**Project Idea: National Digital Public Distribution System (Digi-PDS)**

This is a country-level system designed to modernize and manage the entire lifecycle of food grain and essential commodity distribution under a government's public distribution system (PDS). The goal is to ensure transparency, eliminate fraud, and guarantee that entitled citizens receive their rations efficiently.

**Core Concept**

The **Digi-PDS** is a microservices-based platform that connects farmers/procurement agencies, central warehouses, state-level distribution centers, Fair Price Shops (FPS), and citizens. It uses an event-driven model to track goods from procurement to the final beneficiary, handling everything from inventory and logistics to payments and grievance redressal.

**System Architecture & Microservices**

The system would be broken down into the following key microservices:

1. **👤 User & Auth Service:** Manages all users (Citizens, FPS Owners, Warehouse Managers, Admin Officials). Handles **JWT-based authentication and role-based authorization**.
2. **📇 Citizen & Entitlement Service:** Manages citizen data, ration card details, family members, and their monthly entitlements based on government schemes.
3. **🏢 Procurement & Inventory Service:** Manages the procurement of grains from farmers/agencies. It tracks stock levels, quality checks, and inventory across multiple central and state warehouses.
4. **🚚 Logistics Service:** Handles the transportation of goods between warehouses and from warehouses to Fair Price Shops. It generates transport orders and tracks shipments.
5. **🛒 Fair Price Shop (FPS) Service:** Manages FPS details, their current stock, and daily transaction logs.
6. **💳 Transaction & Payment Service:** Processes the transactions when a citizen collects their ration. It integrates a **payment gateway to generate a dynamic QR code and show a UPI ID** for the subsidized payment amount.
7. **📢 Notification Service:** Sends real-time SMS, email, or app notifications to citizens (e.g., "Your monthly ration is available for pickup") and FPS owners (e.g., "Your stock has been dispatched").
8. **📊 Reporting & Analytics Service:** Gathers data from all services to generate dashboards for government officials, showing distribution statistics, stock levels, and potential bottlenecks.

**How It Meets All Your Requirements**

Here’s a breakdown of how this project satisfies each of your technical points:

**1. Event-Driven Architecture (EDA)** The entire system is built around events using a message broker like **RabbitMQ or Kafka**.

* **Example Flow:** When a warehouse dispatches a truck to an FPS, the **Logistics Service** publishes a StockDispatched event. The **FPS Service** subscribes to this event to update its "incoming stock" status. The **Notification Service** also subscribes to it to send an SMS to the FPS owner. This decouples the services and makes the system resilient.

**2. Multiple Entities & 15+ Database Tables** The domain is complex, leading to a rich database schema.

* **Entities:** User, Role, Citizen, FamilyMember, RationCard, Scheme, Commodity, Warehouse, StockItem, Vehicle, TransportOrder, FairPriceShop, Transaction, PaymentDetail, Grievance, NotificationLog.
* This easily creates **more than 15 tables** with various relationships.

**3. Payment Gateway** The **Transaction & Payment Service** handles this.

* When a citizen collects their ration, the system calculates the subsidized amount. It then makes an API call to a payment gateway (e.g., Razorpay, PhonePe) to generate a **dynamic QR code for that specific transaction amount**, which is displayed on the FPS owner's device.

**4. Authentication & Authorisation** The **User & Auth Service** is the central authority.

* **Authentication:** Citizens, officials, and FPS owners log in with credentials to get a JSON Web Token (JWT).
* **Authorization:** The API Gateway and individual microservices use this JWT to enforce rules. For example, only a user with an FPS\_OWNER role can report a sale, and only a user with a WAREHOUSE\_MANAGER role can accept new inventory.

**5. Multithreading & Async Functions** This is crucial for performance and responsiveness.

* **Spring's @Async annotation** would be used extensively. For instance, after a transaction is completed, the system can **asynchronously** call the Notification Service to send an SMS, update analytics data, and log the transaction without making the FPS owner wait.

**6. Inter-Service Communication** Services communicate both synchronously (REST API) and asynchronously (Events).

* **Sync Example:** The **FPS Service** might make a synchronous REST call to the **Citizen Service** to verify a citizen's details in real-time.
* **Async Example:** The **Transaction Service** publishes a TransactionCompleted event, which is consumed by the **Inventory Service** (to deduct stock) and the **Reporting Service**.

**7. Bidirectional Relationships** The data model is rich with JPA relationships.

* **OneToMany:** A Warehouse has many StockItems.
* **ManyToOne:** Many Transactions are associated with one FairPriceShop.
* **ManyToMany:** A Scheme (e.g., "Food Security Act") can have many Commodity types (Wheat, Rice), and a Commodity can be part of multiple Schemes. A join table scheme\_commodities would manage this.

**8. Validation, Exception Handling & AOP** These Spring concepts are critical for a robust application.

* **Validation:** Using jakarta.validation annotations (@NotNull, @Min, etc.) in DTOs to validate all incoming request data.
* **Global Exception Handling:** Using @ControllerAdvice to handle exceptions gracefully across the application (e.g., returning a 404 Not Found if a citizen ID doesn't exist).
* **Aspect-Oriented Programming (AOP):** Creating custom annotations like @LogActivity to automatically log method entry/exit for auditing purposes, or @TrackExecutionTime to monitor the performance of critical methods. This keeps the business logic clean.

**Data Flow Diagrame – DIGI PDS**

A Data Flow Diagram (DFD) is perfect for visualizing how information moves through the Digi-PDS.

Here’s a breakdown, starting from a high-level overview and then drilling down into the specifics.

**Level 0 DFD: Context Diagram**

This is the simplest, highest-level view. It shows the entire **Digi-PDS System** as a single process and illustrates how external entities interact with it.

**External Entities:**

* **Citizen:** The beneficiary who receives the rations.
* **Fair Price Shop (FPS) Owner:** The person who distributes the rations.
* **Warehouse Manager:** Manages stock at the warehouse level.
* **Procurement Agency:** Supplies grains to the warehouses.
* **Government Official:** Monitors the system and views analytics.
* **Payment Gateway:** An external service to process UPI payments.

**Major Data Flows (Input/Output):**

1. **Citizen** ↔️ **System**:
   * ➡️ **Input:** Login details, Ration Card ID for verification, Grievances.
   * ⬅️ **Output:** Entitlement info, Transaction confirmation SMS, Notification alerts.
2. **FPS Owner** ↔️ **System**:
   * ➡️ **Input:** Login, Sales data, Stock requests.
   * ⬅️ **Output:** Citizen verification status, Stock dispatch alerts, QR Code for payment.
3. **Warehouse Manager** ↔️ **System**:
   * ➡️ **Input:** Login, Stock received confirmation, Dispatch details.
   * ⬅️ **Output:** Transport orders, Inventory reports.
4. **Procurement Agency** ↔️ **System**:
   * ➡️ **Input:** Grain procurement details, Quality check results.
   * ⬅️ **Output:** Procurement orders, Payment confirmations.
5. **Government Official** ➡️ **System**:
   * ⬅️ **Output:** Analytics dashboards, Distribution reports, Stock level reports.
6. **System** ↔️ **Payment Gateway**:
   * ➡️ **Output to Gateway:** Request to generate payment QR with amount.
   * ⬅️ **Input from Gateway:** Payment success/failure confirmation.

**Level 1 DFD: Detailed Breakdown**

This diagram "explodes" the single process from the Level 0 diagram into the main microservices (processes) and shows how data flows between them and the data stores (databases).

