E-Ration PDS System - Complete Testing Guide

Table of Contents

Testing Overview

This guide provides comprehensive testing instructions for the E-Ration PDS system:

- **API Testing**: Testing REST endpoints with sample requests/responses
- **Integration Testing**: Testing component interactions
- **End-to-End Testing**: Complete workflow validation
- All endpoints return JSON with standardized `ApiResponse<T>` format

Prerequisites

1. Development Environment

- Java 17+ installed
- Maven 3.8+ installed
- PostgreSQL (Supabase) configured
- Eureka Server running on port 8761
- Config Server running on port 8888
- Main Application running on port 8081

2. Services Must Be Running

```
# Start Eureka Server (port 8761)
cd eurekaserver
mvn spring-boot:run

# Start Config Server (port 8888)
```

```
cd configserver
mvn spring-boot:run

# Start Main Application (port 8081)
cd mainapp
mvn spring-boot:run

# Start Gateway (port 8080) - API Gateway
cd gateway
mvn spring-boot:run
```

3. Base URL

All API calls use: `http://localhost:8080/api` (via Gateway)

Direct Access: `http://localhost:8081/api` (bypassing Gateway)

API Response Format

All endpoints return responses in this standardized format:

```
{
  "success": true,
  "message": "Operation message",
  "data": { /* response data */ }
}
```

Success Response: `success: true`
Error Response: `success: false`

Authentication APIs

Base Path: `/api/auth`

1. Login

POST `/api/auth/login`

Request Body:

```
{
  "username": "john_citizen",
  "password": "John@123"
}
```

Success Response (Citizen):

```
{
  "success": true,
  "message": "Login successful",
  "data": {
    "user": {
      "id": 2,
      "username": "john_citizen",
      "email": "john@gmail.com",
      "fullName": "John Doe",
      "role": "CITIZEN",
      "phone": "9876543213",
      "active": true,
      "createdAt": "2024-01-15T10:30:00"
    },
    "citizenProfile": {
      "id": 1,
      "userId": 2,
      "rationCardNumber": "RC2024001",
      "address": "23, Palm Street, Mumbai",
      "phoneNumber": "9876543213",
      "familySize": 4,
      "category": "BPL",
      "assignedDealerId": 1,
      "dealerName": "Ram General Store",
      "createdAt": "2024-01-15T10:30:00"
    },
    "dealerProfile": null,
    "message": "Login successful"
  }
}
```


Success Response (Dealer):


```
{
  "success": true,
  "message": "Login successful",
  "data": {
    "user": {
      "id": 3,
      "username": "ram_dealer",
      "email": "ram@dealer.com",
      "fullName": "Ramesh Kumar",
      "role": "DEALER",
      "phone": "9876543211",
      "active": true
    },
    "citizenProfile": null,
    "dealerProfile": {
      "id": 1,
      "userId": 3,
```

```
      "shopName": "Ram General Store",
      "shopLicense": "LIC/2024/001",
      "address": "Shop No. 12, Market Road, Mumbai",
      "phoneNumber": "9876543211",
      "region": "Zone-A",
      "active": true,
      "status": "APPROVED",
      "createdAt": "2024-01-15T09:00:00"
    },
    "message": "Login successful"
  }
}
```

Success Response (Admin):

```
{
  "success": true,
  "message": "Login successful",
  "data": {
    "user": {
      "id": 1,
      "username": "admin",
      "email": "admin@pds.gov.in",
      "fullName": "System Administrator",
      "role": "ADMIN",
      "phone": "9999999999",
      "active": true
    },
    "citizenProfile": null,
    "dealerProfile": null,
    "message": "Login successful"
  }
}
```

Error Response (Invalid Credentials):

```
{
  "success": false,
  "message": "Invalid username or password",
  "data": null
}
```

Error Response (Inactive Account):

```
{
  "success": false,
  "message": "Account is deactivated. Please contact administrator.",
```

```
  "data": null
}
```

User Management APIs

Base Path: `/api/users`

1. Create User

POST `/api/users`

Request Body:

```
{
  "username": "admin123",
  "email": "admin@pds.gov.in",
  "password": "Admin@123",
  "fullName": "System Administrator",
  "role": "ADMIN",
  "phone": "9876543210",
  "aadhaarRef": "123456789012"
}
```

Success Response:

```
{
  "success": true,
  "message": "User created successfully",
  "data": {
    "id": 1,
    "username": "admin123",
    "email": "admin@pds.gov.in",
    "fullName": "System Administrator",
    "role": "ADMIN",
    "phone": "9876543210",
    "aadhaarRef": "123456789012",
    "active": true,
    "createdAt": "2024-01-15T10:30:00"
  }
}
```

2. Get All Users

GET `/api/users`

Success Response:

```
{
  "success": true,
  "message": "Users retrieved successfully",
  "data": [
    {
      "id": 1,
      "username": "admin123",
      "email": "admin@pds.gov.in",
      "fullName": "System Administrator",
      "role": "ADMIN",
      "active": true
    }
  ]
}
```


3. Get User by ID

GET `/api/users/{id}`

Example: `GET /api/users/1`

Success Response:


```
{
  "success": true,
  "message": "User found",
  "data": {
    "id": 1,
    "username": "admin123",
    "email": "admin@pds.gov.in",
    "fullName": "System Administrator",
    "role": "ADMIN",
    "active": true
  }
}
```


4. Update User

PUT `/api/users/{id}`

Request Body:


```
{
  "fullName": "System Admin Updated",
  "phone": "9876543210",
  "email": "admin_new@pds.gov.in"
}
```

Note: Fields are optional on update — only the fields you include will be changed. You do not need to send `password`, `role` or `username` when making a partial update.

