FAZAIA BILQUIS COLLEGE NUR KHAN BASE RAWALPINDI

Course: Data Structures

Submitted To:

Ma'am Sana Maqsood

Submitted By:

Rubab Fatima

BSCS-13-F24-28

Aneesa Sultan

BSCS-13-F24-20

Maria Atta

BSCS-13-F24-01

Table of Content:

Abstract

ExpirySync is a Java-based educational project developed to demonstrate practical **Data Structures and Algorithms (DSA)** in a real-world inspired inventory management scenario. The system tracks perishable products, identifies expired and urgent items using **queues, stacks, and priority queues**, and provides an interactive web interface. The project successfully implements core DSA concepts with efficient operations and serves as a learning tool for visualizing theoretical concepts in practical applications.

1. Introduction

Inventory management is essential for businesses dealing with **perishable products**. Manual tracking systems are slow, error-prone, and inefficient for handling large datasets. ExpirySync addresses these challenges by providing an **automated, DSA-driven solution** that monitors expiry dates, prioritizes urgent items, and demonstrates how fundamental data structures can solve real-world problems effectively through an interactive web application.

2. Problem Statement

Manual inventory systems often fail to:

- Track expiry dates efficiently across large product databases
- Highlight urgent items requiring immediate attention
- Handle inventory operations with optimal time complexity
- Provide real-time sorting and searching capabilities

Inputs: Product details (ID, name, quantity, expiry date)

Outputs: Processed inventory with urgent/expired alerts, sorted views, search results

Constraints: Local storage only, no database persistence, single-user system

3. Objectives

- Implement core *DSA concepts (Queue, Stack, Priority Queue*) in a practical application
- Manage inventory with expiry dates using efficient data structures
- Automatically identify and prioritize urgent and expired products
- Provide an interactive web interface for user operations
- Demonstrate sorting and searching algorithms in inventory management
- Serve as an educational tool for Data Structures students

4. Scope and Limitations

Scope:

- Web-based inventory management interface
- Real-time expiry tracking and alerting
- Sorting by multiple criteria (name, date, quantity)
- Product search functionality
- Operation history logging
- Priority-based urgent item handling

Limitations:

- No database integration (in-memory storage only)

- No user authentication or authorization system
- No HTTPS security implementation
- Designed for local/demonstration use only
- Limited scalability for large datasets
- Basic user interface design

5. Tools / Technologies Used

COMPONENT	TECHNOLOGY
PROGRAMMING LANGUAGE	Java 11+
BUILD TOOL	Maven
WEB SERVER	Java HttpServer
FRONTEND	HTML5, CSS3, JavaScript
IDE	Visual Studio Code
OPERATING SYSTEM	Windows
VERSION CONTROL	Git

6. Data Structures Used

6.1 Queue (FIFO)

- **Purpose:** Process products based on expiry order (first-expiring-first-out)
- **Operations:** enqueue(), dequeue(), peek(), isEmpty()
- **Complexity:** Insert O(1), Delete O(1), Search O(n)

6.2 Stack (LIFO)

- **Purpose:** Maintain operation history for undo/redo functionality
- **Operations:** push(), pop(), peek(), isEmpty()
- **Complexity:** All operations O(1)

6.3 Priority Queue

- **Purpose:** Handle urgent products based on expiry proximity
- **Operations:** insert(), extractMin(), peek()
- **Complexity:** Insert O(log n), Extract O(log n) (using binary heap)

6.4 Array/ArrayList

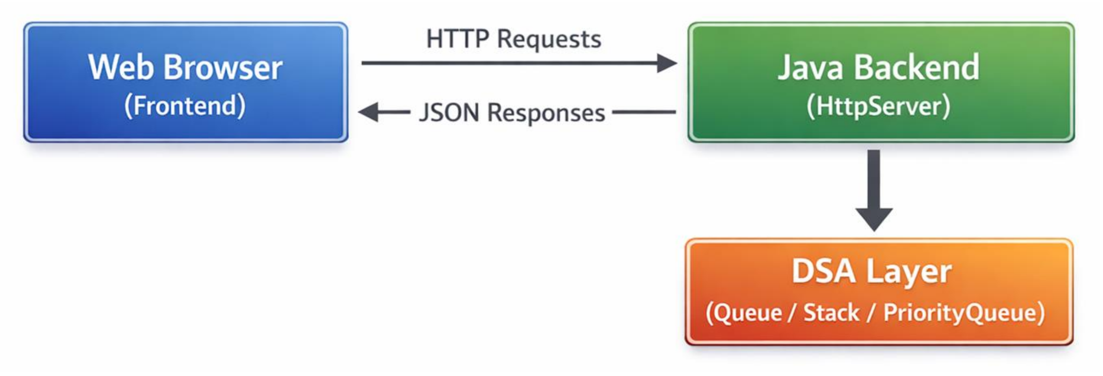
- **Purpose:** Store and manage product collections
- **Operations:** add(), remove(), get(), set()
- **Complexity:** Random access O(1), Insert/Delete O(n)

