E-Ration PDS System - Complete Workflow with Sample Data

This document demonstrates the complete workflow of the E-Ration Public Distribution System with actual sample data showing how the system works from user creation through ration distribution.

Table of Contents

System Overview

The E-Ration PDS system manages ration distribution through dealers to eligible citizens. The system tracks:

- **Users** with role-based access (Admin, Dealer, Citizen)
- **Dealer Profiles** with shop information
- **Citizen Profiles** with ration card details
- **Products** available for distribution
- **Inventory** maintained by each dealer separately
- **Distribution** records tracking who got what from which dealer

---

Step-by-Step Workflow

Step 1: Users Table - Initial Setup

The `users` table stores authentication and role information for all system users.

id	username	email	password	role	aadhaar_ref	phone	created_at
1	admin	admin@pds.gov.in	$2a$10$...	ADMIN	123456789012	9876543210	2024-01-01 10:00:00
2	ram_dealer	ram@dealer.com	$2a$10$...	DEALER	234567890123	9876543211	2024-01-02 09:00:00
3	sita_dealer	sita@dealer.com	$2a$10$...	DEALER	345678901234	9876543212	2024-01-02 09:30:00
4	john_citizen	john@gmail.com	$2a$10$...	CITIZEN	456789012345	9876543213	2024-01-03 10:00:00
5	priya_citizen	priya@gmail.com	$2a$10$...	CITIZEN	567890123456	9876543214	2024-01-03 11:00:00
6	amit_citizen	amit@gmail.com	$2a$10$...	CITIZEN	678901234567	9876543215	2024-01-03 12:00:00

Key Points:

- 1 Admin user for system management
- 2 Dealer users (Ram and Sita) who will operate shops
- 3 Citizen users who will receive rations
- Each user has a unique username, email, and role

Step 2: Dealer Profiles Table

The `dealer_profiles` table stores shop information for dealers.

id	user_id	shop_name	owner_name	shop_license	region	active	contact_number	address
1	2	Ram General Store	Ramesh Kumar	LIC/2024/001	Zone-A	true	9876543211	Shop No. 12, Market Road, Mumbai
2	3	Sita Provisions	Sita Devi	LIC/2024/002	Zone-B	true	9876543212	Plot 45, Gandhi Nagar, Mumbai

Key Points:

- Each dealer profile is linked to a user via `user_id`
- Each dealer has a unique `shop_license`
- Dealers operate in different regions (Zone-A, Zone-B)
- Both dealers are active and ready to distribute

Step 3: Citizen Profiles Table

The `citizen_profiles` table stores citizen information and assigns them to specific dealers.

id	user_id	name	ration_card_number	family_size	category	dealer_id	address	monthly_income
1	4	John Doe	RC2024001	4	BPL	1	23, Palm Street, Mumbai	8000.00
2	5	Priya Sharma	RC2024002	3	APL	1	67, Lake View, Mumbai	15000.00
3	6	Amit Patel	RC2024003	5	BPL	2	89, Garden Colony, Mumbai	7500.00

Key Points:

- Each citizen profile is linked to a user via `user_id`
- Each citizen has a unique `ration_card_number`
- Citizens are categorized as BPL (Below Poverty Line) or APL (Above Poverty Line)
- `dealer_id` shows which dealer serves this citizen:
  - John (RC2024001) → Ram's shop (dealer_id = 1)

- Priya (RC2024002) → Ram's shop (dealer_id = 1)
- Amit (RC2024003) → Sita's shop (dealer_id = 2)

Step 4: Products Table

The `products` table defines available ration items.

id	product_name	category	unit	price_per_unit	active	description
1	Rice	GRAINS	kg	30.00	true	Premium quality rice
2	Wheat	GRAINS	kg	25.00	true	Whole wheat flour
3	Sugar	GROCERY	kg	40.00	true	White refined sugar
4	Kerosene	FUEL	litre	50.00	true	Cooking fuel

Key Points:

- 4 essential products available
- Each product has a fixed price per unit
- Products categorized for easy management
- All products currently active

Step 5: Initial Inventory Allocation

The `inventory` table tracks stock available with each dealer for each product.