Success Response:

```
{
  "success": true,
  "message": "User updated successfully",
  "data": {
    "id": 1,
    "fullName": "System Admin Updated",
    "phone": "9876543210",
    "email": "admin_new@pds.gov.in"
  }
}
```

5. Delete User

DELETE `/api/users/{id}`

Success Response:

```
{
  "success": true,
  "message": "User deleted successfully",
  "data": null
}
```

Citizen Management APIs

Base Path: `/api/citizens`

1. Create Citizen (with User)

POST `/api/citizens`

Request Body:

```
{
  "username": "john_citizen",
  "email": "john@gmail.com",
  "password": "John@123",
  "fullName": "John Doe",
  "phone": "9876543213",
  "rationCardNumber": "RC2024001",
  "address": "23, Palm Street, Mumbai",
  "phoneNumber": "9876543213",
```

```
  "familySize": 4,
  "category": "BPL",
  "assignedDealerId": 1
}
```

Success Response:

```
{
  "success": true,
  "message": "Citizen created successfully",
  "data": {
    "id": 1,
    "rationCardNumber": "RC2024001",
    "address": "23, Palm Street, Mumbai",
    "phoneNumber": "9876543213",
    "familySize": 4,
    "category": "BPL",
    "user": {
      "id": 2,
      "username": "john_citizen",
      "fullName": "John Doe",
      "email": "john@gmail.com"
    },
    "assignedDealer": {
      "id": 1,
      "shopName": "Ram General Store"
    }
  }
}
```

2. Get All Citizens

GET `/api/citizens`

Success Response:

```
{
  "success": true,
  "message": "Citizens retrieved successfully",
  "data": [
    {
      "id": 1,
      "rationCardNumber": "RC2024001",
      "fullName": "John Doe",
      "category": "BPL",
      "familySize": 4
    }
  ]
}
```

3. Get Citizen by ID

GET `/api/citizens/{id}`

4. Update Citizen

PUT `/api/citizens/{id}`

Request Body:

```
{
  "address": "New Address, Mumbai",
  "phoneNumber": "9876543214",
  "familySize": 5,
  "category": "APL"
}
```

5. Delete Citizen

DELETE `/api/citizens/{id}`

Success Response:

```
{
  "success": true,
  "message": "Citizen deleted successfully",
  "data": null
}
```

6. Assign Dealer to Citizen

PATCH `/api/citizens/{citizenId}/assign-dealer/{dealerId}`

Example: `PATCH /api/citizens/1/assign-dealer/1`

Success Response:

```
{
  "success": true,
  "message": "Dealer assigned successfully",
  "data": {
    "id": 1,
    "userId": 2,
    "username": "john_citizen",
    "email": "john@gmail.com",
    "fullName": "John Doe",
    "rationCardNumber": "RC2024001",
```

```
    "address": "23, Palm Street, Mumbai",
    "phoneNumber": "9876543213",
    "familySize": 4,
    "category": "BPL",
    "dealerId": 1,
    "dealerShopName": "Ram General Store",
    "createdAt": "2024-01-15T10:30:00"
  }
}
```

Error Response (Dealer Not Found):

```
{
  "success": false,
  "message": "Dealer not found with id: 999",
  "data": null
}
```

Error Response (Citizen Not Found):

```
{
  "success": false,
  "message": "Citizen not found with id: 999",
  "data": null
}
```

Dealer Management APIs

Base Path: `/api/dealers`

1. Create Dealer (with User)

POST `/api/dealers`

Request Body:

```
{
  "username": "ram_dealer",
  "email": "ram@dealer.com",
  "password": "Ram@123",
  "fullName": "Ramesh Kumar",
  "phone": "9876543211",
  "shopName": "Ram General Store",
  "ownerName": "Ramesh Kumar",
  "shopLicense": "LIC/2024/001",
  "region": "Zone-A",
```

```
  "contactNumber": "9876543211",
  "address": "Shop No. 12, Market Road, Mumbai"
}
```

Success Response:

```
{
  "success": true,
  "message": "Dealer created successfully",
  "data": {
    "id": 1,
    "shopName": "Ram General Store",
    "ownerName": "Ramesh Kumar",
    "shopLicense": "LIC/2024/001",
    "region": "Zone-A",
    "contactNumber": "9876543211",
    "address": "Shop No. 12, Market Road, Mumbai",
    "approved": false,
    "user": {
      "id": 3,
      "username": "ram_dealer",
      "fullName": "Ramesh Kumar",
      "email": "ram@dealer.com"
    }
  }
}
```

2. Get All Dealers

GET `/api/dealers`

3. Get Dealer by ID

GET `/api/dealers/{id}`

4. Update Dealer

PUT `/api/dealers/{id}`

5. Delete Dealer

DELETE `/api/dealers/{id}`

6. Approve Dealer

PUT `/api/dealers/{id}/approve`

Success Response:

```
{
  "success": true,
  "message": "Dealer approved successfully",
  "data": {
    "id": 1,
    "shopName": "Ram General Store",
    "approved": true
  }
}
```

7. Get Dealers by Region

GET `/api/dealers/region/{region}`

Example: `GET /api/dealers/region/Zone-A`

Product Management APIs

Base Path: `/api/products`

1. Create Product

POST `/api/products`

Request Body:

```
{
  "productName": "Rice",
  "category": "GRAINS",
  "unit": "kg",
  "pricePerUnit": 30.0,
  "description": "Premium quality rice"
}
```

Success Response:

```
{
  "success": true,
  "message": "Product created successfully",
  "data": {
    "id": 1,
    "productName": "Rice",
    "category": "GRAINS",
    "unit": "kg",
    "pricePerUnit": 30.0,
    "description": "Premium quality rice"
```

```
  }
}
```

2. Get All Products

GET `/api/products`

Success Response:

```
{
  "success": true,
  "message": "Products retrieved successfully",
  "data": [
    {
      "id": 1,
      "productName": "Rice",
      "category": "GRAINS",
      "unit": "kg",
      "pricePerUnit": 30.0
    }
  ]
}
```

3. Get Product by ID

GET `/api/products/{id}`

4. Update Product

PUT `/api/products/{id}`

Request Body:

```
{
  "productName": "Premium Rice",
  "pricePerUnit": 35.0,
  "description": "Premium basmati rice"
}
```

5. Delete Product

DELETE `/api/products/{id}`

6. Get Products by Category

GET `/api/products/category/{category}`

Example: `GET /api/products/category/GRAINS`

Quota Management APIs

Base Path: `/api/quotas`

1. Set Quota

POST `/api/quotas`

Request Body:

```
{
  "productId": 1,
  "category": "BPL",
  "month": 1,
  "year": 2024,
  "quotaPerCitizen": 25.0,
  "description": "January 2024 quota"
}
```