7. Algorithms Used

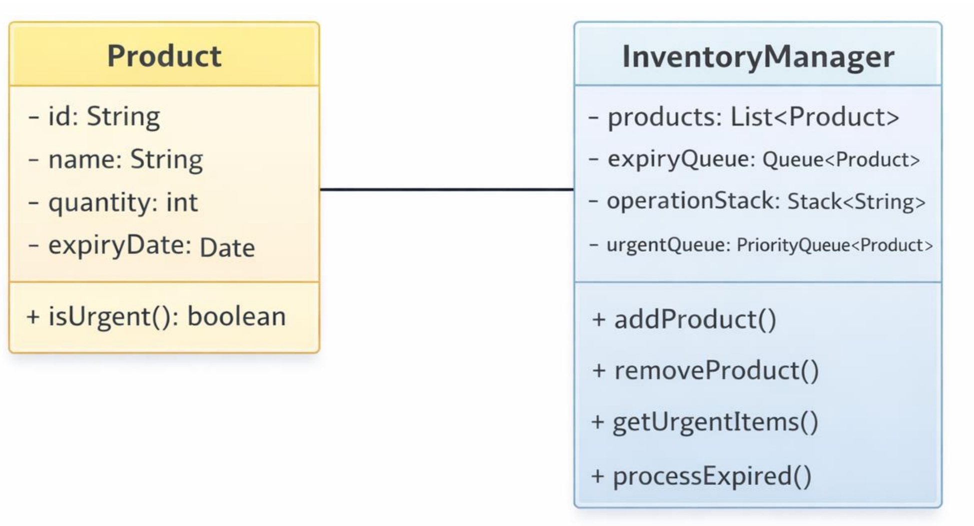
ALGORITHM	PURPOSE	TIME COMPLEXITY
QUICK SORT	Sort products by name/date/quantity	O(n log n) average
MERGE SORT	Alternative sorting for stability	O(n log n)
LINEAR SEARCH	Search products by ID or name	O(n)
BINARY SEARCH	Search in sorted collections	O(log n)
BFS TRAVERSAL	Process queue elements	O(V + E)

8. System Design

Architecture Diagram:



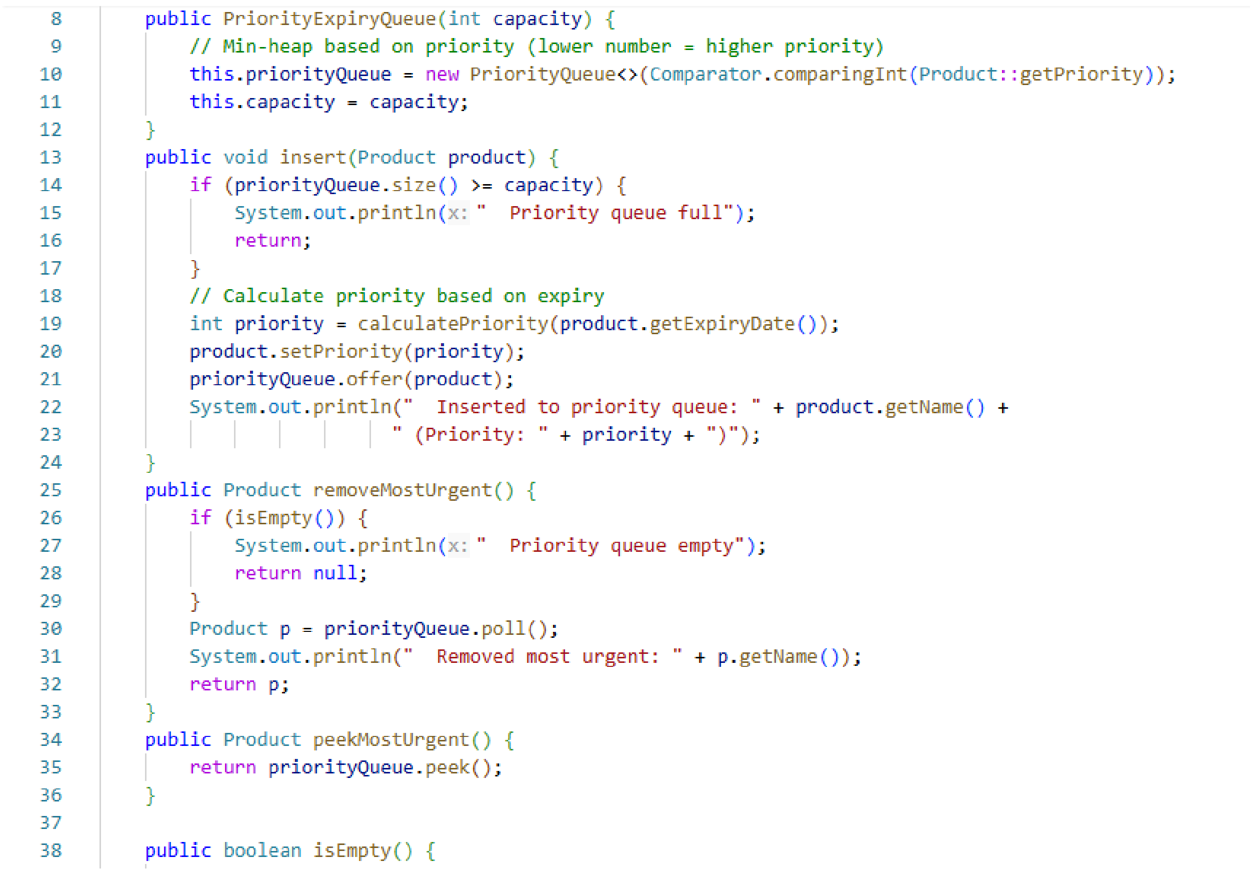
Class Diagram (Simplified):