**Key Processes (Microservices):**

* **1.0** User & Auth Service
* **2.0** Citizen & Entitlement Service
* **3.0** Inventory & Procurement Service
* **4.0** Logistics Service
* **5.0** FPS Transaction Service
* **6.0** Payment Service
* **7.0** Notification Service
* **8.0** Reporting & Analytics Service

**Data Stores (Databases):**

* **D1** - Users & Roles DB
* **D2** - Citizen & Schemes DB
* **D3** - Warehouse & Inventory DB
* **D4** - FPS & Transactions DB

**Core Data Flow Example: Citizen Collecting Ration**

Let's trace the most common data flow:

1. A **Citizen** goes to an FPS. The **FPS Owner** enters the Citizen's Ration Card ID into the system.
2. The Ration Card ID goes to the **5.0 FPS Transaction Service**.
3. The **FPS Service (5.0)** sends a Verification Request to the **2.0 Citizen & Entitlement Service** to check if the citizen is valid and what their monthly entitlement is.
4. The **Citizen Service (2.0)** reads from the **D2 Citizen & Schemes DB** and returns Entitlement Details (e.g., "Eligible for 5kg Wheat, 10kg Rice").
5. The **FPS Service (5.0)** calculates the subsidized Bill Amount and sends a Payment Request to the **6.0 Payment Service**.
6. The **Payment Service (6.0)** contacts the external **Payment Gateway** to generate a Dynamic QR Code. This is displayed to the **FPS Owner**.
7. The **Citizen** scans the QR and pays. The **Payment Gateway** sends a Payment Confirmation back to the **Payment Service (6.0)**.
8. The **Payment Service (6.0)** sends the Success Status to the **FPS Service (5.0)**.
9. The **FPS Service (5.0)** records the sale by writing Transaction Data to the **D4 FPS & Transactions DB**.
10. **CRITICAL EVENT:** The **FPS Service (5.0)** then publishes a TransactionCompleted event to the message queue (Kafka/RabbitMQ).
11. Two services are listening for this event:
    * The **3.0 Inventory Service** hears it and updates the stock levels for that FPS in the **D3 Warehouse & Inventory DB**.
    * The **7.0 Notification Service** hears it and sends a confirmation SMS/Push Notification to the **Citizen**.
12. Finally, the **8.0 Reporting Service** periodically pulls data from all databases to update the dashboards for the **Government Official**.

**UML (Unified Modelling Language)**

UML (Unified Modeling Language) diagrams is an excellent way to visualize the software design before implementation.

Of course. Creating UML (Unified Modeling Language) diagrams is an excellent way to visualize the software design before implementation.

Here are the key UML diagrams for the **Digi-PDS project**: a **Use Case Diagram**, a **Component Diagram**, and a detailed **Class Diagram**.

**1. Use Case Diagram**

This diagram illustrates the high-level functionalities of the system and how different users (actors) interact with it. It shows *what* the system does, not *how* it does it.

**Actors:**

* **Citizen:** The primary beneficiary.
* **FPS Owner:** The distributor at the Fair Price Shop.
* **Warehouse Manager:** Manages inventory at warehouses.
* **Admin Official:** A government user who monitors the system.

**Key Use Cases:**

* **Verify Entitlement:** A core process where a citizen's eligibility is checked.
* **Dispense Rations:** The main transaction flow, initiated by the FPS Owner. This use case <<includes>> verifying entitlement and processing payment.
* **Process Payment:** A specific sub-task, handled via the payment gateway.
* **Manage Inventory:** Actions performed by the Warehouse Manager, like updating stock levels.
* **View Reports:** A function for Admin Officials to monitor the system's performance.
* **File Grievance:** A citizen's ability to report issues.

**2. Component Diagram**

This diagram shows the high-level architecture of the system, breaking it down into its physical components (in this case, our microservices). It's perfect for visualizing the service-oriented architecture.

**Components (Microservices):**

* **APIGateway:** The single entry point for all client requests. It handles routing, authentication, and rate limiting.
* **AuthService:** Manages user authentication and authorization.
* **CitizenService:** Manages citizen data and entitlements.
* **InventoryService:** Manages stock across all warehouses.
* **FPSService:** Manages Fair Price Shops and their transactions.
* **PaymentService:** Integrates with the external payment gateway.
* **NotificationService:** Sends alerts via SMS/email.
* **MessageQueue (e.g., Kafka/RabbitMQ):** The central nervous system for our event-driven architecture. This isn't a service we build, but a crucial infrastructure component that facilitates asynchronous communication.

**3. Class Diagram**

This is the most detailed diagram, showing the static structure of the system. It maps directly to the database tables and Java classes you'll create with Spring Data JPA. It includes attributes, methods, and the relationships between entities.

**Key Classes & Relationships:**

* **User & Role:** A standard security setup. A User has one Role, but a Role can be assigned to many Users (**ManyToOne**).
* **Citizen & RationCard:** A Citizen has one RationCard, which contains a list of FamilyMember objects (**OneToOne** and **OneToMany** composition).
* **FairPriceShop & Transaction:** An FPS can have many Transactions, but each transaction belongs to only one FPS (**OneToMany**).
* **Transaction & PaymentDetail:** Each Transaction has exactly one PaymentDetail associated with it (**OneToOne**).
* **Warehouse & StockItem:** A Warehouse contains many StockItems (**OneToMany**). A StockItem represents a specific commodity at that location.
* **Commodity & Scheme (ManyToMany):** This is a classic ManyToMany relationship. A government Scheme can include multiple Commodities (e.g., rice, wheat), and a single Commodity can be part of multiple schemes. This relationship is resolved through a join table/class, SchemeCommodity.
* **Abstract Classes:** Classes like BaseEntity can be used to hold common fields like id, createdAt, and updatedAt, which all other entities can extend.

These three diagrams provide a solid foundation for understanding the system's architecture, user interactions, and underlying data structure before you start writing code.

ER Relations

**Digi-PDS - Detailed Entity Relationship Diagram**

**Entities and Attributes**

**1. USER**

* **user\_id** (PK) - BIGINT
* username - VARCHAR(50) UNIQUE NOT NULL
* password\_hash - VARCHAR(255) NOT NULL
* email - VARCHAR(100) UNIQUE
* phone\_number - VARCHAR(15)
* is\_active - BOOLEAN DEFAULT TRUE
* created\_at - TIMESTAMP
* updated\_at - TIMESTAMP
* **role\_id** (FK) → ROLE

**2. ROLE**

* **role\_id** (PK) - INT
* role\_name - VARCHAR(50) UNIQUE NOT NULL
* description - TEXT
* permissions - JSON/TEXT
* created\_at - TIMESTAMP

**3. CITIZEN**

* **citizen\_id** (PK) - BIGINT
* **user\_id** (FK) → USER
* first\_name - VARCHAR(50) NOT NULL
* last\_name - VARCHAR(50) NOT NULL
* date\_of\_birth - DATE
* gender - ENUM('MALE', 'FEMALE', 'OTHER')
* address\_line1 - VARCHAR(255)
* address\_line2 - VARCHAR(255)
* city - VARCHAR(100)
* state - VARCHAR(100)
* pincode - VARCHAR(10)
* aadhar\_number - VARCHAR(12) UNIQUE
* is\_head\_of\_family - BOOLEAN DEFAULT FALSE
* created\_at - TIMESTAMP
* updated\_at - TIMESTAMP

**4. RATION\_CARD**

* **ration\_card\_id** (PK) - BIGINT
* ration\_card\_number - VARCHAR(20) UNIQUE NOT NULL
* **citizen\_id** (FK) → CITIZEN (Head of Family)
* card\_type - ENUM('APL', 'BPL', 'AAY') NOT NULL
* issue\_date - DATE
* expiry\_date - DATE
* is\_active - BOOLEAN DEFAULT TRUE
* monthly\_income - DECIMAL(10,2)
* created\_at - TIMESTAMP
* updated\_at - TIMESTAMP

**5. FAMILY\_MEMBER**

* **family\_member\_id** (PK) - BIGINT
* **ration\_card\_id** (FK) → RATION\_CARD
* **citizen\_id** (FK) → CITIZEN
* relationship - ENUM('HEAD', 'SPOUSE', 'CHILD', 'PARENT', 'OTHER')
* is\_primary - BOOLEAN DEFAULT FALSE
* added\_date - DATE
* created\_at - TIMESTAMP

**6. SCHEME**

* **scheme\_id** (PK) - INT
* scheme\_name - VARCHAR(100) NOT NULL
* description - TEXT
* scheme\_type - ENUM('FEDERAL', 'STATE', 'SPECIAL')
* start\_date - DATE
* end\_date - DATE
* is\_active - BOOLEAN DEFAULT TRUE
* eligibility\_criteria - JSON/TEXT
* created\_at - TIMESTAMP
* updated\_at - TIMESTAMP

**7. COMMODITY**

* **commodity\_id** (PK) - INT
* commodity\_name - VARCHAR(50) NOT NULL
* commodity\_code - VARCHAR(10) UNIQUE
* unit\_of\_measurement - VARCHAR(20)
* category - ENUM('GRAIN', 'PULSE', 'OIL', 'SUGAR', 'OTHER')
* nutritional\_info - JSON
* is\_active - BOOLEAN DEFAULT TRUE
* created\_at - TIMESTAMP