Initial State (After Admin allocates stock):

id	dealer_id	product_id	current_stock	last_updated
1	1	1	100.00	2024-01-05 08:00:00
2	1	2	80.00	2024-01-05 08:00:00
3	2	1	80.00	2024-01-05 08:30:00
4	2	3	50.00	2024-01-05 08:30:00

Interpretation:

- **Ram's shop (dealer_id = 1)** has:
  - 100 kg Rice (product_id = 1)
  - 80 kg Wheat (product_id = 2)
- **Sita's shop (dealer_id = 2)** has:
  - 80 kg Rice (product_id = 1)
  - 50 kg Sugar (product_id = 3)

Key Points:

- Each dealer maintains separate inventory
- Same product can exist in multiple dealers' inventory
- Inventory is tracked by combination of `dealer_id` + `product_id`

Step 6: Distribution Event 1 - John Gets Rice from Ram

Distribution Record:

id	citizen_id	dealer_id	product_id	quantity	total_amount	distribution_date	ration_card_number
1	1	1	1	5.00	150.00	2024-01-10 09:00:00	RC2024001

Business Logic Executed:

```
1. Citizen John (id=1) requests 5 kg Rice from his assigned dealer Ram (id=1)
2. System checks: Does Ram have Rice in stock?
   → Query: SELECT * FROM inventory WHERE dealer_id=1 AND product_id=1
   → Result: current_stock = 100 kg ✓
3. System checks: Is 5 kg available?
   → 100 >= 5 ✓
4. Create distribution record (shown above)
5. Update Ram's inventory: 100 - 5 = 95 kg
6. Both operations succeed atomically (@Transactional)
```

Updated Inventory:

id	dealer_id	product_id	current_stock	last_updated
1	1	1	**95.00**	2024-01-10 09:00:00
2	1	2	80.00	2024-01-05 08:00:00
3	2	1	80.00	2024-01-05 08:30:00
4	2	3	50.00	2024-01-05 08:30:00

Key Observation: Only Ram's Rice stock decreased (100→95). Sita's Rice stock (80 kg) remains unchanged.

Step 7: Distribution Event 2 - Priya Gets Rice from Ram

Distribution Record:

id	citizen_id	dealer_id	product_id	quantity	total_amount	distribution_date	ration_card_number
2	2	1	1	4.00	120.00	2024-01-10 10:30:00	RC2024002

Business Logic Executed:

```
1. Citizen Priya (id=2) requests 4 kg Rice from her assigned dealer Ram (id=1)
2. System checks Ram's Rice stock: 95 kg available ✓
3. System checks: Is 4 kg available? 95 >= 4 ✓
4. Create distribution record
5. Update Ram's inventory: 95 - 4 = 91 kg
```

Updated Inventory:

id	dealer_id	product_id	current_stock	last_updated

id	dealer_id	product_id	current_stock	last_updated
1	1	1	**91.00**	2024-01-10 10:30:00
2	1	2	80.00	2024-01-05 08:00:00
3	2	1	80.00	2024-01-05 08:30:00
4	2	3	50.00	2024-01-05 08:30:00

Key Observation: Ram's Rice further decreased (95→91). Total distributed by Ram: 9 kg (5+4).