9. Implementation Details

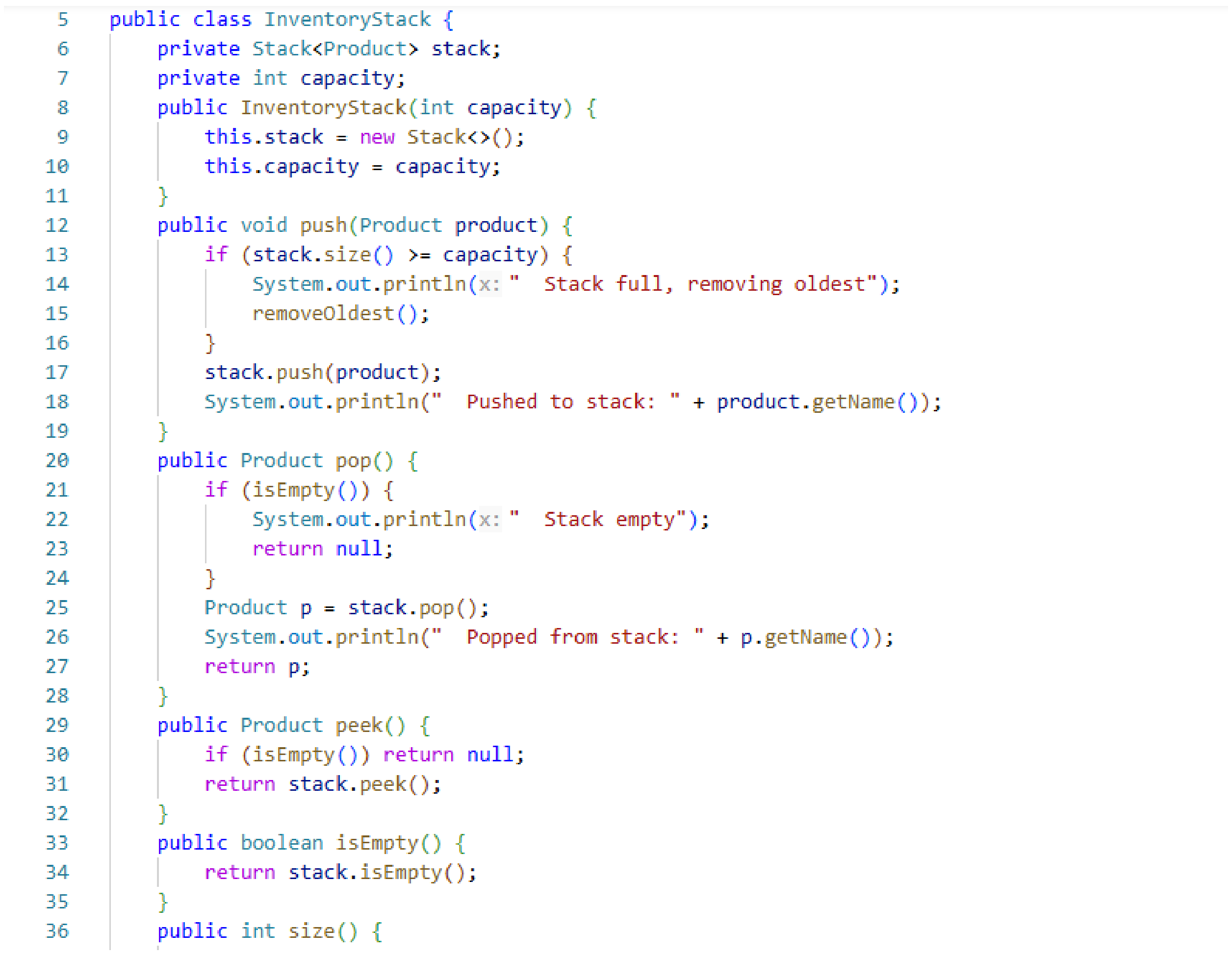
Key Classes and Methods:

- **PriorityQueue Class:**



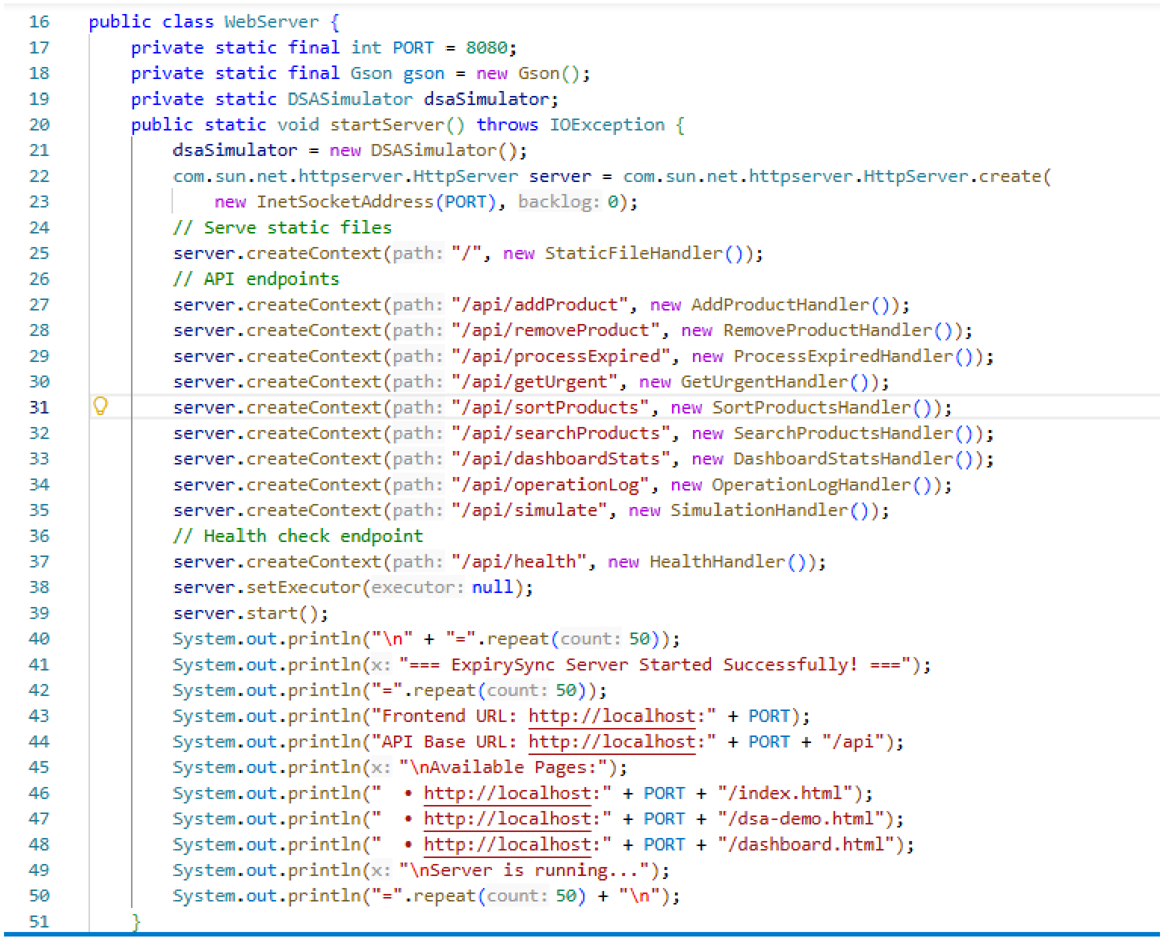
```java
    public PriorityExpiryQueue(int capacity) {
        // Min-heap based on priority (lower number = higher priority)
        this.priorityQueue = new PriorityQueue<>(Comparator.comparingInt(Product::getPriority));
        this.capacity = capacity;
    }
    public void insert(Product product) {
        if (priorityQueue.size() >= capacity) {
            System.out.println(x: "  Priority queue full");
            return;
        }
        // Calculate priority based on expiry
        int priority = calculatePriority(product.getExpiryDate());
        product.setPriority(priority);
        priorityQueue.offer(product);
        System.out.println("  Inserted to priority queue: " + product.getName() +
                          " (Priority: " + priority + ")");
    }
    public Product removeMostUrgent() {
        if (isEmpty()) {
            System.out.println(x: "  Priority queue empty");
            return null;
        }
        Product p = priorityQueue.poll();
        System.out.println("  Removed most urgent: " + p.getName());
        return p;
    }
    public Product peekMostUrgent() {
        return priorityQueue.peek();
    }

    public boolean isEmpty() {
```