**8. SCHEME\_COMMODITY (Junction Table)**

* **scheme\_commodity\_id** (PK) - BIGINT
* **scheme\_id** (FK) → SCHEME
* **commodity\_id** (FK) → COMMODITY
* quantity\_per\_person - DECIMAL(8,2)
* subsidized\_rate - DECIMAL(8,2)
* market\_rate - DECIMAL(8,2)
* is\_active - BOOLEAN DEFAULT TRUE
* effective\_from - DATE
* effective\_to - DATE

**9. ENTITLEMENT**

* **entitlement\_id** (PK) - BIGINT
* **ration\_card\_id** (FK) → RATION\_CARD
* **scheme\_commodity\_id** (FK) → SCHEME\_COMMODITY
* monthly\_quota - DECIMAL(8,2)
* consumed\_quantity - DECIMAL(8,2) DEFAULT 0
* remaining\_quota - DECIMAL(8,2)
* reset\_date - DATE
* created\_at - TIMESTAMP
* updated\_at - TIMESTAMP

**10. WAREHOUSE**

* **warehouse\_id** (PK) - INT
* warehouse\_code - VARCHAR(20) UNIQUE NOT NULL
* warehouse\_name - VARCHAR(100) NOT NULL
* warehouse\_type - ENUM('CENTRAL', 'STATE', 'DISTRICT')
* address\_line1 - VARCHAR(255)
* address\_line2 - VARCHAR(255)
* city - VARCHAR(100)
* state - VARCHAR(100)
* pincode - VARCHAR(10)
* capacity\_in\_tons - DECIMAL(10,2)
* **manager\_user\_id** (FK) → USER
* is\_operational - BOOLEAN DEFAULT TRUE
* created\_at - TIMESTAMP
* updated\_at - TIMESTAMP

**11. STOCK\_ITEM**

* **stock\_item\_id** (PK) - BIGINT
* **warehouse\_id** (FK) → WAREHOUSE
* **commodity\_id** (FK) → COMMODITY
* batch\_number - VARCHAR(50)
* quantity\_in\_stock - DECIMAL(10,2) NOT NULL
* quality\_grade - ENUM('A', 'B', 'C')
* procurement\_date - DATE
* expiry\_date - DATE
* cost\_per\_unit - DECIMAL(8,2)
* **supplier\_id** (FK) → PROCUREMENT\_AGENCY
* last\_updated - TIMESTAMP

**12. PROCUREMENT\_AGENCY**

* **supplier\_id** (PK) - INT
* agency\_name - VARCHAR(100) NOT NULL
* contact\_person - VARCHAR(100)
* phone\_number - VARCHAR(15)
* email - VARCHAR(100)
* address - TEXT
* license\_number - VARCHAR(50)
* is\_approved - BOOLEAN DEFAULT FALSE
* created\_at - TIMESTAMP
* updated\_at - TIMESTAMP

**13. FAIR\_PRICE\_SHOP**

* **fps\_id** (PK) - INT
* fps\_code - VARCHAR(20) UNIQUE NOT NULL
* shop\_name - VARCHAR(100) NOT NULL
* **owner\_user\_id** (FK) → USER
* license\_number - VARCHAR(50)
* address\_line1 - VARCHAR(255)
* address\_line2 - VARCHAR(255)
* city - VARCHAR(100)
* state - VARCHAR(100)
* pincode - VARCHAR(10)
* **assigned\_warehouse\_id** (FK) → WAREHOUSE
* max\_daily\_customers - INT
* operating\_hours - VARCHAR(50)
* is\_operational - BOOLEAN DEFAULT TRUE
* created\_at - TIMESTAMP
* updated\_at - TIMESTAMP

**14. FPS\_STOCK**

* **fps\_stock\_id** (PK) - BIGINT
* **fps\_id** (FK) → FAIR\_PRICE\_SHOP
* **commodity\_id** (FK) → COMMODITY
* current\_stock - DECIMAL(8,2) NOT NULL
* reserved\_stock - DECIMAL(8,2) DEFAULT 0
* last\_restocked\_date - DATE
* minimum\_stock\_level - DECIMAL(8,2)
* last\_updated - TIMESTAMP

**15. TRANSPORT\_ORDER**

* **transport\_order\_id** (PK) - BIGINT
* order\_number - VARCHAR(30) UNIQUE NOT NULL
* **source\_warehouse\_id** (FK) → WAREHOUSE
* **destination\_fps\_id** (FK) → FAIR\_PRICE\_SHOP
* vehicle\_number - VARCHAR(20)
* driver\_name - VARCHAR(100)
* driver\_phone - VARCHAR(15)
* dispatch\_date - DATE
* expected\_delivery\_date - DATE
* actual\_delivery\_date - DATE
* status - ENUM('CREATED', 'DISPATCHED', 'IN\_TRANSIT', 'DELIVERED', 'CANCELLED')
* created\_at - TIMESTAMP
* updated\_at - TIMESTAMP

**16. TRANSPORT\_ITEM**

* **transport\_item\_id** (PK) - BIGINT
* **transport\_order\_id** (FK) → TRANSPORT\_ORDER
* **commodity\_id** (FK) → COMMODITY
* quantity\_dispatched - DECIMAL(8,2)
* quantity\_received - DECIMAL(8,2)
* batch\_number - VARCHAR(50)
* condition\_on\_delivery - ENUM('GOOD', 'DAMAGED', 'PARTIAL\_DAMAGE')

**17. TRANSACTION**

* **transaction\_id** (PK) - BIGINT
* transaction\_number - VARCHAR(30) UNIQUE NOT NULL
* **fps\_id** (FK) → FAIR\_PRICE\_SHOP
* **ration\_card\_id** (FK) → RATION\_CARD
* **served\_by\_user\_id** (FK) → USER (FPS Owner)
* transaction\_date - TIMESTAMP
* transaction\_type - ENUM('PURCHASE', 'REFUND')
* total\_amount - DECIMAL(10,2)
* subsidy\_amount - DECIMAL(10,2)
* net\_payable - DECIMAL(10,2)
* status - ENUM('INITIATED', 'PAYMENT\_PENDING', 'COMPLETED', 'CANCELLED')
* created\_at - TIMESTAMP
* updated\_at - TIMESTAMP

**18. TRANSACTION\_ITEM**

* **transaction\_item\_id** (PK) - BIGINT
* **transaction\_id** (FK) → TRANSACTION
* **commodity\_id** (FK) → COMMODITY
* quantity\_purchased - DECIMAL(8,2)
* unit\_price - DECIMAL(8,2)
* subsidized\_price - DECIMAL(8,2)
* line\_total - DECIMAL(10,2)

**19. PAYMENT\_DETAIL**

* **payment\_id** (PK) - BIGINT
* **transaction\_id** (FK) → TRANSACTION
* payment\_method - ENUM('CASH', 'UPI', 'CARD', 'WALLET')
* payment\_gateway\_id - VARCHAR(100)
* payment\_reference\_number - VARCHAR(100)
* qr\_code\_data - TEXT
* upi\_id - VARCHAR(100)
* payment\_status - ENUM('PENDING', 'SUCCESS', 'FAILED', 'REFUNDED')
* payment\_timestamp - TIMESTAMP
* gateway\_response - JSON
* created\_at - TIMESTAMP

**20. GRIEVANCE**

* **grievance\_id** (PK) - BIGINT
* **citizen\_id** (FK) → CITIZEN
* **fps\_id** (FK) → FAIR\_PRICE\_SHOP (nullable)
* grievance\_type - ENUM('QUALITY\_ISSUE', 'QUANTITY\_SHORTAGE', 'OVERCHARGING', 'UNAVAILABILITY', 'STAFF\_BEHAVIOR', 'OTHER')
* title - VARCHAR(200)
* description - TEXT
* priority - ENUM('LOW', 'MEDIUM', 'HIGH', 'CRITICAL')
* status - ENUM('OPEN', 'IN\_PROGRESS', 'RESOLVED', 'CLOSED')
* **assigned\_to\_user\_id** (FK) → USER (nullable)
* created\_at - TIMESTAMP
* resolved\_at - TIMESTAMP
* resolution\_notes - TEXT

**21. NOTIFICATION\_LOG**

* **notification\_id** (PK) - BIGINT
* **user\_id** (FK) → USER
* notification\_type - ENUM('SMS', 'EMAIL', 'PUSH', 'IN\_APP')
* title - VARCHAR(200)
* message - TEXT
* delivery\_status - ENUM('PENDING', 'SENT', 'DELIVERED', 'FAILED')
* sent\_at - TIMESTAMP
* delivered\_at - TIMESTAMP
* **related\_transaction\_id** (FK) → TRANSACTION (nullable)
* **related\_grievance\_id** (FK) → GRIEVANCE (nullable)
* created\_at - TIMESTAMP

