1. Introduction

The Food Store Inventory Management System is a robust Java application designed to streamline inventory operations for small to medium-sized grocery businesses. This documentation provides a complete technical overview of the system's architecture, functionality, and implementation details across six comprehensive sections.

2. System Overview

2.1 Purpose

This system replaces manual inventory tracking with automated processes to:

- Minimize human errors in stock recording
- Optimize product ordering cycles
- Reduce financial losses from expired goods
- Provide real-time business insights

2.2 Technical Specifications

- Development Language: Java 11
- Architecture: Layered (Presentation-Business-Data)
- Storage: In-memory data structures
- Interface: Console-based with menu navigation
- Data Format: JSON-compatible structure

2.3 Functional Scope

The system handles:

✓ Product lifecycle management
✓ Perishable goods tracking
✓ Automated stock alerts
✓ Inventory valuation
✓ Categorical analysis

3. Detailed System Architecture

3.1 Component Diagram

The system comprises three primary layers:

1. Presentation Layer
    - Main.java: Application entry point
    - GestorInventario.java: User interface controller
2. Business Logic Layer
    - ProductoDAO.java: Operation contracts
    - ProductoDAOImpl.java: Business rules implementation
3. Data Layer
    - Producto.java: Data model definition

3.2 Class Responsibilities

- Producto: Models product attributes including perishability status
- ProductoDAO: Defines CRUD operations interface
- ProductoDAOImpl: Implements inventory algorithms
- GestorInventario: Manages user workflows

3.3 Data Flow

1. User interacts with console menu
2. GestorInventario validates input
3. ProductoDAOImpl processes requests
4. Results return through presentation layer

4. Core Functionality

4.1 Inventory Management

- Product Registration: Captures 6 data fields per item
- Stock Updates: Real-time quantity modifications
- Product Removal: Complete inventory deletion

4.2 Monitoring Features

- Low-Stock Detection: <10 units threshold

- Expiration Tracking: Date-based alerts
- Category Watch: Department-level monitoring

4.3 Reporting Capabilities

- Inventory Valuation: Current stock worth
- Category Breakdown: Product distribution
- Alert Summaries: Consolidated warnings

5. Implementation Details

5.1 Data Modeling

The Producto class contains:

- Primitive fields (ID, name, quantity)
- Financial data (price, inventory value)
- Temporal data (expiration date)
- Boolean flags (perishable status)

5.2 Business Logic

Key algorithms include:

- Stock level evaluation
- Expiration date parsing
- Inventory summation
- Categorical aggregation

5.3 User Interface

The console interface features:

- Numbered menu system
- Input validation
- Tabular data display
- Contextual help prompts

6. System Limitations & Future Enhancements

6.1 Current Constraints

- Volatile storage (no database persistence)

- Basic date handling
- Single-user operation

6.2 Proposed Improvements

1. Database Integration
    - MySQL for persistent storage
    - Transaction history logging
2. Advanced Features
    - Barcode scanning support
    - Multi-store inventory sync
    - Supplier management module
3. Enhanced UI
    - Graphical interface
    - Touchscreen compatibility
    - Mobile access
4. Analytics Expansion
    - Sales trend forecasting
    - Waste reduction analysis
    - Profit margin calculations

Conclusion

This inventory management system provides small grocery businesses with essential digital tools to replace error-prone manual processes. While currently implementing core inventory functionality, the modular design permits straightforward expansion into a comprehensive retail management solution. Future development will focus on adding persistence, advanced analytics, and multi-platform accessibility to transform this into an enterprise-grade system.

By implementing DAOs, we have a simple program that won't cause us any problems in the future if we want to change or tweak any part of the project without any problems, allowing it

to be modified in the future depending on the company's changing interests. Therefore, this way, we only change certain parts of the project without altering it or accidentally breaking the code.