- **InventoryStack Class (Core Logic):**



```java
public class InventoryStack {
    private Stack<Product> stack;
    private int capacity;
    public InventoryStack(int capacity) {
        this.stack = new Stack<>();
        this.capacity = capacity;
    }
    public void push(Product product) {
        if (stack.size() >= capacity) {
            System.out.println(x: "  Stack full, removing oldest");
            removeOldest();
        }
        stack.push(product);
        System.out.println("  Pushed to stack: " + product.getName());
    }
    public Product pop() {
        if (isEmpty()) {
            System.out.println(x: "  Stack empty");
            return null;
        }
        Product p = stack.pop();
        System.out.println("  Popped from stack: " + p.getName());
        return p;
    }
    public Product peek() {
        if (isEmpty()) return null;
        return stack.peek();
    }
    public boolean isEmpty() {
        return stack.isEmpty();
    }
    public int size() {
```

- **WebServer Implementation:**

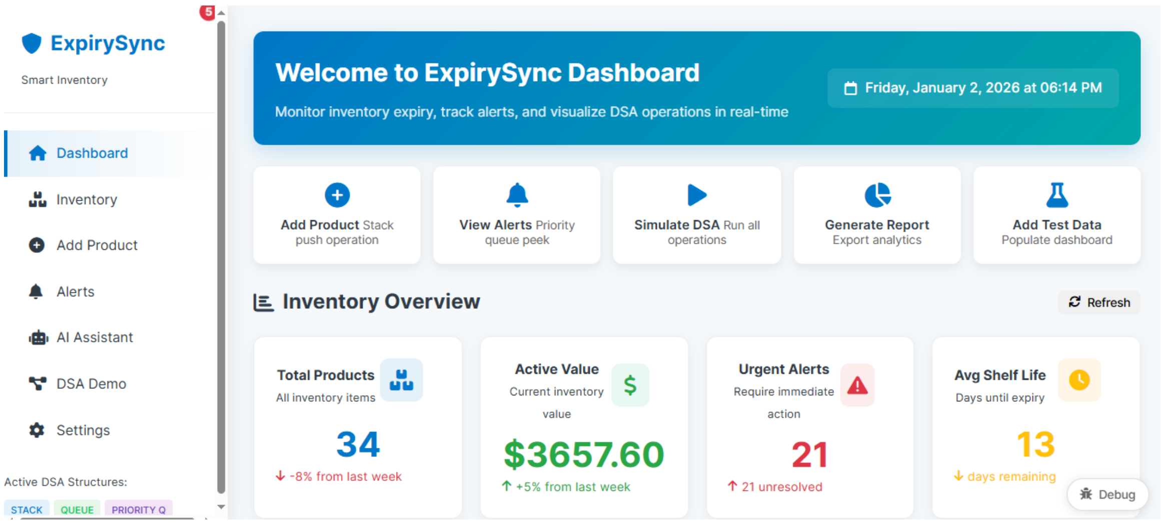


```java
public class WebServer {
    private static final int PORT = 8080;
    private static final Gson gson = new Gson();
    private static DSASimulator dsaSimulator;
    public static void startServer() throws IOException {
        dsaSimulator = new DSASimulator();
        com.sun.net.httpserver.HttpServer server = com.sun.net.httpserver.HttpServer.create(
            new InetSocketAddress(PORT), backlog: 0);
        // Serve static files
        server.createContext(path: "/", new StaticFileHandler());
        // API endpoints
        server.createContext(path: "/api/addProduct", new AddProductHandler());
        server.createContext(path: "/api/removeProduct", new RemoveProductHandler());
        server.createContext(path: "/api/processExpired", new ProcessExpiredHandler());
        server.createContext(path: "/api/getUrgent", new GetUrgentHandler());
        server.createContext(path: "/api/sortProducts", new SortProductsHandler());
        server.createContext(path: "/api/searchProducts", new SearchProductsHandler());
        server.createContext(path: "/api/dashboardStats", new DashboardStatsHandler());
        server.createContext(path: "/api/operationLog", new OperationLogHandler());
        server.createContext(path: "/api/simulate", new SimulationHandler());
        // Health check endpoint
        server.createContext(path: "/api/health", new HealthHandler());
        server.setExecutor(executor: null);
        server.start();
        System.out.println("\n" + "=".repeat(count: 50));
        System.out.println(x: "=== ExpirySync Server Started Successfully! ===");
        System.out.println("=".repeat(count: 50));
        System.out.println("Frontend URL: http://localhost:" + PORT);
        System.out.println("API Base URL: http://localhost:" + PORT + "/api");
        System.out.println(x: "\nAvailable Pages:");
        System.out.println("  • http://localhost:" + PORT + "/index.html");
        System.out.println("  • http://localhost:" + PORT + "/dsa-demo.html");
        System.out.println("  • http://localhost:" + PORT + "/dashboard.html");
        System.out.println(x: "\nServer is running...");
        System.out.println("=".repeat(count: 50) + "\n");
    }
```