Step 8: Distribution Event 3 - Amit Gets Rice from Sita

Distribution Record:

id	citizen_id	dealer_id	product_id	quantity	total_amount	distribution_date	ration_card_number
3	3	2	1	6.00	180.00	2024-01-10 11:00:00	RC2024003

Business Logic Executed:

```
1. Citizen Amit (id=3) requests 6 kg Rice from his assigned dealer Sita (id=2)
2. System checks: Does Sita have Rice in stock?
   → Query: SELECT * FROM inventory WHERE dealer_id=2 AND product_id=1
   → Result: current_stock = 80 kg ✓
3. System checks: Is 6 kg available? 80 >= 6 ✓
4. Create distribution record
5. Update Sita's inventory: 80 - 6 = 74 kg
```

Updated Inventory:

id	dealer_id	product_id	current_stock	last_updated
1	1	1	91.00	2024-01-10 10:30:00
2	1	2	80.00	2024-01-05 08:00:00
3	2	1	**74.00**	2024-01-10 11:00:00
4	2	3	50.00	2024-01-05 08:30:00

Critical Observation:

- Amit is assigned to Sita's shop, so stock deducts from **Sita's inventory only**
- Ram's Rice stock (91 kg) remains unchanged
- This proves dealer-specific inventory isolation

Step 9: Distribution Event 4 - Priya Gets Wheat from Ram

Distribution Record:

id	citizen_id	dealer_id	product_id	quantity	total_amount	distribution_date	ration_card_number

id	citizen_id	dealer_id	product_id	quantity	total_amount	distribution_date	ration_card_number
4	2	1	2	3.00	75.00	2024-01-11 09:00:00	RC2024002

Business Logic Executed:

```
1. Citizen Priya (id=2) requests 3 kg Wheat from her assigned dealer Ram (id=1)
2. System checks: Does Ram have Wheat in stock?
   → Query: SELECT * FROM inventory WHERE dealer_id=1 AND product_id=2
   → Result: current_stock = 80 kg ✓
3. System checks: Is 3 kg available? 80 >= 3 ✓
4. Create distribution record
5. Update Ram's inventory: 80 - 3 = 77 kg
```

Final Inventory State:

id	dealer_id	product_id	current_stock	last_updated
1	1	1	91.00	2024-01-10 10:30:00
2	1	2	**77.00**	2024-01-11 09:00:00
3	2	1	74.00	2024-01-10 11:00:00
4	2	3	50.00	2024-01-05 08:30:00

Complete Data Flow Summary

All Distribution Records (distributions table)

id	citizen_id	dealer_id	product_id	quantity	total_amount	distribution_date	ration_card_number
1	1	1	1	5.00	150.00	2024-01-10 09:00:00	RC2024001
2	2	1	1	4.00	120.00	2024-01-10 10:30:00	RC2024002
3	3	2	1	6.00	180.00	2024-01-10 11:00:00	RC2024003
4	2	1	2	3.00	75.00	2024-01-11 09:00:00	RC2024002

Complete Audit Trail:

- Total distributions: 4
- Total quantity distributed: 18 kg/litre
- Total revenue: ₹525.00
- Distributions by Ram: 3 (John, Priya twice)
- Distributions by Sita: 1 (Amit)

Final Inventory Summary

Dealer	Product	Initial Stock	Distributed	Current Stock
Ram (Zone-A)	Rice	100 kg	9 kg	**91 kg**
Ram (Zone-A)	Wheat	80 kg	3 kg	**77 kg**
Sita (Zone-B)	Rice	80 kg	6 kg	**74 kg**
Sita (Zone-B)	Sugar	50 kg	0 kg	**50 kg**

Key Insights:

- Each dealer maintains completely separate inventory
- Stock deduction happens only from the specific dealer who distributes
- System prevents cross-dealer stock deduction
- Complete traceability: Every transaction recorded

Key Business Logic

1. Dealer-Specific Inventory Tracking

Code Implementation:

```
// In InventoryRepository.java
Optional<Inventory> findByDealerIdAndProductId(Long dealerId, Long productId);

// In DistributionService.java
Inventory inventory = inventoryRepository
    .findByDealerIdAndProductId(request.getDealerId(), request.getProductId())
    .orElseThrow(() -> new ResourceNotFoundException("Inventory not found"));
```

How It Works:

- When citizen requests ration, system looks up inventory using **both** dealer_id and product_id
- This ensures stock is checked and deducted from the correct dealer only
- Example: John (assigned to Ram) requesting Rice → System queries `WHERE dealer_id=1 AND product_id=1`