Success Response:

```
{
  "success": true,
  "message": "Quota set successfully",
  "data": {
    "id": 1,
    "product": {
      "id": 1,
      "productName": "Rice"
    },
    "category": "BPL",
    "quotaAmount": 25.0,
    "validFrom": "2024-01-01",
    "validTo": "2024-12-31"
  }
}
```

2. Get All Quotas

GET `/api/quotas`

3. Get Quota by ID

GET `/api/quotas/{id}`

4. Update Quota

PUT `/api/quotas/{id}`

5. Delete Quota

DELETE `/api/quotas/{id}`

6. Get Quotas by Category

GET `/api/quotas/category/{category}`

Example: `GET /api/quotas/category/BPL`

7. Get Quotas by Product

GET `/api/quotas/product/{productId}`

Inventory Management APIs

Base Path: `/api/inventory`

1. Add Stock to Dealer

POST `/api/inventory/add-stock`

Request Body:

```
{
  "dealerId": 1,
  "productId": 1,
  "quantity": 100.0
}
```

Success Response:

```
{
  "success": true,
  "message": "Stock added successfully",
  "data": {
    "id": 1,
    "dealer": {
      "id": 1,
      "shopName": "Ram General Store"
    },
    "product": {
      "id": 1,
      "productName": "Rice"
    },
    "stockLevel": 100.0,
    "lastUpdated": "2024-01-15T10:30:00"
  }
}
```

2. Deduct Stock from Dealer

POST `/api/inventory/deduct`

Request Body:

```
{
  "dealerId": 1,
  "productId": 1,
  "quantity": 25.0
}
```

Success Response:

```
{
  "success": true,
  "message": "Stock deducted successfully",
  "data": {
    "id": 1,
    "stockLevel": 75.0,
    "lastUpdated": "2024-01-15T11:00:00"
  }
}
```

3. Get Dealer Inventory

GET `/api/inventory/dealer/{dealerId}`

Success Response:

```
{
  "success": true,
  "message": "Inventory retrieved successfully",
  "data": [
    {
      "id": 1,
      "product": {
        "id": 1,
        "productName": "Rice"
      },
      "stockLevel": 75.0,
      "lastUpdated": "2024-01-15T11:00:00"
    }
  ]
}
```

4. Get Product Stock by Dealer

GET `/api/inventory/dealer/{dealerId}/product/{productId}`

Success Response:


```
{
  "success": true,
  "message": "Stock level retrieved",
  "data": 75.0
}
```


5. Get All Inventory

GET `/api/inventory`

6. Get Low Stock Alerts (Stock < threshold)

GET `/api/inventory/low-stock?threshold=20`

Success Response:


```
{
  "success": true,
  "message": "Low stock items retrieved",
  "data": [
    {
      "id": 2,
      "dealer": {
        "shopName": "Ram General Store"
      },
      "product": {
        "productName": "Wheat"
      },
      "stockLevel": 15.0
    }
  ]
}
```

Distribution Management APIs

Base Path: `/api/distributions`

1. Create Distribution (Issue Ration)

POST `/api/distributions`

Request Body:

```
{
  "citizenId": 1,
  "productId": 1,
  "dealerId": 1,
  "quantity": 5.0
}
```

Success Response:

```
{
  "success": true,
  "message": "Distribution created successfully",
  "data": {
    "id": 1,
    "citizen": {
      "id": 1,
      "rationCardNumber": "RC2024001",
      "fullName": "John Doe"
    },
    "product": {
      "id": 1,
      "productName": "Rice"
    },
    "dealer": {
      "id": 1,
      "shopName": "Ram General Store"
    },
    "quantity": 5.0,
    "distributionDate": "2024-01-15T10:30:00",
    "status": "COMPLETED"
  }
}
```

2. Get All Distributions

GET `/api/distributions`

3. Get Distribution by ID

GET `/api/distributions/{id}`

4. Get Distributions by Citizen

GET `/api/distributions/citizen/{citizenId}`

Success Response:

```
{
  "success": true,
  "message": "Distributions retrieved successfully",
  "data": [
    {
      "id": 1,
      "product": {
        "productName": "Rice"
      },
      "quantity": 5.0,
      "distributionDate": "2024-01-15T10:30:00",
      "status": "COMPLETED"
    }
  ]
}
```


5. Get Distributions by Dealer

GET `/api/distributions/dealer/{dealerId}`

6. Get Distributions by Date Range

GET `/api/distributions/date-range?startDate=2024-01-01&endDate=2024-01-31`

Success Response:


```
{
  "success": true,
  "message": "Distributions in date range retrieved",
  "data": [
    {
      "id": 1,
      "citizen": {
        "rationCardNumber": "RC2024001"
      },
      "product": {
        "productName": "Rice"
      },
      "quantity": 5.0,
      "distributionDate": "2024-01-15"
    }
  ]
}
```

Admin/Reports APIs

Base Path: `/api/admin`

1. Get Distribution Summary

GET `/api/admin/reports/summary`

Success Response:


```
{
  "success": true,
  "message": "Distribution summary retrieved",
  "data": {
    "totalDistributions": 150,
    "totalQuantityDistributed": 3750.0,
    "totalCitizensServed": 50,
    "totalDealersActive": 10,
    "periodStart": "2024-01-01",
    "periodEnd": "2024-01-31"
  }
}
```


2. Get Distribution by Product Report

GET `/api/admin/reports/by-product`

Success Response:


```
{
  "success": true,
  "message": "Product-wise distribution report",
  "data": [
    {
      "productId": 1,
      "productName": "Rice",
      "totalQuantity": 2000.0,
      "distributionCount": 80
    },
    {
      "productId": 2,
      "productName": "Wheat",
      "totalQuantity": 1750.0,
      "distributionCount": 70
    }
  ]
}
```


3. Get Distribution by Dealer Report

GET `/api/admin/reports/by-dealer`

Success Response:

```
{
  "success": true,
  "message": "Dealer-wise distribution report",
  "data": [
    {
      "dealerId": 1,
      "dealerName": "Ramesh Kumar",
      "shopName": "Ram General Store",
      "totalDistributions": 45,
      "totalQuantityDistributed": 1125.0
    }
  ]
}
```