10. Results / Output

Sample Output Screenshots:

- **Main Dashboard:**



ExpirySync
Smart Inventory

Dashboard
Inventory
Add Product
Alerts
AI Assistant
DSA Demo
Settings

Active DSA Structures:
STACK QUEUE PRIORITY Q

Welcome to ExpirySync Dashboard
Monitor inventory expiry, track alerts, and visualize DSA operations in real-time
Friday, January 2, 2026 at 06:14 PM

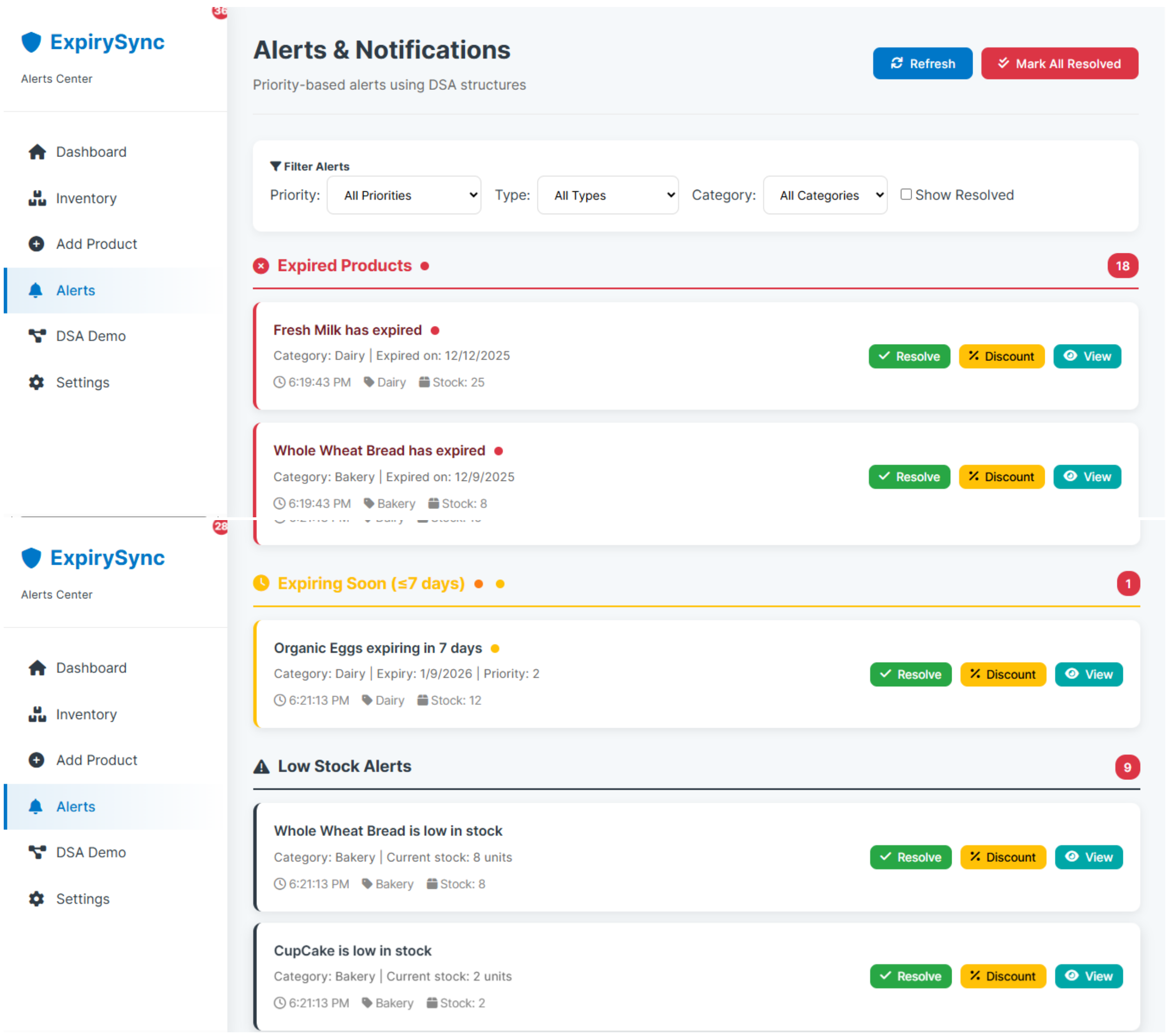
Add Product Stack push operation
View Alerts Priority queue peek
Simulate DSA Run all operations
Generate Report Export analytics
Add Test Data Populate dashboard