**22. AUDIT\_LOG**

* **audit\_id** (PK) - BIGINT
* **user\_id** (FK) → USER
* table\_name - VARCHAR(50)
* record\_id - BIGINT
* action - ENUM('CREATE', 'UPDATE', 'DELETE')
* old\_values - JSON
* new\_values - JSON
* ip\_address - VARCHAR(45)
* user\_agent - TEXT
* timestamp - TIMESTAMP

**Relationships Summary**

**One-to-One (1:1)**

* USER ↔ CITIZEN
* RATION\_CARD ↔ CITIZEN (Head of Family)
* TRANSACTION ↔ PAYMENT\_DETAIL

**One-to-Many (1:M)**

* ROLE → USER
* RATION\_CARD → FAMILY\_MEMBER
* WAREHOUSE → STOCK\_ITEM
* WAREHOUSE → TRANSPORT\_ORDER (source)
* FAIR\_PRICE\_SHOP → TRANSPORT\_ORDER (destination)
* FAIR\_PRICE\_SHOP → FPS\_STOCK
* FAIR\_PRICE\_SHOP → TRANSACTION
* TRANSPORT\_ORDER → TRANSPORT\_ITEM
* TRANSACTION → TRANSACTION\_ITEM
* CITIZEN → GRIEVANCE
* USER → NOTIFICATION\_LOG
* USER → AUDIT\_LOG

**Many-to-Many (M:M)**

* SCHEME ↔ COMMODITY (via SCHEME\_COMMODITY)
* RATION\_CARD ↔ SCHEME\_COMMODITY (via ENTITLEMENT)

**Key Business Rules & Constraints**

1. **Each Ration Card** must have exactly one Head of Family (Citizen)
2. **Family Members** can only belong to one Ration Card
3. **Monthly Entitlements** reset based on scheme rules
4. **Stock Items** cannot have negative quantities
5. **Transactions** must have corresponding Payment Details
6. **Transport Orders** track commodity movement between warehouses and FPS
7. **FPS Stock** is updated based on Transport deliveries and Transactions
8. **Grievances** can be linked to specific FPS or be general complaints
9. **Audit Logs** maintain complete transaction history for compliance

**Database Indexes (Recommended)**

**Primary Performance Indexes**

* ration\_card\_number (UNIQUE)
* aadhar\_number (UNIQUE)
* transaction\_number (UNIQUE)
* fps\_code (UNIQUE)
* warehouse\_code (UNIQUE)

**Composite Indexes for Query Performance**

* (fps\_id, transaction\_date) - for FPS daily reports
* (ration\_card\_id, transaction\_date) - for citizen transaction history
* (warehouse\_id, commodity\_id) - for inventory queries
* (user\_id, timestamp) - for audit trail queries
* (citizen\_id, grievance\_status) - for grievance management

Use case Diagram, Activity Diagram, Class Diagram, Sequence Diagram, Database Diagram for the same

**Digi-PDS - Complete UML Diagrams Suite**

**1. Use Case Diagram**

DIGI-PDS SYSTEM

┌─────────────────────────────────────────────────────────┐

│ │

│ ┌─────────────────┐ ┌─────────────────────────┐ │

│ │ Verify Identity │◇───│ Verify Entitlement │ │

│ └─────────────────┘ └─────────────────────────┘ │

│ │ │ │

│ ▼ ▼ │

│ ┌─────────────────┐ ┌─────────────────────────┐ │

│ │ Collect Ration │────│ Process Payment │ │

│ └─────────────────┘ └─────────────────────────┘ │

│ │ │ │

│ ▼ ▼ │

│ ┌─────────────────┐ ┌─────────────────────────┐ │

│ │ Update Stock │ │ Generate Receipt │ │

│ └─────────────────┘ └─────────────────────────┘ │

│ │

│ ┌─────────────────┐ ┌─────────────────────────┐ │

│ │ Manage Inventory│ │ Track Shipments │ │

│ └─────────────────┘ └─────────────────────────┘ │

│ │

│ ┌─────────────────┐ ┌─────────────────────────┐ │

│ │ File Grievance │ │ View Transaction History│ │

│ └─────────────────┘ └─────────────────────────┘ │

│ │

│ ┌─────────────────┐ ┌─────────────────────────┐ │

│ │ Generate Reports│ │ Monitor System Health │ │

│ └─────────────────┘ └─────────────────────────┘ │

│ │

└─────────────────────────────────────────────────────────┘

ACTORS:

👤 Citizen -------------- Verify Identity, Collect Ration, File Grievance, View Transaction History

🏪 FPS Owner ------------ Verify Entitlement, Process Payment, Update Stock, Generate Receipt

🏢 Warehouse Manager ---- Manage Inventory, Track Shipments

👨‍💼 Admin Official ------- Generate Reports, Monitor System Health

💳 Payment Gateway ------ Process Payment (External System)

**Use Case Descriptions:**

**Primary Use Cases:**

* **Verify Identity**: Citizen authentication using Ration Card/Aadhar
* **Verify Entitlement**: Check monthly quota and remaining balance
* **Collect Ration**: Main transaction flow for ration distribution
* **Process Payment**: Handle subsidized payment via UPI/QR code
* **Update Stock**: Real-time inventory management at FPS level

**Secondary Use Cases:**

* **Manage Inventory**: Warehouse-level stock operations
* **Track Shipments**: Monitor goods movement between warehouses and FPS
* **File Grievance**: Citizens can report issues and complaints
* **Generate Reports**: Analytics and compliance reports for officials
* **Monitor System Health**: Real-time system monitoring and alerts

**2. Activity Diagram - Ration Collection Process**

START

│

▼

┌─────────────────┐

│ Citizen arrives │

│ at FPS │

└─────────────────┘

│

▼

┌─────────────────┐

│ FPS Owner enters│

│ Ration Card ID │

└─────────────────┘

│

▼

◇─────────────────◇ NO

│ Valid Ration │────→ ┌─────────────────┐

│ Card? │ │ Show Error │

◇─────────────────◇ │ Message │

│ YES └─────────────────┘

▼ │

┌─────────────────┐ │

│ Fetch Citizen │ │

│ Entitlements │ │

└─────────────────┘ │

│ │

▼ │

◇─────────────────◇ NO │

│ Has Remaining │────→──────┘

│ Quota? │

◇─────────────────◇

│ YES

▼

┌─────────────────┐

│ Display │

│ Available Items │

└─────────────────┘

│

▼

┌─────────────────┐

│ FPS Owner │

│ Selects Items │

└─────────────────┘

│

▼

◇─────────────────◇ NO

│ Sufficient │────→ ┌─────────────────┐

│ Stock at FPS? │ │ Request Stock │

◇─────────────────◇ │ from Warehouse │

│ YES └─────────────────┘

▼ │

┌─────────────────┐ │

│ Calculate │ │

│ Bill Amount │ │

└─────────────────┘ │

│ │

▼ │

┌─────────────────┐ │

│ Generate QR │ │

│ Code for Payment│ │

└─────────────────┘ │

│ │

▼ │

◇─────────────────◇ FAILED │

│ Payment │────→──────┘

│ Successful? │

◇─────────────────◇

│ SUCCESS

▼

║═══════════════════════════════════════║

║ PARALLEL ACTIVITIES ║

║ ║

║ ┌─────────────────┐ ┌───────────────┐ ║

║ │ Update FPS │ │ Send SMS │ ║

║ │ Stock Levels │ │ to Citizen │ ║

║ └─────────────────┘ └───────────────┘ ║

║ ║

║ ┌─────────────────┐ ┌───────────────┐ ║

║ │ Record │ │ Update │ ║

║ │ Transaction │ │ Entitlement │ ║

║ └─────────────────┘ └───────────────┘ ║

║ ║

║═══════════════════════════════════════║

│

▼

┌─────────────────┐

│ Print Receipt │

└─────────────────┘

│

▼

END

**3. Class Diagram - Core Domain Classes**

┌─────────────────┐

│ BaseEntity │

│ <<abstract>> │

├─────────────────┤

│ + id: Long │

│ + createdAt: TS │

│ + updatedAt: TS │

└─────────────────┘

▲

│ (extends)