4. Get Distribution by Category Report

GET `/api/admin/reports/by-category`

Success Response:


```
{
  "success": true,
  "message": "Category-wise distribution report",
  "data": [
    {
      "category": "BPL",
      "citizenCount": 30,
      "totalQuantity": 2250.0,
      "distributionCount": 90
    },
    {
      "category": "APL",
      "citizenCount": 20,
      "totalQuantity": 1500.0,
      "distributionCount": 60
    }
  ]
}
```


5. Get Top Dealers by Distribution

GET `/api/admin/reports/top-dealers?limit=5`

Success Response:


```
{
  "success": true,
  "message": "Top dealers retrieved",
```

```
  "data": [
    {
      "dealerId": 1,
      "dealerName": "Ramesh Kumar",
      "distributionCount": 45,
      "rank": 1
    }
  ]
}
```

6. Get System Statistics

GET `/api/admin/stats`

Success Response:

```
{
  "success": true,
  "message": "System statistics retrieved",
  "data": {
    "totalUsers": 125,
    "totalCitizens": 100,
    "totalDealers": 10,
    "totalProducts": 8,
    "totalDistributions": 450,
    "activeQuotas": 16,
    "approvedDealers": 8,
    "pendingDealers": 2
  }
}
```

Complete Test Scenarios

End-to-End Scenario: Complete Distribution Flow

Prerequisites

1. Eureka Server running on port 8761
2. Config Server running on port 8888
3. Gateway running on port 8080
4. Main Application running on port 8081
5. Database connected (Supabase PostgreSQL)

Step 1: Setup Base Data

```
# Create Admin User
curl -X POST http://localhost:8080/api/users \
```

```
  -H "Content-Type: application/json" \
  -d '{
    "username": "admin",
    "email": "admin@pds.gov.in",
    "password": "Admin@123",
    "fullName": "System Admin",
    "role": "ADMIN",
    "phone": "9999999999"
  }'

# Test Admin Login
curl -X POST http://localhost:8080/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "username": "admin",
    "password": "Admin@123"
  }'
```

Step 2: Create Products

```
# Create Rice
curl -X POST http://localhost:8080/api/products \
  -H "Content-Type: application/json" \
  -d '{
    "productName": "Rice",
    "category": "GRAINS",
    "unit": "kg",
    "pricePerUnit": 30.0,
    "description": "Premium quality rice"
  }'

# Create Wheat
curl -X POST http://localhost:8080/api/products \
  -H "Content-Type: application/json" \
  -d '{
    "productName": "Wheat",
    "category": "GRAINS",
    "unit": "kg",
    "pricePerUnit": 25.0,
    "description": "Whole wheat"
  }'
```

Step 3: Set Quotas

```
# BPL Quota for Rice
curl -X POST http://localhost:8080/api/quotas \
  -H "Content-Type: application/json" \
  -d '{
```

```
    "productId": 1,
    "category": "BPL",
    "quotaAmount": 25.0,
    "validFrom": "2024-01-01",
    "validTo": "2024-12-31"
  }'

# APL Quota for Rice
curl -X POST http://localhost:8080/api/quotas \
  -H "Content-Type: application/json" \
  -d '{
    "productId": 1,
    "category": "APL",
    "quotaAmount": 15.0,
    "validFrom": "2024-01-01",
    "validTo": "2024-12-31"
  }'
```

Step 4: Create Dealer

```
curl -X POST http://localhost:8080/api/dealers \
  -H "Content-Type: application/json" \
  -d '{
    "username": "dealer1",
    "email": "dealer1@pds.com",
    "password": "Dealer@123",
    "fullName": "Ramesh Kumar",
    "phone": "9876543211",
    "shopName": "Ram General Store",
    "ownerName": "Ramesh Kumar",
    "shopLicense": "LIC/2024/001",
    "region": "Zone-A",
    "contactNumber": "9876543211",
    "address": "Market Road, Mumbai"
  }'
```

Step 5: Approve Dealer (Admin Action)

```
curl -X PUT http://localhost:8080/api/dealers/1/approve
```

Test Dealer Login after Approval:

```
curl -X POST http://localhost:8080/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "username": "dealer1",
```

```
    "password": "Dealer@123"
  }'
```

Step 6: Add Stock to Dealer

```
# Add Rice Stock
curl -X POST http://localhost:8080/api/inventory/add-stock \
  -H "Content-Type: application/json" \
  -d '{
    "dealerId": 1,
    "productId": 1,
    "quantity": 500.0
  }'

# Add Wheat Stock
curl -X POST http://localhost:8080/api/inventory/add-stock \
  -H "Content-Type: application/json" \
  -d '{
    "dealerId": 1,
    "productId": 2,
    "quantity": 400.0
  }'
```

Step 7: Create Citizens

```
# BPL Citizen
curl -X POST http://localhost:8080/api/citizens \
  -H "Content-Type: application/json" \
  -d '{
    "username": "citizen1",
    "email": "citizen1@gmail.com",
    "password": "Citizen@123",
    "fullName": "John Doe",
    "phone": "9876543213",
    "rationCardNumber": "RC2024001",
    "address": "Palm Street, Mumbai",
    "phoneNumber": "9876543213",
    "familySize": 4,
    "category": "BPL",
    "assignedDealerId": 1
  }'

# Test Citizen Login
curl -X POST http://localhost:8080/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "username": "citizen1",
    "password": "Citizen@123"
```

```
  }'

# APL Citizen
curl -X POST http://localhost:8080/api/citizens \
  -H "Content-Type: application/json" \
  -d '{
    "username": "citizen2",
    "email": "citizen2@gmail.com",
    "password": "Citizen@123",
    "fullName": "Jane Smith",
    "phone": "9876543214",
    "rationCardNumber": "RC2024002",
    "address": "Oak Street, Mumbai",
    "phoneNumber": "9876543214",
    "familySize": 3,
    "category": "APL",
    "assignedDealerId": 1
  }'

# Test APL Citizen Login
curl -X POST http://localhost:8080/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "username": "citizen2",
    "password": "Citizen@123"
  }'
```