Inventory Overview Refresh

Total Products — All inventory items — 34 — -8% from last week
Active Value — Current inventory value — $3657.60 — +5% from last week
Urgent Alerts — Require immediate action — 21 — 21 unresolved
Avg Shelf Life — Days until expiry — 13 — days remaining

Debug

- **Urgent Items List:**



ExpirySync
Alerts Center

Dashboard
Inventory
Add Product
Alerts
DSA Demo
Settings

Alerts & Notifications

Priority-based alerts using DSA structures

Refresh | Mark All Resolved

Filter Alerts
Priority: All Priorities | Type: All Types | Category: All Categories | Show Resolved

Expired Products 18

Fresh Milk has expired
Category: Dairy | Expired on: 12/12/2025
6:19:43 PM Dairy Stock: 25
Resolve | Discount | View

Whole Wheat Bread has expired
Category: Bakery | Expired on: 12/9/2025
6:19:43 PM Bakery Stock: 8
Resolve | Discount | View

Expiring Soon (≤7 days) 1

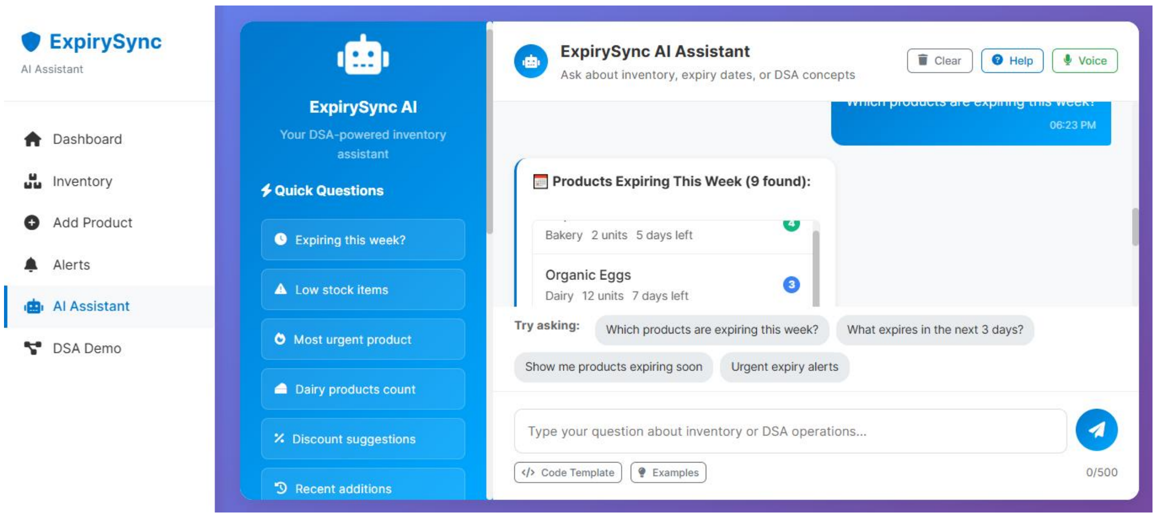
Organic Eggs expiring in 7 days
Category: Dairy | Expiry: 1/9/2026 | Priority: 2
6:21:13 PM Dairy Stock: 12
Resolve | Discount | View