┌───────────────────────────┼───────────────────────────┐

│ │ │

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐

│ User │ │ Citizen │ │ RationCard │

├─────────────────┤ ├─────────────────┤ ├─────────────────┤

│ - username: STR │◇────────│ + firstName: STR│◆───────│ + cardNumber:STR│

│ - password: STR │ 1 1 │ + lastName: STR │ 1 1 │ + cardType: ENUM│

│ - email: STR │ │ + dateOfBirth:DT│ │ + issueDate: DT │

│ - phone: STR │ │ + aadharNo: STR │ │ + isActive: BOOL│

│ - isActive:BOOL │ │ + address: STR │ └─────────────────┘

│ + role: Role │ └─────────────────┘ │

├─────────────────┤ │ │ 1

│ + authenticate()│ │ 1 │

│ + authorize() │ │ │

└─────────────────┘ │ │ \*

│ \* │ ▼

│ │ ┌─────────────────┐

▼ 1 │ │ FamilyMember │

┌─────────────────┐ │ ├─────────────────┤

│ Role │ │ │ + relationship: │

├─────────────────┤ │ │ ENUM │

│ + roleName: STR │ │ │ + isPrimary:BOOL│

│ + permissions: │ │ └─────────────────┘

│ JSON │ │

└─────────────────┘ │

│ \*

▼

┌─────────────────┐

│ Transaction │

├─────────────────┤

│ + transNo: STR │◆─────┐ 1

│ + totalAmt: DEC │ │

│ + netPayable:DEC│ │

│ + status: ENUM │ │ 1

│ + txnDate: TS │ ▼

└─────────────────┘ ┌─────────────────┐

│ │ PaymentDetail │

│ 1 ├─────────────────┤

│ │ + paymentMethod:│

│ │ ENUM │

│ │ + gatewayRef:STR│

│ \* │ + qrCodeData:STR│

▼ │ + paymentStatus:│

┌─────────────────┐ │ ENUM │

│TransactionItem │ └─────────────────┘

├─────────────────┤

│ + quantity: DEC │

│ + unitPrice:DEC │◆─────┐

│ + lineTotal:DEC │ │ \*

└─────────────────┘ │

│ 1

▼

┌─────────────────┐

│ Commodity │

├─────────────────┤

│ + name: STR │

│ + code: STR │

│ + unit: STR │

│ + category:ENUM │

└─────────────────┘

▲

│ \*

│

┌─────────────────┐

│SchemeCommodity │

│ <<join table>> │

├─────────────────┤

│ + qtyPerPerson: │

│ DEC │

│ + subsidizedRate│

│ DEC │

└─────────────────┘

│ \*

│

▼ 1

┌─────────────────┐

│ Scheme │

├─────────────────┤

│ + schemeName:STR│

│ + schemeType: │

│ ENUM │

│ + startDate: DT │

│ + isActive:BOOL │

└─────────────────┘

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐

│ Warehouse │ │ FairPriceShop │ │ StockItem │

├─────────────────┤ ├─────────────────┤ ├─────────────────┤

│ + warehouseCode:│◆────────│ + fpsCode: STR │ │ + batchNo: STR │

│ STR │ 1 \* │ + shopName: STR │◆────────│ + quantity: DEC │

│ + name: STR │ │ + licenseNo:STR │ 1 \* │ + qualityGrade: │

│ + type: ENUM │ │ + address: STR │ │ ENUM │

│ + capacity: DEC │ │ + isOperational:│ │ + expiryDate:DT │

│ + address: STR │ │ BOOL │ └─────────────────┘

│ + isOperational:│ └─────────────────┘

│ BOOL │

└─────────────────┘

┌─────────────────┐

│ Grievance │

├─────────────────┤

│ + grievanceType:│◇───────┐

│ ENUM │ \* 1 │

│ + title: STR │ │

│ + description: │ ▼

│ TEXT │ ┌─────────────────┐

│ + priority:ENUM │ │ Citizen │

│ + status: ENUM │ │ (from above) │

│ + createdAt: TS │ └─────────────────┘

└─────────────────┘

**4. Sequence Diagram - Ration Collection Flow**

**Citizen FPS\_Owner API\_Gateway Auth\_Service FPS\_Service Citizen\_Service Inventory\_Service Payment\_Service Payment\_Gateway Notification\_Service**

**│ │ │ │ │ │ │ │ │ │**

**│ 1. Visits │ │ │ │ │ │ │ │ │**

**│ FPS Shop │ │ │ │ │ │ │ │ │**

**│───────────▶│ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ 2. FPS Owner │ │ │ │ │ │ │ │**

**│ │ Login │ │ │ │ │ │ │ │**

**│ │─────────────▶│ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ 3. Validate │ │ │ │ │ │ │**

**│ │ │ JWT Token │ │ │ │ │ │ │**

**│ │ │──────────────▶│ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ 4. Token │ │ │ │ │ │ │**

**│ │ │ Valid │ │ │ │ │ │ │**

**│ │ │◀──────────────│ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ 5. Login │ │ │ │ │ │ │ │**

**│ │ Success │ │ │ │ │ │ │ │**

**│ │◀─────────────│ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ 6. Enter │ │ │ │ │ │ │ │ │**

**│ Ration │ │ │ │ │ │ │ │ │**

**│ Card ID │ │ │ │ │ │ │ │ │**

**│───────────▶│ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ 7. Verify │ │ │ │ │ │ │ │**

**│ │ Entitlement │ │ │ │ │ │ │ │**

**│ │ Request │ │ │ │ │ │ │ │**

**│ │─────────────▶│ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ 8. Route to │ │ │ │ │ │ │**

**│ │ │ FPS Service │ │ │ │ │ │ │**

**│ │ │──────────────────────────────▶│ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ 9. Get Citizen │ │ │ │ │**

**│ │ │ │ │ & Entitlement │ │ │ │ │**

**│ │ │ │ │ Details │ │ │ │ │**

**│ │ │ │ │────────────────▶│ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ 10. Query DB for │ │ │ │**

**│ │ │ │ │ │ Citizen Details │ │ │ │**

**│ │ │ │ │ │ & Monthly Quota │ │ │ │**

**│ │ │ │ │ │ ──────────────── │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ 11. Citizen & │ │ │ │ │**

**│ │ │ │ │ Entitlement │ │ │ │ │**

**│ │ │ │ │ Data │ │ │ │ │**

**│ │ │ │ │◀────────────────│ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ 12. Check FPS │ │ │ │ │**

**│ │ │ │ │ Stock Levels │ │ │ │ │**

**│ │ │ │ │──────────────────────────────────────▶│ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ 13. Query FPS │ │ │ │**

**│ │ │ │ │ │ Stock DB │ │ │ │**

**│ │ │ │ │ │ ──────────────── │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ 14. Available │ │ │ │ │**

**│ │ │ │ │ Stock Data │ │ │ │ │**

**│ │ │ │ │◀──────────────────────────────────────│ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ 15. Display │ │ │ │ │ │ │**

**│ │ │ Available │ │ │ │ │ │ │**

**│ │ │ Items & │ │ │ │ │ │ │**

**│ │ │ Quantities │ │ │ │ │ │ │**

**│ │◀─────────────│ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ 16. Select │ │ │ │ │ │ │ │ │**

**│ Items & │ │ │ │ │ │ │ │ │**

**│ Quantities │ │ │ │ │ │ │ │ │**

**│───────────▶│ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ 17. Create │ │ │ │ │ │ │ │**

**│ │ Transaction │ │ │ │ │ │ │ │**

**│ │ Request │ │ │ │ │ │ │ │**

**│ │─────────────▶│ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ 18. Route to │ │ │ │ │ │ │**

**│ │ │ FPS Service │ │ │ │ │ │ │**

**│ │ │──────────────────────────────▶│ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ 19. Calculate │ │ │ │ │**

**│ │ │ │ │ Bill Amount │ │ │ │ │**

**│ │ │ │ │ (Market Rate - │ │ │ │ │**

**│ │ │ │ │ Subsidy) │ │ │ │ │**

**│ │ │ │ │ ──────────── │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ 20. Generate │ │ │ │ │**

**│ │ │ │ │ Payment Request │ │ │ │ │**

**│ │ │ │ │──────────────────────────────────────────────────────────▶│ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ 21. Create QR │ │ │**

**│ │ │ │ │ │ │ Code & UPI ID │ │ │**

**│ │ │ │ │ │ │ Request │ │ │**

**│ │ │ │ │ │ │─────────────────▶│ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ 22. Generate │ │**

**│ │ │ │ │ │ │ │ Dynamic QR & │ │**

**│ │ │ │ │ │ │ │ UPI ID │ │**

**│ │ │ │ │ │ │ │ ────────────── │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ 23. QR Code & │ │ │**

**│ │ │ │ │ │ │ UPI Details │ │ │**

**│ │ │ │ │ │ │◀─────────────────│ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ 24. Payment │ │ │ │ │**

**│ │ │ │ │ Details │ │ │ │ │**

**│ │ │ │ │◀──────────────────────────────────────────────────────────│ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ 25. Display │ │ │ │ │ │ │**

