# **Domain Driven Design Document**

## **[Digital Payment System]**

### **I. Introduction**

### **A. Purpose/Objective**

The purpose of this Domain Driven Design (DDD) documentation is to provide a structured and comprehensive understanding of the Digital Payment System domain. The primary objective is to guide the design and development of a secure, scalable, and efficient payment processing system by identifying core domain concepts, their relationships, and the boundaries within which they operate. This document aims to align technical implementation with business goals, ensuring the system meets the needs of stakeholders effectively.

#### **B. Domain Scope**

The Digital Payment System targets end-users requiring electronic payment solutions, including individual consumers, merchants, and potentially financial institutions. The system supports multiple payment methods (e.g., credit/debit cards, digital wallets, bank transfers,SWIFT,Digital wallets) and focuses on secure, real-time transaction processing, account management.

#### **C. Domain Description**

The Digital Payment System domain encapsulates the processes and entities involved in facilitating electronic payments. Key functionalities include user authentication, payment method management, transaction processing, fraud detection, financial record-keeping, and integration with external financial systems (e.g., banks, payment gateways). The system ensures secure handling of sensitive data, supports multi-currency transactions, and provides users with transaction histories and account management capabilities.

#### **D. Context**

This analysis centers on the core operations of a digital payment system, excluding peripheral services such as customer support or marketing analytics unless they directly influence payment processing workflows. The focus is on the domain logic critical to payment execution and management.

### **II. Strategic Design**

#### **A. Bounded Contexts**

The Digital Payment System is divided into distinct bounded contexts, each with specific responsibilities, ubiquitous languages, and domain concepts. Below are the identified contexts:

1. **Payment Processing**
   1. **Responsibility and Boundaries**: Handles the lifecycle of payment transactions, including validation, authorization, capture, settlement, and refunds. Interacts with Directory Servers and Banks.
   2. **Ubiquitous Language**: Transaction, Payment Gateway, Authorization, Capture, Settlement, Refund, Amount, Currency.
   3. **Key Domain Concepts**:
      1. Transaction: A financial operation with states (e.g, Captured,Rejected , Not Approved,Voided).
      2. Payment Method: The instrument used for payment (e.g, card, wallet).
      3. Merchant Account: The recipient’s account for settled funds.
2. **User Management**
   1. **Responsibility and Boundaries**: Manages user accounts, including registration, authentication, and role-based access control.
   2. **Ubiquitous Language**: User, Account, Authentication, Authorization, Role, Permission.
   3. **Key Domain Concepts**:
      1. User Profile: Details of a registered user.
      2. Role: Defines user permissions.
3. **Payment Method Management**
   1. **Responsibility and Boundaries**: Stores and manages payment methods associated with user accounts, ensuring secure handling of sensitive data.
   2. **Ubiquitous Language**: Payment Method, Card Details, Wallet, Validation, Tokenization.
   3. **Key Domain Concepts**:
      1. Payment Method: A reusable payment instrument.
      2. Token: A secure representation of payment details.
4. **Fraud Detection**
   1. **Responsibility and Boundaries**: Monitors transactions for suspicious activity, flagging or blocking potential fraud.
   2. **Ubiquitous Language**: Fraud, Risk Score, Transaction Monitoring, Alert, Block.
   3. **Key Domain Concepts**:
      1. Risk Score:A calculated likelihood of fraud.
      2. Fraud Alert: Rejection of transaction.
5. **Compliance and Regulation**
   1. **Responsibility and Boundaries**: Ensures adherence to financial regulations (e.g., PCI-DSS) across all operations.
   2. **Ubiquitous Language**: Compliance, Regulation, Audit, Standard, Policy.
   3. **Key Domain Concepts**:
      1. Audit Trail: Record of compliance-related activities.
      2. Policy: Rules enforcing regulatory standards.