Low Stock Alerts 9

Whole Wheat Bread is low in stock
Category: Bakery | Current stock: 8 units
6:21:13 PM Bakery Stock: 8
Resolve | Discount | View

CupCake is low in stock
Category: Bakery | Current stock: 2 units
6:21:13 PM Bakery Stock: 2
Resolve | Discount | View

- **AI ChatBot:**



ExpirySync
AI Assistant

Dashboard
Inventory
Add Product
Alerts
AI Assistant
DSA Demo

ExpirySync AI
Your DSA-powered inventory assistant

Quick Questions
Expiring this week?
Low stock items
Most urgent product
Dairy products count
Discount suggestions
Recent additions

ExpirySync AI Assistant
Ask about inventory, expiry dates, or DSA concepts
Clear | Help | Voice

Which products are expiring this week?
06:23 PM

Products Expiring This Week (9 found):
Bakery 2 units 5 days left
Organic Eggs
Dairy 12 units 7 days left

Try asking: Which products are expiring this week? | What expires in the next 3 days? | Show me products expiring soon | Urgent expiry alerts

Type your question about inventory or DSA operations...
Code Template | Examples
0/500

11. Testing

Test Case Table:

TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
ADD PRODUCT	Valid product JSON	Success message	Success message	✅ PASS
ADD PRODUCT	Invalid date format	Error message	Error message	✅ PASS
GET URGENT	No urgent items	Empty list	Empty list	✅ PASS
SEARCH PRODUCT	Existing ID	Product details	Product details	✅ PASS
SEARCH PRODUCT	Non-existent ID	"Not found"	"Not found"	✅ PASS
SORT PRODUCTS	By name	Sorted list	Sorted list	✅ PASS
PROCESS EXPIRED	2 expired items	Removed 2 items	Removed 2 items	✅ PASS

Edge Cases Tested:

- Empty inventory operations
- Duplicate product IDs
- Invalid expiry dates (past dates)
- Large quantity values
- Special characters in product names
- Concurrent operations simulation

12. Conclusion

ExpirySync successfully demonstrates the practical application of Data Structures and Algorithms in solving real-world inventory management problems. The project effectively implements **Queue** for FIFO expiry processing, **Stack** for operation history, and **Priority Queue** for urgent item handling. Through this implementation, we achieved:

- Efficient management of perishable products
- Automatic identification of urgent/expired items
- Practical demonstration of DSA concepts beyond theoretical study
- Interactive web interface for user operations
- Foundation for further enhancements and scalability

The project serves as an excellent educational tool, bridging the gap between theoretical data structures and their practical applications in software development.

13. Future Enhancements

1. **Database Integration:** Implement MySQL/PostgreSQL for persistent storage
2. **User Authentication:** Add login system with role-based access control
3. **Enhanced Security:** Implement HTTPS and input validation
4. **Advanced UI:** React/Angular-based modern frontend
5. **Reporting:** Generate PDF/Excel reports of inventory status
6. **Mobile Application:** Android/iOS companion app
7. **Machine Learning:** Predictive analytics for stock optimization
8. **Cloud Deployment:** Docker containerization and cloud hosting
9. **Automated Alerts:** Email/SMS notifications for urgent items
10. **Barcode Integration:** Scanner support for product management

14. References

1. Java Documentation - Oracle
2. Maven Official Documentation
3. "Data Structures and Algorithms in Java" - Robert Lafore

4. "Introduction to Algorithms" - Cormen, Leiserson, Rivest, Stein
5. MDN Web Docs - HTML, CSS, JavaScript
6. GeeksforGeeks Data Structures Tutorials
7. W3Schools Java and Web Technologies Guides

15. Appendix

A. How to Run the Project

Prerequisites:

- Java JDK 11 or higher
- Apache Maven 3.6+
- Web browser (Chrome/Firefox)

Steps:

1. Extract project files to a directory
2. Open command prompt/terminal in project root
3. Run: mvn clean compile
4. Run: mvn exec:java -Dexec.mainClass="com.expirysync.Main"
5. Open browser and navigate to: http://localhost:8080
6. Use the web interface to manage inventory

Alternative (Without Maven):

1. Compile: javac -d bin src/*.java
2. Run: java -cp bin com.expirysync.Main