**│ │ │ QR Code & │ │ │ │ │ │ │**

**│ │ │ Payment │ │ │ │ │ │ │**

**│ │ │ Amount │ │ │ │ │ │ │**

**│ │◀─────────────│ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ 26. Scan │ │ │ │ │ │ │ │ │**

**│ QR Code & │ │ │ │ │ │ │ │ │**

**│ Make │ │ │ │ │ │ │ │ │**

**│ Payment │ │ │ │ │ │ │ │ │**

**│ (via UPI) │ │ │ │ │ │ │ ◀─────────────────│ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ 27. Payment │ │**

**│ │ │ │ │ │ │ │ Webhook/ │ │**

**│ │ │ │ │ │ │ │ Callback │ │**

**│ │ │ │ │ │ │◀─────────────────│ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ 28. Payment │ │ │ │ │**

**│ │ │ │ │ Success │ │ │ │ │**

**│ │ │ │ │ Confirmation │ │ │ │ │**

**│ │ │ │ │◀──────────────────────────────────────────────────────────│ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ 29. Update │ │ │ │ │**

**│ │ │ │ │ Transaction │ │ │ │ │**

**│ │ │ │ │ Status to │ │ │ │ │**

**│ │ │ │ │ COMPLETED │ │ │ │ │**

**│ │ │ │ │ ──────────── │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ 30. Publish │ │ │ │ │**

**│ │ │ │ │ "Transaction │ │ │ │ │**

**│ │ │ │ │ Completed" │ │ │ │ │**

**│ │ │ │ │ Event (Async) │ │ │ │ │**

**│ │ │ │ │ ════════════════════════════════════════════════════════════════════════════════════════════════════▶│**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ 31. Event │**

**│ │ │ │ │ │ │ │ │ Distribution │**

**│ │ │ │ │ │ │ │ │ to Subscribers │**

**│ │ │ │ │ │ │ │ │ ────────────── │**

**│ │ │ │ │ │ │ │ │ │**

**║════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════║**

**║ PARALLEL ASYNC PROCESSING (Event-Driven) ║**

**║════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════║**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ 32a. Update FPS │ │ │**

**│ │ │ │ │ │ │ Stock Levels │ │ │**

**│ │ │ │ │ │ │ (Async Event) │ │ │**

**│ │ │ │ │ │ │◀═════════════════════════════════════════════════════════════════════│**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ 33a. Deduct │ │ │**

**│ │ │ │ │ │ │ Sold Quantities │ │ │**

**│ │ │ │ │ │ │ from FPS Stock │ │ │**

**│ │ │ │ │ │ │ ────────────── │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ 32b. Update │ │ │ │**

**│ │ │ │ │ │ Citizen Monthly │ │ │ │**

**│ │ │ │ │ │ Entitlement │ │ │ │**

**│ │ │ │ │ │ (Async Event) │ │ │ │**

**│ │ │ │ │ │◀═══════════════════════════════════════════════════════════════════════════════════════════│**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ 33b. Reduce │ │ │ │**

**│ │ │ │ │ │ Remaining Quota │ │ │ │**

**│ │ │ │ │ │ by Purchased Qty │ │ │ │**

**│ │ │ │ │ │ ────────────── │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ 32c. Send SMS │**

**│ │ │ │ │ │ │ │ │ Notification │**

**│ │ │ │ │ │ │ │ │ (Async Event) │**

**│ │ │ │ │ │ │ │ │◀═════════════════════│**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ 33c. Send Success │**

**│ │ │ │ │ │ │ │ │ SMS to Citizen │**

**│ │ │ │ │ │ │ │ │ ────────────── │**

**║════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════║**

**│ │ │ │ │ │ │ │ │ │**

**│ │ │ 34. Transaction │ │ │ │ │ │**

**│ │ │ Successful │ │ │ │ │ │ │**

**│ │ │ Response │ │ │ │ │ │ │**

**│ │◀─────────────│ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ │ 35. Print │ │ │ │ │ │ │ │**

**│ │ Receipt & │ │ │ │ │ │ │ │**

**│ │ Hand Over │ │ │ │ │ │ │ │**

**│ │ Rations │ │ │ │ │ │ │ │**

**│◀───────────│ │ │ │ │ │ │ │ │**

**│ │ │ │ │ │ │ │ │ │**

**│ 36. SMS │ │ │ │ │ │ │ │ │**

**│ Confirmation │ │ │ │ │ │ │ │**

**│ Received │ │ │ │ │ │ │ │ │**

**│◀═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════│**

**│ │ │ │ │ │ │ │ │ │**

**Legend:**

**──────▶ Synchronous HTTP Request/Response**

**════▶ Asynchronous Event/Message via Message Queue**

**──── Internal Processing**

**◀────── Response/Reply**

**Key Process Flow Summary:**

**Phase 1: Authentication & Setup (Steps 1-5)**

* Citizen visits FPS, Owner logs in and gets authenticated

**Phase 2: Entitlement Verification (Steps 6-14)**

* Citizen provides Ration Card ID
* System verifies citizen identity and checks monthly quotas
* FPS stock levels validated against requested items

**Phase 3: Transaction Creation (Steps 15-19)**

* Available items displayed to citizen
* Citizen selects items and quantities
* Bill calculated with government subsidy applied

**Phase 4: Payment Processing (Steps 20-28)**

* Dynamic QR code generated for exact payment amount
* Citizen pays via UPI/mobile payment
* Payment confirmed via gateway webhook

**Phase 5: Event-Driven Updates (Steps 29-33)**

* Transaction Completed event triggers parallel processing:
  + **Inventory Service**: Updates FPS stock levels
  + **Citizen Service**: Reduces monthly entitlement quota
  + **Notification Service**: Sends SMS confirmation

**Phase 6: Completion (Steps 34-36)**

* Receipt printed, rations handed over
* SMS confirmation received by citizen

**Architecture Benefits:**

* **Microservices**: Each service handles specific domain logic
* **Event-Driven**: Parallel processing for better performance
* **Security**: JWT-based authentication throughout
* **Real-time**: Immediate payment confirmation and stock updates

**5. Database Schema Diagram**

**Schema Organization:**

**🔐 Core User Management (3 tables):**

* ROLE, USER, CITIZEN with proper authentication flow

**📋 Entitlement Management (6 tables):**

* RATION\_CARD, FAMILY\_MEMBER, SCHEME, COMMODITY, SCHEME\_COMMODITY (junction), ENTITLEMENT

**🏢 Supply Chain Management (7 tables):**

* PROCUREMENT\_AGENCY, WAREHOUSE, STOCK\_ITEM, FAIR\_PRICE\_SHOP, FPS\_STOCK, TRANSPORT\_ORDER, TRANSPORT\_ITEM

**💳 Transaction Processing (3 tables):**

* TRANSACTION, TRANSACTION\_ITEM, PAYMENT\_DETAIL

**📞 Support & Monitoring (3 tables):**

* GRIEVANCE, NOTIFICATION\_LOG, AUDIT\_LOG

**Key Database Features:**

**✅ Proper Normalization:**

* 3NF compliance with appropriate junction tables
* Eliminates data redundancy while maintaining performance

**🔗 Comprehensive Relationships:**

* 22 tables with proper foreign key constraints
* Mix of 1:1, 1:M, and M:M relationships via junction tables

**⚡ Performance Optimization:**

* Strategic indexes on frequently queried columns
* Composite indexes for complex queries
* Unique constraints on business identifiers

**🛡️ Data Integrity:**

* Foreign key constraints for referential integrity
* Check constraints for valid enum values
* Audit trail for compliance and debugging

This schema supports all your microservices architecture requirements and handles the complex business rules of the PDS ecosystem efficiently!

