment Channel, Authorization, Capture

#### ****2. Refund Context****

**Responsibility**: Initiates and tracks refunds for successful transactions.

**Ubiquitous Language**: RefundRequest, RefundStatus, Reversal

**Key Concepts**: Refund ID, OriginalTransaction, RefundReason

#### ****3. Settlement Context****

**Responsibility**: Performs settlement operations with banks or merchants.

**Ubiquitous Language**: SettlementBatch, NetAmount, Fee, Commission

**Key Concepts**: Batch ID, SettlementDate, Charges

#### ****4. Reconciliation Context****

**Responsibility**: Matches transactions between DPMS and external systems (banks).

**Ubiquitous Language**: MatchStatus, ReconFile, Dispute

**Key Concepts**: Mismatch, MissingRecord, ReconciledEntry

#### ****5. Notification Context****

**Responsibility**: Handles alerts, status notifications (SMS, Email, Webhook).

**Ubiquitous Language**: AlertType, DeliveryChannel, NotificationStatus

**Key Concepts**: Message ID, RetryPolicy, DeliveryStatus

### ****Context Map****

### ****Sub-Domains****

### ****1. Payment Context****

#### Entities

PaymentTransaction

**Attributes:** ID, Amount, Timestamp, Status, Channel, Currency

#### Value Objects

Amount

PaymentMethod

Status

#### Aggregates

PaymentTransaction (Aggregate Root)

#### Domain Services

TransactionValidatorService

PaymentProcessorService

#### Domain Events

PaymentInitiated

PaymentAuthorized

PaymentFailed

#### 

#### Application Services

initiatePayment()

checkStatus()

cancelTransaction()

### ****2. Refund Context****

#### Entities

RefundTransaction

**Attributes:** RefundID, ParentTransactionID, Amount, Status

#### Value Objects

RefundReason

RefundAmount

#### Aggregate

RefundTransaction (Aggregate Root)

#### Domain Services

RefundService

#### Domain Events

RefundInitiated

RefundCompleted

#### Repositories

RefundTransactionRepository

#### Application Services

initiateRefund()

trackRefundStatus()

### ****3. Settlement Context****

#### Entities

SettlementBatch

#### Value Objects

NetAmount

Fee

#### Aggregates

SettlementBatch (Aggregate Root)

#### Domain Events

SettlementCompleted

#### Application Services

generateSettlement()

getSettlementReport()

### ****4. Notification Context****

#### Entities

NotificationMessage

#### Value Objects

AlertType

DeliveryChannel

#### Application Services

sendSMS()

sendEmail()

sendWebhook()

## ****Implementation Considerations****

### ****Technology Stack****

**Backend**: Spring Boot

**Frontend**: Angular

**Messaging**: RabbitMQ or Kafka

**ORM**: Hibernate (JPA), OXM (for XML mappings)

### ****Architectural Patterns****

## ****Testing Security Glossary****

| **Term** | **Definition** |
| --- | --- |
| PaymentTransaction | Represents a digital payment from user to merchant |
| Refund | Reversal of a previously successful payment |
| Settlement | Movement of funds from acquirer to merchant |
| Reconciliation | Matching transaction data between DPMS and external sources |
| Notification | Communication sent for updates (SMS/Email/Webhook) |