#### **B. Context Map**

The context map illustrates relationships between bounded contexts, specifying interaction types:

* **Payment Processing <-> Payment Method Management**:
  + **Relationship**: Customer-Supplier
  + **Description**: Payment Processing retrieves and validates payment methods.
  + **Integration**: API calls to fetch tokenized payment details.
* **Payment Processing <-> Fraud Detection**:
  + **Relationship**: Partnership
  + **Description**: Fraud Detection monitors transactions and may instruct Payment Processing to cancel them.
  + **Integration**: Event-driven via Transaction Initiated and Fraud Detected events.
* **Compliance and Regulation ↔ All Contexts**:
  + **Relationship**: Conformist
  + **Description**: All contexts adhere to compliance rules and standards.
  + **Integration**: Policy enforcement and audit logging.

#### **C. Sub-Domains**

Sub-domains are categorized to prioritize development efforts:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sub-Domain** | **Type** | **Description** | **Priority** |
| Payment Processing | Core | Core business logic for payment execution | High |
| User Management | Supporting | Enables user interaction with the system | Medium |
| Payment Method Management | Supporting | Manages payment instruments | Medium |
| Fraud Detection | Supporting | Ensures transaction integrity | Medium |
| Accounting | Supporting | Tracks financial data | Medium |
| Compliance and Regulation | Generic | Maintians standard compliance solutions | Low |

The core sub-domain (Payment Processing) drives competitive advantage and requires the most focus, while supporting sub-domains enable its functionality, and generic sub-domains rely on reusable solutions.

### **III. Tactical Design**

#### **A. Payment Processing**

* **Entities**:
  + Transaction: ID, amount, currency, status (Pending, Authorized, etc.), timestamps.
  + Merchant Account: ID, recipient details.
* **Value Objects**:
  + Money: Amount and currency.
  + Payment Details: Tokenized payment information.
* **Aggregates/Aggregate Root**:
  + Transaction Aggregate: Root is Transaction, includes Payment Details.
* **Domain Services**:
  + Payment Gateway Service: Communicates with external gateways.
  + Transaction Validation Service: Ensures transaction validity.
* **Domain Events**:
  + Transaction Initiated
  + Transaction Authorized
  + Transaction Settled
* **Repositories**:
  + Transaction Repository: Persists and retrieves transactions.
* **Application Services**:
  + Payment Service: processPayment(), voidPayment(), refundPayment().

#### **B. User Management**

* **Entities**:
  + User: ID, name, email, password hash.
  + Role: ID, name, permissions.
* **Value Objects**:
  + Address: Street, city, country.
  + Contact Info: Phone, email.
* **Domain Services**:
  + Authentication Service: Verifies user credentials.
  + Authorization Service: Checks permissions.
* **Domain Events**:
  + User Registered
  + User Authenticated
* **Repositories**:
  + User Repository: Manages user data persistence.
* **Factories**:
  + User Factory: Creates user instances.
* **Application Services**:
  + User Management Service: registerUser(), updateProfile().

### **IV. Implementation Considerations**

#### **A. Technology Considerations**

* **Backend**: Spring MVC/Java.
* **Frontend**: JSP for user interface.
* **Data Persistence**: JDBC/JNDI .

#### **B. Testing**

* **Unit Testing**: JUnit .
* **Integration Testing**: Manual testing.
* **Security Testing**: PCI DSS.

#### **C. Security**

* **AuthN/AuthZ**: 3DS2.2 /base24 Switch .
* **Encryption**: Sensitive data encrypted at rest and in transit.
* **Compliance**: PCI-DSS adherence for payment security.

### **V. Glossary**

#### **A. Key Terms**

* **Transaction**: A financial operation transferring funds from issuer to acquierer.
* **Payment Method**: Instrument for payment (e.g., card, digital wallet, RTGS,SWIFT).
* **Authorization**: Verification of funds availability.
* **Capture**: Transfer of authorized funds.
* **Settlement**: Final deposit into the acquirer’s account.

#### **B. Clarifications**

* Authorization vs. Capture: Authorization reserves funds; capture executes the transfer.

#### **C. Illustrative Snippets**

* **Payment Processing – Entity**
  + Entity: Transaction
  + Description: Represents a payment transaction.
  + Attributes:
    - Transaction\_Id
    - Amount
    - Currency
    - Status
    - Timestamps
* **Context Map: Payment Processing <-> User Management**
  + Relationship: Customer-Supplier
  + Description: Payment Processing depends on User Management for authentication.
  + Integration: API calls for user validation.