**Digi-PDS - Database Schema Diagram**

**Complete Database Schema with Relationships**

═════════════════════════════════════════════════════════

**CORE USER MANAGEMENT**

═════════════════════════════════════════════════════════

┌─────────────────────────────────┐ ┌─────────────────────────────────┐

│ ROLE │ │ USER │

├─────────────────────────────────| 1 |────────────────────────────────|

│ PK role\_id (INT) │────▶ │ PK user\_id (BIGINT) │

│ role\_name (VARCHAR 50) UK │ \* │ username (VARCHAR 50) UK │

│ description (TEXT) │ │ password\_hash (VARCHAR 255) │

│ permissions (JSON) │ │ email (VARCHAR 100) UK │

│ created\_at (TIMESTAMP) │ │ phone\_number (VARCHAR 15) │

└─────────────────────────────────┘ │ is\_active (BOOLEAN) │

│ FK role\_id → ROLE │

│ created\_at (TIMESTAMP) │

│ updated\_at (TIMESTAMP) │

└─────────────────────────────────┘

│ 1

│

▼ 1

┌─────────────────────────────────┐

│ CITIZEN │

├─────────────────────────────────┤

│ PK citizen\_id (BIGINT) │

│ FK user\_id → USER │

│ first\_name (VARCHAR 50) │

│ last\_name (VARCHAR 50) │

│ date\_of\_birth (DATE) │

│ gender (ENUM) │

│ address\_line1 (VARCHAR 255) │

│ city (VARCHAR 100) │

│ state (VARCHAR 100) │

│ pincode (VARCHAR 10) │

│ aadhar\_number (VARCHAR 12)UK │

│ is\_head\_of\_family (BOOLEAN) │

│ created\_at (TIMESTAMP) │

│ updated\_at (TIMESTAMP) │

└─────────────────────────────────┘

│ 1

│

▼ 1

═════════════════════════════════════════════════════════

**ENTITLEMENT MANAGEMENT**

═════════════════════════════════════════════════════════

┌─────────────────────────────────┐ ┌─────────────────────────────────┐

│ RATION\_CARD │ 1 │ FAMILY\_MEMBER │

├─────────────────────────────────┤────▶ ├─────────────────────────────────|

│ PK ration\_card\_id (BIGINT) │ \* │ PK family\_member\_id (BIGINT) │

│ ration\_card\_number (V20) UK │ │ FK ration\_card\_id → RATION\_CARD │

│ FK citizen\_id → CITIZEN (HOF) │ │ FK citizen\_id → CITIZEN │

│ card\_type (ENUM APL/BPL/AAY) │ │ relationship (ENUM) │

│ issue\_date (DATE) │ │ is\_primary (BOOLEAN) │

│ expiry\_date (DATE) │ │ added\_date (DATE) │

│ is\_active (BOOLEAN) │ │ created\_at (TIMESTAMP) │

│ monthly\_income (DECIMAL) │ └─────────────────────────────────┘

│ created\_at (TIMESTAMP) │

│ updated\_at (TIMESTAMP) │

└─────────────────────────────────┘

│ 1

│

▼ \*

┌─────────────────────────────────┐

│ ENTITLEMENT │

├─────────────────────────────────┤

│ PK entitlement\_id (BIGINT) │

│ FK ration\_card\_id → RATION\_CARD │

│ FK scheme\_commodity\_id → SC │

│ monthly\_quota (DECIMAL) │

│ consumed\_quantity (DECIMAL) │

│ remaining\_quota (DECIMAL) │

│ reset\_date (DATE) │

│ created\_at (TIMESTAMP) │

│ updated\_at (TIMESTAMP) │

└─────────────────────────────────┘

▲ \*

│

│ 1

┌─────────────────────────────────┐

│ SCHEME\_COMMODITY │ ← Junction Table for M:M

|─────────────────────────────────|

│ PK scheme\_commodity\_id (BIGINT) │

│ FK scheme\_id → SCHEME │

│ FK commodity\_id → COMMODITY │

│ quantity\_per\_person (DEC) │

│ subsidized\_rate (DECIMAL) │

│ market\_rate (DECIMAL) │

│ is\_active (BOOLEAN) │

│ effective\_from (DATE) │

│ effective\_to (DATE) │

└─────────────────────────────────┘

▲ \* ▲ \*

│ │

│ 1 │ 1

┌─────────────────┐ ┌─────────────────┐

│ SCHEME │ │ COMMODITY │

├─────────────────┤ ├─────────────────┤

│PK scheme\_id(INT)│ │PK commodity\_id │

│ scheme\_name │ │ (INT) │

│ (VARCHAR 100) │ │ commodity\_name │

│ description │ │ (VARCHAR 50) │

│ (TEXT) │ │ commodity\_code │

│ scheme\_type │ │ (VARCHAR 10)UK │

│ (ENUM) │ │ unit\_of\_measure│

│ start\_date │ │ (VARCHAR 20) │

│ (DATE) │ │ category (ENUM)│

│ end\_date(DATE) │ │ nutritional\_ │

│ is\_active │ │ info (JSON) │

│ (BOOLEAN) │ │ is\_active │

│ eligibility\_ │ │ (BOOLEAN) │

│ criteria(JSON) │ │ created\_at(TS) │

│ created\_at(TS) │ └─────────────────┘

│ updated\_at(TS) │

└─────────────────┘

═════════════════════════════════════════════════════════

**SUPPLY CHAIN MANAGEMENT**

═════════════════════════════════════════════════════════

┌─────────────────────────────────┐ ┌─────────────────────────────────┐

│ PROCUREMENT\_AGENCY │ 1 │ STOCK\_ITEM │

├─────────────────────────────────┤────▶├─────────────────────────────────┤

│ PK supplier\_id (INT) │ \* │ PK stock\_item\_id (BIGINT) │

│ agency\_name (VARCHAR 100) │ │ FK warehouse\_id → WAREHOUSE │

│ contact\_person (VARCHAR 100)│ │ FK commodity\_id → COMMODITY │

│ phone\_number (VARCHAR 15) │ │ FK supplier\_id → PROCUREMENT\_A │

│ email (VARCHAR 100) │ │ batch\_number (VARCHAR 50) │

│ address (TEXT) │ │ quantity\_in\_stock (DECIMAL) │

│ license\_number (VARCHAR 50) │ │ quality\_grade (ENUM A/B/C) │

│ is\_approved (BOOLEAN) │ │ procurement\_date (DATE) │

│ created\_at (TIMESTAMP) │ │ expiry\_date (DATE) │

│ updated\_at (TIMESTAMP) │ │ cost\_per\_unit (DECIMAL) │

└─────────────────────────────────┘ │ last\_updated (TIMESTAMP) │

└─────────────────────────────────┘

▲ \*

│

│ 1

┌─────────────────────────────────┐ ┌─────────────────────────────────┐

│ WAREHOUSE │ 1 │ FAIR\_PRICE\_SHOP │

├─────────────────────────────────┤────▶├─────────────────────────────────┤

│ PK warehouse\_id (INT) │ \* │ PK fps\_id (INT) │

│ warehouse\_code (VARCHAR20)UK│ │ fps\_code (VARCHAR 20) UK │

│ warehouse\_name (VARCHAR100) │ │ shop\_name (VARCHAR 100) │

│ warehouse\_type (ENUM) │ │ FK owner\_user\_id → USER │

│ address\_line1 (VARCHAR255) │ │ license\_number (VARCHAR 50) │

│ city (VARCHAR 100) │ │ address\_line1 (VARCHAR 255) │

│ state (VARCHAR 100) │ │ city (VARCHAR 100) │

│ pincode (VARCHAR 10) │ │ state (VARCHAR 100) │

│ capacity\_in\_tons (DECIMAL) │ │ pincode (VARCHAR 10) │

│ FK manager\_user\_id → USER │ │ FK assigned\_warehouse\_id → WH │

│ is\_operational (BOOLEAN) │ │ max\_daily\_customers (INT) │

│ created\_at (TIMESTAMP) │ │ operating\_hours (VARCHAR50) │

│ updated\_at (TIMESTAMP) │ │ is\_operational (BOOLEAN) │

└─────────────────────────────────┘ │ created\_at (TIMESTAMP) │

│ 1 │ updated\_at (TIMESTAMP) │

│ └─────────────────────────────────┘

▼ \* │ 1

┌─────────────────────────────────┐ │

│ TRANSPORT\_ORDER │ ▼ \*

├─────────────────────────────────┤ ┌─────────────────────────────────┐

│ PK transport\_order\_id (BIGINT) │ │ FPS\_STOCK │

│ order\_number (VARCHAR 30)UK │ ├─────────────────────────────────┤

│ FK source\_warehouse\_id → WH │ │ PK fps\_stock\_id (BIGINT) │

│ FK destination\_fps\_id → FPS │ │ FK fps\_id → FAIR\_PRICE\_SHOP │