Step 8: Issue Rations (Distribution)

```
# Distribute Rice to BPL Citizen
curl -X POST http://localhost:8080/api/distributions \
  -H "Content-Type: application/json" \
  -d '{
    "citizenId": 1,
    "productId": 1,
    "dealerId": 1,
    "quantity": 25.0
  }'

# Distribute Rice to APL Citizen
curl -X POST http://localhost:8080/api/distributions \
  -H "Content-Type: application/json" \
  -d '{
    "citizenId": 2,
    "productId": 1,
    "dealerId": 1,
    "quantity": 15.0
  }'
```

Step 9: Verify Inventory Deduction

```
# Check Dealer Inventory
curl -X GET http://localhost:8080/api/inventory/dealer/1

# Expected: Rice stock should be 460 (500 - 25 - 15)
```

Step 10: Generate Reports

```
# Distribution Summary
curl -X GET http://localhost:8080/api/admin/reports/summary

# By Product Report
curl -X GET http://localhost:8080/api/admin/reports/by-product

# By Dealer Report
curl -X GET http://localhost:8080/api/admin/reports/by-dealer

# By Category Report
curl -X GET http://localhost:8080/api/admin/reports/by-category
```

Error Responses

Common Error Formats

Resource Not Found (404)

```
{
  "success": false,
  "message": "Citizen not found with id: 999",
  "data": null
}
```

Resource Already Exists (409)

```
{
  "success": false,
  "message": "User already exists with username: john_citizen",
  "data": null
}
```

Validation Error (400)

```
{
  "success": false,
  "message": "Validation failed",
  "data": {
    "username": "Username is required",
    "email": "Invalid email format"
  }
}
```

Insufficient Stock (400)

```
{
  "success": false,
  "message": "Insufficient stock. Available: 10.0, Requested: 25.0",
  "data": null
}
```

Testing Tools

Using Postman

1. **Import Collection**: Create collections for each API group
2. **Environment Variables**:
   - `base_url` = `http://localhost:8080/api` (via Gateway)
   - `direct_url` = `http://localhost:8081/api` (direct to mainapp)
   - `admin_token` (if authentication is added later)
3. **Test Scripts** (Postman Tests tab):

```
// Verify success response
pm.test("Status is 200", function() {
    pm.response.to.have.status(200);
});

pm.test("Response has success flag", function() {
    var jsonData = pm.response.json();
    pm.expect(jsonData.success).to.eql(true);
});
```

Using cURL (Command Line)

```
# Get all users (via Gateway)
curl -X GET http://localhost:8080/api/users

# Or directly to mainapp
curl -X GET http://localhost:8081/api/users

# Create user (formatted for readability)
curl -X POST http://localhost:8080/api/users \
  -H "Content-Type: application/json" \
  -d @user_data.json

# With inline JSON
curl -X POST http://localhost:8080/api/users \
  -H "Content-Type: application/json" \
  -d
'{"username":"test","email":"test@test.com","password":"Test@123","fullName":"Test
User","role":"CITIZEN"}'
```

Using PowerShell (Windows)

```
# Get all users (via Gateway)
Invoke-RestMethod -Uri "http://localhost:8080/api/users" -Method Get

# Create user
$body = @{
    username = "test"
    email = "test@test.com"
    password = "Test@123"
    fullName = "Test User"
    role = "CITIZEN"
} | ConvertTo-Json

Invoke-RestMethod -Uri "http://localhost:8080/api/users" -Method Post -Body $body
-ContentType "application/json"
```

Database Verification

Check Tables Created

```
-- Connect to Supabase PostgreSQL
-- Host: aws-1-ap-southeast-1.pooler.supabase.com
-- Port: 5432
-- Database: postgres
-- User: postgres.donziacyqhankvqsclex

-- Verify tables
SELECT table_name FROM information_schema.tables
```

```
WHERE table_schema = 'public';

-- Expected tables:
-- users, citizen_profiles, dealer_profiles, products, quotas,
-- inventory, distributions, stock_predictions
```

Verify Data

```
-- Check users
SELECT id, username, full_name, role FROM users;

-- Check citizens with dealers
SELECT c.ration_card_number, u.full_name, c.category, d.shop_name
FROM citizen_profiles c
JOIN users u ON c.user_id = u.id
LEFT JOIN dealer_profiles d ON c.assigned_dealer_id = d.id;

-- Check inventory levels
SELECT d.shop_name, p.product_name, i.stock_level
FROM inventory i
JOIN dealer_profiles d ON i.dealer_id = d.id
JOIN products p ON i.product_id = p.id;

-- Check distributions
SELECT u.full_name, p.product_name, dist.quantity, dist.distribution_date
FROM distributions dist
JOIN citizen_profiles c ON dist.citizen_id = c.id
JOIN users u ON c.user_id = u.id
JOIN products p ON dist.product_id = p.id
ORDER BY dist.distribution_date DESC;
```

Troubleshooting

Application Not Starting

1. Check if Eureka Server is running: `http://localhost:8761`
2. Check if Config Server is running: `http://localhost:8888`
3. Verify database connection in logs
4. Check port 8081 is not in use

Database Connection Issues

1. Verify Supabase credentials in `configserver/src/main/resources/config/mainapp.properties`
2. Check SSL mode is set to `require`
3. Test connection using psql or pgAdmin

API Returning 404

1. Verify application started successfully
2. Check base URL: `http://localhost:8081/api`
3. Verify endpoint path is correct

Inventory Not Deducting

1. Verify inventory exists for dealer + product
2. Check stock level is sufficient
3. Review application logs for errors

Next Steps

1. ☑ All APIs documented with request/response examples
2. ☑ End-to-end test scenario provided
3. ☑ Error handling documented
4. ☑ Database verification queries provided

Recommended Testing Order

1. Start with User Management APIs
2. Create Products and set Quotas
3. Create and approve Dealers
4. Add Inventory to Dealers
5. Create Citizens
6. Perform Distributions
7. Generate Reports

Last Updated: January 2024
Application Version: Spring Boot 3.5.5
Database: PostgreSQL (Supabase)
Ports: Eureka (8761), Config (8888), Main App (8081) }'

```
**Expected**: Inventory created/updated

#### Step 2: Check Dealer Inventory
```bash
curl -X GET http://localhost:8080/api/inventory/dealer/1
```

Expected: List of all products in dealer's inventory

Step 3: Check Low Stock