---

2. Atomic Transactions

Code Implementation:

```
@Transactional
public DistributionResponse distributeRation(DistributionRequest request) {
    // Step 1: Validate citizen, dealer, product
    // Step 2: Check if dealer has sufficient stock
    // Step 3: Create distribution record
    Distribution savedDistribution = distributionRepository.save(distribution);

    // Step 4: Deduct from dealer's inventory
    inventoryService.removeStock(request.getDealerId(),
                                 request.getProductId(),
                                 request.getQuantity());

    // If any step fails, entire transaction rolls back
```

```
    return mapToResponse(savedDistribution);
}
```

Benefits:

- Both distribution record creation AND stock deduction happen together
- If stock deduction fails, distribution record is not created
- Database remains consistent even if errors occur

---

3. Citizen-Dealer Assignment

Code Implementation:

```
// In Citizen.java (model)
@ManyToOne
@JoinColumn(name = "dealer_id")
private Dealer assignedDealer;

// In DistributionService.java
// Validate citizen is assigned to this dealer
if (!citizen.getDealer().getId().equals(request.getDealerId())) {
    throw new InvalidOperationException("Citizen not assigned to this dealer");
}
```

Benefits:

- Each citizen permanently assigned to one dealer
- System validates that citizen can only get rations from their assigned dealer
- Prevents unauthorized distributions

---

4. Stock Availability Check

Code Implementation:

```
// In DistributionService.java
if (inventory.getCurrentStock() < request.getQuantity()) {
    throw new InsufficientStockException(
        "Insufficient stock. Available: " + inventory.getCurrentStock() +
        " " + product.getUnit()
    );
}
```

Benefits:

- Prevents over-distribution
- Provides clear error messages
- Ensures dealers cannot distribute more than they have

API Endpoints Used in This Workflow

User Management

- `POST /api/users/register` - Create users (Admin, Dealer, Citizen)
- `GET /api/users/{id}` - Get user details

Citizen Management

- `POST /api/citizens` - Create citizen profile
- `GET /api/citizens/ration-card/{rationCardNumber}` - Get citizen by ration card

Dealer Management

- `POST /api/dealers` - Create dealer profile
- `GET /api/dealers/{id}` - Get dealer details

Product Management

- `POST /api/products` - Add new products
- `GET /api/products` - List all products

Inventory Management

- `POST /api/inventory/add` - Add stock to dealer's inventory
- `GET /api/inventory/dealer/{dealerId}` - View dealer's current stock

Distribution

- `POST /api/distributions` - Distribute ration to citizen
- `GET /api/distributions/ration-card/{rationCardNumber}` - Get citizen's distribution history
- `GET /api/distributions/dealer/{dealerId}` - Get dealer's distribution history

Admin Reports

- `GET /api/admin/dashboard` - Dashboard statistics
- `GET /api/admin/dealer-report/{dealerId}` - Detailed dealer report
- `GET /api/admin/low-stock-alerts` - Products below threshold

Conclusion

This workflow demonstrates:

1. **Role-Based Architecture**: Admin, Dealer, and Citizen roles with separate responsibilities
2. **Dealer-Specific Inventory**: Each dealer maintains independent stock
3. **Citizen Assignment**: Each citizen linked to specific dealer
4. **Atomic Transactions**: Distribution and stock deduction happen together
5. **Complete Audit Trail**: Every transaction recorded with timestamp
6. **Stock Isolation**: Dealer A's distribution doesn't affect Dealer B's inventory

The system ensures transparency, accountability, and prevents misuse through proper validation and transaction management.

Document Version: 1.0
Last Updated: January 2024
Total Tables: 6 (users, dealer_profiles, citizen_profiles, products, inventory, distributions)

Sample Users: 6 (1 Admin, 2 Dealers, 3 Citizens)
Sample Transactions: 4 distributions tracked