│ vehicle\_number (VARCHAR 20) │ │ FK commodity\_id → COMMODITY │

│ driver\_name (VARCHAR 100) │ │ current\_stock (DECIMAL) │

│ driver\_phone (VARCHAR 15) │ │ reserved\_stock (DECIMAL) │

│ dispatch\_date (DATE) │ │ last\_restocked\_date (DATE) │

│ expected\_delivery\_date(DATE)│ │ minimum\_stock\_level (DEC) │

│ actual\_delivery\_date (DATE) │ │ last\_updated (TIMESTAMP) │

│ status (ENUM) │ └─────────────────────────────────┘

│ created\_at (TIMESTAMP) │

│ updated\_at (TIMESTAMP) │

└─────────────────────────────────┘

│ 1

│

▼ \*

┌─────────────────────────────────┐

│ TRANSPORT\_ITEM │

├─────────────────────────────────┤

│ PK transport\_item\_id (BIGINT) │

│ FK transport\_order\_id → TO │

│ FK commodity\_id → COMMODITY │

│ quantity\_dispatched (DEC) │

│ quantity\_received (DECIMAL) │

│ batch\_number (VARCHAR 50) │

│ condition\_on\_delivery(ENUM) │

└─────────────────────────────────┘

═════════════════════════════════════════════════════════

**TRANSACTION PROCESSING**

═════════════════════════════════════════════════════════

┌─────────────────────────────────┐ ┌─────────────────────────────────┐

│ TRANSACTION │ 1 │ PAYMENT\_DETAIL │

├─────────────────────────────────┤────▶├─────────────────────────────────┤

│ PK transaction\_id (BIGINT) │ 1 │ PK payment\_id (BIGINT) │

│ transaction\_number (V30) UK │ │ FK transaction\_id → TRANSACTION│

│ FK fps\_id → FAIR\_PRICE\_SHOP │ │ payment\_method (ENUM) │

│ FK ration\_card\_id → RATION\_CARD│ │ payment\_gateway\_id (V100) │

│ FK served\_by\_user\_id → USER │ │ payment\_reference\_no (V100) │

│ transaction\_date (TIMESTAMP)│ │ qr\_code\_data (TEXT) │

│ transaction\_type (ENUM) │ │ upi\_id (VARCHAR 100) │

│ total\_amount (DECIMAL) │ │ payment\_status (ENUM) │

│ subsidy\_amount (DECIMAL) │ │ payment\_timestamp (TS) │

│ net\_payable (DECIMAL) │ │ gateway\_response (JSON) │

│ status (ENUM) │ │ created\_at (TIMESTAMP) │

│ created\_at (TIMESTAMP) │ └─────────────────────────────────┘

│ updated\_at (TIMESTAMP) │

└─────────────────────────────────┘

│ 1

│

▼ \*

┌─────────────────────────────────┐

│ TRANSACTION\_ITEM │

├─────────────────────────────────┤

│ PK transaction\_item\_id (BIGINT)│

│ FK transaction\_id → TRANSACTION│

│ FK commodity\_id → COMMODITY │

│ quantity\_purchased (DEC) │

│ unit\_price (DECIMAL) │

│ subsidized\_price (DECIMAL) │

│ line\_total (DECIMAL) │

└─────────────────────────────────┘

═════════════════════════════════════════════════════════

**SUPPORT & MONITORING SYSTEM**

═════════════════════════════════════════════════════════

┌─────────────────────────────────┐ ┌─────────────────────────────────┐

│ GRIEVANCE │ │ NOTIFICATION\_LOG │

├─────────────────────────────────┤ ├─────────────────────────────────┤

│ PK grievance\_id (BIGINT) │ │ PK notification\_id (BIGINT) │

│ FK citizen\_id → CITIZEN │ │ FK user\_id → USER │

│ FK fps\_id → FAIR\_PRICE\_SHOP │ │ notification\_type (ENUM) │

│ grievance\_type (ENUM) │ │ title (VARCHAR 200) │

│ title (VARCHAR 200) │ │ message (TEXT) │

│ description (TEXT) │ │ delivery\_status (ENUM) │

│ priority (ENUM) │ │ sent\_at (TIMESTAMP) │

│ status (ENUM) │ │ delivered\_at (TIMESTAMP) │

│ FK assigned\_to\_user\_id → USER │ │ FK related\_transaction\_id → TX │

│ created\_at (TIMESTAMP) │ │ FK related\_grievance\_id → GR │

│ resolved\_at (TIMESTAMP) │ │ created\_at (TIMESTAMP) │

│ resolution\_notes (TEXT) │ └─────────────────────────────────┘

└─────────────────────────────────┘

┌─────────────────────────────────┐

│ AUDIT\_LOG │

├─────────────────────────────────┤

│ PK audit\_id (BIGINT) │

│ FK user\_id → USER │

│ table\_name (VARCHAR 50) │

│ record\_id (BIGINT) │

│ action (ENUM) │

│ old\_values (JSON) │

│ new\_values (JSON) │

│ ip\_address (VARCHAR 45) │

│ user\_agent (TEXT) │

│ timestamp (TIMESTAMP) │

└─────────────────────────────────┘

═════════════════════════════════════════════════════════

**RELATIONSHIP SUMMARY**

═════════════════════════════════════════════════════════

/\*

ONE-TO-ONE RELATIONSHIPS (1:1):

- USER ←→ CITIZEN

- RATION\_CARD ←→ CITIZEN (Head of Family)

- TRANSACTION ←→ PAYMENT\_DETAIL

ONE-TO-MANY RELATIONSHIPS (1:M):

- ROLE → USER (1:\*)

- CITIZEN → RATION\_CARD (1:1)

- RATION\_CARD → FAMILY\_MEMBER (1:\*)

- RATION\_CARD → ENTITLEMENT (1:\*)

- RATION\_CARD → TRANSACTION (1:\*)

- WAREHOUSE → STOCK\_ITEM (1:\*)

- WAREHOUSE → TRANSPORT\_ORDER (1:\*) [as source]

- FAIR\_PRICE\_SHOP → TRANSPORT\_ORDER (1:\*) [as destination]

- FAIR\_PRICE\_SHOP → FPS\_STOCK (1:\*)

- FAIR\_PRICE\_SHOP → TRANSACTION (1:\*)

- TRANSPORT\_ORDER → TRANSPORT\_ITEM (1:\*)

- TRANSACTION → TRANSACTION\_ITEM (1:\*)

- CITIZEN → GRIEVANCE (1:\*)

- USER → NOTIFICATION\_LOG (1:\*)

- USER → AUDIT\_LOG (1:\*)

- PROCUREMENT\_AGENCY → STOCK\_ITEM (1:\*)

MANY-TO-MANY RELATIONSHIPS (M:M):

- SCHEME ←→ COMMODITY (via SCHEME\_COMMODITY junction table)

- RATION\_CARD ←→ SCHEME\_COMMODITY (via ENTITLEMENT table)

KEY CONSTRAINTS:

1. Unique constraints on business identifiers (ration\_card\_number, aadhar\_number, etc.)

2. Foreign key constraints maintain referential integrity

3. Check constraints ensure valid enum values

4. Not null constraints on essential fields

5. Composite indexes for performance optimization

\*/

═════════════════════════════════════════════════════════

**INDEXES FOR PERFORMANCE**

═════════════════════════════════════════════════════════

/\*

RECOMMENDED INDEXES:

Primary Performance Indexes:

- CREATE UNIQUE INDEX idx\_ration\_card\_number ON ration\_card(ration\_card\_number);

- CREATE UNIQUE INDEX idx\_aadhar\_number ON citizen(aadhar\_number);

- CREATE UNIQUE INDEX idx\_transaction\_number ON transaction(transaction\_number);

- CREATE UNIQUE INDEX idx\_fps\_code ON fair\_price\_shop(fps\_code);

- CREATE UNIQUE INDEX idx\_warehouse\_code ON warehouse(warehouse\_code);

Composite Indexes for Query Performance:

- CREATE INDEX idx\_fps\_transactions ON transaction(fps\_id, transaction\_date);

- CREATE INDEX idx\_citizen\_transactions ON transaction(ration\_card\_id, transaction\_date);

- CREATE INDEX idx\_warehouse\_inventory ON stock\_item(warehouse\_id, commodity\_id);

- CREATE INDEX idx\_user\_audit\_trail ON audit\_log(user\_id, timestamp);

- CREATE INDEX idx\_citizen\_grievances ON grievance(citizen\_id, status);

- CREATE INDEX idx\_entitlement\_lookup ON entitlement(ration\_card\_id, scheme\_commodity\_id);

- CREATE INDEX idx\_fps\_stock\_lookup ON fps\_stock(fps\_id, commodity\_id);

\*/