```
curl -X GET http://localhost:8080/api/inventory/low-stock?threshold=50
```
```

Expected: Products below threshold

Scenario 4: Quota Management Flow

Step 1: Set Monthly Quotas

```
# BPL Rice Quota
curl -X POST http://localhost:8080/api/quotas \
  -H "Content-Type: application/json" \
  -d '{
    "productId": 1,
    "category": "BPL",
    "month": 11,
    "year": 2024,
    "quotaPerCitizen": 10.0,
    "description": "November 2024 Rice quota for BPL"
  }'

# APL Rice Quota
curl -X POST http://localhost:8080/api/quotas \
  -H "Content-Type: application/json" \
  -d '{
    "productId": 1,
    "category": "APL",
    "month": 11,
    "year": 2024,
    "quotaPerCitizen": 5.0,
    "description": "November 2024 Rice quota for APL"
  }'
```

Step 2: Check Citizen Quota Balance

```
curl -X GET http://localhost:8080/api/quotas/citizen/RC2024001/current
```

Expected: List of products with quota status

```
[
  {
    "productName": "Rice",
    "totalQuota": 10.0,
    "redeemedQuantity": 0.0,
    "remainingQuota": 10.0,
    "percentageUsed": 0.0,
    "status": "AVAILABLE"
  }
]
```

Scenario 5: Distribution Flow (Complete Workflow)

Step 1: Check Prerequisites

```
# Verify citizen exists
curl -X GET http://localhost:8080/api/citizens/ration-card/RC2024001

# Verify dealer has stock
curl -X GET http://localhost:8080/api/inventory/dealer/1

# Check citizen's quota
curl -X GET http://localhost:8080/api/quotas/citizen/RC2024001/current
```

Step 2: Distribute Ration

```
curl -X POST http://localhost:8080/api/distributions \
  -H "Content-Type: application/json" \
  -d '{
    "rationCardNumber": "RC2024001",
    "dealerId": 1,
    "productId": 1,
    "quantity": 5.0,
    "remarks": "Regular monthly distribution"
  }'
```

Expected Response: 201 Created

```
{
  "id": 1,
  "citizenName": "John Doe",
  "rationCardNumber": "RC2024001",
  "dealerName": "Ram General Store",
  "productName": "Rice",
  "quantity": 5.0,
  "totalAmount": 150.0,
  "distributionDate": "2024-11-09T10:30:00",
  "status": "COMPLETED"
}
```

Step 3: Verify Post-Distribution State

Check Updated Inventory:

```
curl -X GET http://localhost:8080/api/inventory/dealer/1
```

Expected: Rice stock reduced by 5 kg

Check Updated Quota:

```
curl -X GET http://localhost:8080/api/quotas/citizen/RC2024001/current
```

Expected:

```
{
  "productName": "Rice",
  "totalQuota": 10.0,
  "redeemedQuantity": 5.0,
  "remainingQuota": 5.0,
  "percentageUsed": 50.0,
  "status": "AVAILABLE"
}
```

Step 4: View Distribution History

```
curl -X GET http://localhost:8080/api/distributions/ration-card/RC2024001
```

Scenario 6: Admin Reports and Analytics

Dashboard Statistics

```
curl -X GET http://localhost:8080/api/admin/dashboard
```

Expected:

```
{
  "totalCitizens": 3,
  "totalDealers": 2,
  "totalProducts": 4,
  "totalDistributions": 10,
  "activeDealers": 2,
  "activeProducts": 4
}
```

Dealer Performance Report

```
curl -X GET http://localhost:8080/api/admin/dealer-report/1
```

Expected: Detailed dealer statistics

Low Stock Alerts

```
curl -X GET http://localhost:8080/api/admin/low-stock-alerts
```

Date Range Distributions

```
curl -X GET "http://localhost:8080/api/admin/distributions/date-range?
startDate=2024-11-01T00:00:00&endDate=2024-11-30T23:59:59"
```

Test Data Setup

Complete Test Data Script

```
#!/bin/bash
BASE_URL="http://localhost:8080"

# 1. Create Admin
curl -X POST $BASE_URL/api/users \
  -H "Content-Type: application/json" \
  -d
'{"username":"admin","email":"admin@pds.gov.in","password":"Admin@123","fullName":
"Admin","role":"ADMIN"}'

# 2. Create Dealers
curl -X POST $BASE_URL/api/dealers \
  -H "Content-Type: application/json" \
  -d
'{"username":"ram_dealer","email":"ram@dealer.com","password":"Ram@123","fullName"
:"Ramesh Kumar","shopName":"Ram Store","ownerName":"Ramesh
Kumar","shopLicense":"LIC001","region":"Zone-
A","contactNumber":"9876543211","address":"Mumbai"}'

curl -X POST $BASE_URL/api/dealers \
  -H "Content-Type: application/json" \
  -d
'{"username":"sita_dealer","email":"sita@dealer.com","password":"Sita@123","fullNa
me":"Sita Devi","shopName":"Sita Provisions","ownerName":"Sita
Devi","shopLicense":"LIC002","region":"Zone-
```

```
B","contactNumber":"9876543212","address":"Mumbai"}'

# 3. Create Citizens
curl -X POST $BASE_URL/api/citizens \
  -H "Content-Type: application/json" \
  -d
'{"username":"john","email":"john@gmail.com","password":"John@123","fullName":"Joh
n
Doe","rationCardNumber":"RC2024001","address":"Mumbai","phoneNumber":"9876543213",
"familySize":4,"category":"BPL","dealerId":1}'

curl -X POST $BASE_URL/api/citizens \
  -H "Content-Type: application/json" \
  -d
'{"username":"priya","email":"priya@gmail.com","password":"Priya@123","fullName":"
Priya
Sharma","rationCardNumber":"RC2024002","address":"Mumbai","phoneNumber":"987654321
4","familySize":3,"category":"APL","dealerId":1}'

# 4. Create Products
curl -X POST $BASE_URL/api/products \
  -H "Content-Type: application/json" \
  -d
'{"productName":"Rice","category":"GRAINS","unit":"kg","pricePerUnit":30.0,"descri
ption":"Premium rice"}'

curl -X POST $BASE_URL/api/products \
  -H "Content-Type: application/json" \
  -d
'{"productName":"Wheat","category":"GRAINS","unit":"kg","pricePerUnit":25.0,"descr
iption":"Wheat flour"}'

# 5. Add Inventory
curl -X POST $BASE_URL/api/inventory/add \
  -H "Content-Type: application/json" \
  -d '{"dealerId":1,"productId":1,"quantity":100.0}'

curl -X POST $BASE_URL/api/inventory/add \
  -H "Content-Type: application/json" \
  -d '{"dealerId":1,"productId":2,"quantity":80.0}'

# 6. Set Quotas
curl -X POST $BASE_URL/api/quotas \
  -H "Content-Type: application/json" \
  -d
'{"productId":1,"category":"BPL","month":11,"year":2024,"quotaPerCitizen":10.0}'

curl -X POST $BASE_URL/api/quotas \
  -H "Content-Type: application/json" \
  -d
'{"productId":1,"category":"APL","month":11,"year":2024,"quotaPerCitizen":5.0}'

echo "Test data setup complete!"
```

Unit Testing

Example Unit Tests

UserService Test

```
@SpringBootTest
public class UserServiceTest {

    @Autowired
    private UserService userService;

    @Test
    public void testCreateUser_Success() {
        UserRequest request = UserRequest.builder()
            .username("testuser")
            .email("test@example.com")
            .password("Test@123")
            .fullName("Test User")
            .role("CITIZEN")
            .build();

        UserResponse response = userService.createUser(request);

        assertNotNull(response);
        assertEquals("testuser", response.getUsername());
        assertEquals("CITIZEN", response.getRole());
    }

    @Test
    public void testCreateUser_DuplicateUsername_ThrowsException() {
        // First user
        UserRequest request1 = UserRequest.builder()
            .username("duplicate")
            .email("user1@example.com")
            .password("Test@123")
            .fullName("User One")
            .role("CITIZEN")
            .build();
        userService.createUser(request1);

        // Duplicate username
        UserRequest request2 = UserRequest.builder()
            .username("duplicate")
            .email("user2@example.com")
            .password("Test@123")
            .fullName("User Two")
            .role("CITIZEN")
            .build();
```

```
        assertThrows(ResourceAlreadyExistsException.class, () -> {
            userService.createUser(request2);
        });
    }
}
```

DistributionService Test

```
@SpringBootTest
@Transactional
public class DistributionServiceTest {

    @Autowired
    private DistributionService distributionService;

    @Autowired
    private InventoryService inventoryService;

    @Test
    public void testDistributeRation_Success() {
        // Setup: Create citizen, dealer, product, inventory

        DistributionRequest request = DistributionRequest.builder()
            .rationCardNumber("RC2024001")
            .dealerId(1L)
            .productId(1L)
            .quantity(5.0)
            .remarks("Test distribution")
            .build();

        DistributionResponse response =
distributionService.distributeRation(request);

        assertNotNull(response);
        assertEquals(5.0, response.getQuantity());
        assertEquals("COMPLETED", response.getStatus());
    }

    @Test
    public void testDistributeRation_InsufficientStock_ThrowsException() {
        DistributionRequest request = DistributionRequest.builder()
            .rationCardNumber("RC2024001")
            .dealerId(1L)
            .productId(1L)
            .quantity(1000.0) // More than available
            .build();

        assertThrows(RuntimeException.class, () -> {
            distributionService.distributeRation(request);
        });
    }
```

```
    @Test
    public void testDistributeRation_QuotaExceeded_ThrowsException() {
        // Setup: Quota = 10kg, Already redeemed = 9kg

        DistributionRequest request = DistributionRequest.builder()
            .rationCardNumber("RC2024001")
            .dealerId(1L)
            .productId(1L)
            .quantity(5.0) // Would exceed quota
            .build();

        assertThrows(RuntimeException.class, () -> {
            distributionService.distributeRation(request);
        });
    }
}
```

QuotaService Test

```
@SpringBootTest
public class QuotaServiceTest {

    @Autowired
    private QuotaService quotaService;

    @Test
    public void testGetCitizenQuotaStatus_Success() {
        // Setup: Create citizen, quota, some distributions

        List<CitizenQuotaStatus> status = quotaService
            .getCitizenCurrentMonthQuota("RC2024001");

        assertNotNull(status);
        assertFalse(status.isEmpty());

        CitizenQuotaStatus riceStatus = status.get(0);
        assertEquals(10.0, riceStatus.getTotalQuota());
        assertTrue(riceStatus.getRemainingQuota() > 0);
    }

    @Test
    public void testIsQuotaAvailable_ReturnsTrue_WhenSufficientQuota() {
        boolean available = quotaService.isQuotaAvailable(
            1L,      // citizenId
            1L,      // productId (Rice)
            5.0,     // requestedQuantity
            11,      // month
            2024     // year
        );
```

```
        assertTrue(available);
    }

    @Test
    public void testIsQuotaAvailable_ReturnsFalse_WhenQuotaExceeded() {
        // Setup: Citizen already redeemed 9kg out of 10kg quota

        boolean available = quotaService.isQuotaAvailable(
            1L,      // citizenId
            1L,      // productId
            5.0,     // requestedQuantity (would exceed)
            11,      // month
            2024     // year
        );

        assertFalse(available);
    }
}
```

Integration Testing

End-to-End Distribution Test

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
public class DistributionIntegrationTest {

    @Autowired
    private TestRestTemplate restTemplate;

    private static Long dealerId;
    private static Long citizenId;
    private static Long productId;

    @Test
    @Order(1)
    public void setupTestData() {
        // Create dealer
        DealerRequest dealerRequest = // ... create request
        ResponseEntity<DealerResponse> dealerResponse =
            restTemplate.postForEntity("/api/dealers", dealerRequest,
DealerResponse.class);
        dealerId = dealerResponse.getBody().getId();

        // Create citizen
        // Create product
        // Add inventory
        // Set quota
    }
```

```
    @Test
    @Order(2)
    public void testCompleteDistributionWorkflow() {
        // 1. Check initial quota
        String quotaUrl = "/api/quotas/citizen/RC2024001/current";
        ResponseEntity<List> initialQuota = restTemplate.getForEntity(quotaUrl,
List.class);
        assertEquals(HttpStatus.OK, initialQuota.getStatusCode());

        // 2. Perform distribution
        DistributionRequest request = DistributionRequest.builder()
            .rationCardNumber("RC2024001")
            .dealerId(dealerId)
            .productId(productId)
            .quantity(5.0)
            .build();

        ResponseEntity<DistributionResponse> distResponse =
            restTemplate.postForEntity("/api/distributions", request,
DistributionResponse.class);
        assertEquals(HttpStatus.CREATED, distResponse.getStatusCode());

        // 3. Verify quota updated
        ResponseEntity<List> updatedQuota = restTemplate.getForEntity(quotaUrl,
List.class);
        // Assert quota reduced

        // 4. Verify inventory updated
        String inventoryUrl = "/api/inventory/dealer/" + dealerId;
        ResponseEntity<List> inventory = restTemplate.getForEntity(inventoryUrl,
List.class);
        // Assert stock reduced
    }
}
```

Common Test Cases

Test Case Matrix

Test Case	API Endpoint	Method	Expected Result
Create user with valid data	/api/users	POST	201 Created
Create user with duplicate username	/api/users	POST	409 Conflict
Create user with invalid email	/api/users	POST	400 Bad Request
Get user by valid ID	/api/users/{id}	GET	200 OK
Get user by invalid ID	/api/users/{id}	GET	404 Not Found

Test Case	API Endpoint	Method	Expected Result
Distribute with sufficient stock	/api/distributions	POST	201 Created
Distribute with insufficient stock	/api/distributions	POST	400 Bad Request
Distribute exceeding quota	/api/distributions	POST	400 Bad Request
Check quota before distribution	/api/quotas/citizen/{ration}/current	GET	200 OK with balance
Add inventory to dealer	/api/inventory/add	POST	201 Created
Get low stock alerts	/api/inventory/low-stock	GET	200 OK with list

Error Testing

Test Invalid Scenarios

1. Distribution with Non-existent Citizen

```
curl -X POST http://localhost:8080/api/distributions \
  -H "Content-Type: application/json" \
  -d '{
    "rationCardNumber": "INVALID999",
    "dealerId": 1,
    "productId": 1,
    "quantity": 5.0
  }'
```

Expected: 404 Not Found - "Citizen not found"

2. Distribution with Inactive Dealer

```
# First deactivate dealer
curl -X PATCH http://localhost:8080/api/dealers/1/toggle-status

# Then try distribution
curl -X POST http://localhost:8080/api/distributions \
  -H "Content-Type: application/json" \
  -d '{
    "rationCardNumber": "RC2024001",
    "dealerId": 1,
    "productId": 1,
    "quantity": 5.0
  }'
```

Expected: 400 Bad Request - "Dealer is not active"

3. Create Quota with Invalid Month

```
curl -X POST http://localhost:8080/api/quotas \
  -H "Content-Type: application/json" \
  -d '{
    "productId": 1,
    "category": "BPL",
    "month": 13,
    "year": 2024,
    "quotaPerCitizen": 10.0
  }'
```

Expected: 400 Bad Request - Validation error

Performance Testing

Load Testing with Apache JMeter

Test Plan 1: Distribution API

- **Threads**: 100 concurrent users
- **Ramp-up**: 10 seconds
- **Duration**: 60 seconds
- **Expected**: < 500ms response time, 0% error rate

Test Plan 2: Quota Check API

- **Threads**: 500 concurrent users
- **Ramp-up**: 20 seconds
- **Duration**: 120 seconds
- **Expected**: < 200ms response time, 0% error rate

Test Coverage Goals

- **Unit Test Coverage**: > 80%
- **Integration Test Coverage**: > 70%
- **API Test Coverage**: 100% of endpoints
- **Critical Path Coverage**: 100%

Continuous Integration Testing

GitHub Actions Workflow

```
name: PDS CI/CD
```

```yaml
on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2

      - name: Set up JDK 17
        uses: actions/setup-java@v2
        with:
          java-version: '17'

      - name: Run tests
        run: mvn clean test

      - name: Generate coverage report
        run: mvn jacoco:report
```

Troubleshooting

Application Not Starting

1. Check if Eureka Server is running: `http://localhost:8761`
2. Check if Config Server is running: `http://localhost:8888`
3. Check if Gateway is running: `http://localhost:8080`
4. Verify database connection in logs
5. Check ports 8080 (Gateway) and 8081 (MainApp) are not in use

Database Connection Issues

1. Verify Supabase credentials in `configserver/src/main/resources/config/mainapp.properties`
2. Check SSL mode is set to `require`
3. Test connection using psql or pgAdmin

API Returning 404

1. Verify all services started successfully
2. Check base URL: `http://localhost:8080/api` (Gateway) or `http://localhost:8081/api` (Direct)
3. Verify endpoint path is correct
4. Check Gateway routing configuration in Eureka: `http://localhost:8761`

Inventory Not Deducting

1. Verify inventory exists for dealer + product
2. Check stock level is sufficient
3. Review application logs for errors

Next Steps

1. ☑ All APIs documented with request/response examples
2. ☑ End-to-end test scenario provided
3. ☑ Error handling documented
4. ☑ Database verification queries provided

Recommended Testing Order

1. Start with User Management APIs
2. Create Products and set Quotas
3. Create and approve Dealers
4. Add Inventory to Dealers
5. Create Citizens
6. Perform Distributions
7. Generate Reports

Access Points

- **Gateway (Recommended)**: `http://localhost:8080/api/*` - Load balanced via Spring Cloud Gateway
- **Direct Access**: `http://localhost:8081/api/*` - Bypass gateway for debugging
- **Service Discovery**: `http://localhost:8761` - Eureka dashboard
- **Config Server**: `http://localhost:8888` - Centralized configuration

Last Updated: November 2024
Application Version: Spring Boot 3.5.5
Database: PostgreSQL (Supabase)
Architecture: Microservices with Spring Cloud
Ports:

- Eureka Server: 8761
- Config Server: 8888
- Gateway: 8080 ★ (Main Entry Point)
- Main Application: 8081