UCS503 - SOFTWAREENGINEERING LAB

INVENTORY MANAGEMENT SYSTEM

UCS 503 Software Engineering Project Report

Mid-Semester Evaluation

Submitted by:

(102303874)Diya Shah

(102303875)Ananya Sharma

(102303854)Harkirat Singh

Submitted to:

Dr. Raghav B.Venkatarmayier

Computer Science andEngineering Department

Thapar Institute of Engineeringand Technology (TIET), Patiala

TABLE OF CONTENTS

1.PROJECT SELECTION PHASE

1.1Software Bid

Project Name: Inventory Management System for small businesses.

Problem Statement:

In Indian retail environments, whether small corner shops, supermarkets, or departmental stores, manual inventory management remains a significant challenge. Shopkeepers face critical issues:

- **ManualStockTracking:** Error-prone, time-consuming inventory record-keeping leads to stock discrepancies.
- **Real-Time Updates:** Lack of instant visibility into current stock levels causes overstocking or stockouts.
- **Billing Inefficiency:** Manual entry during checkout increases transaction time and customer wait times.
- **Low Stock Alerts:** Without automated notification systems, essential items go out of stock unexpectedly.
- **No Sales Analytics:** Difficulty in tracking which products are fast-moving vs. slow-moving.

Proposed Solution:

An **Inventory Management System with Barcode Scanning**leveraging modern web technologies to enable:

- Real-time barcode scanning at billing counters for instant inventory updates.
- Automated low-stock threshold alerts to prevent stockouts.
- Dynamic order creation with live barcode integration.
- Dashboard analytics for stock status and sales trends.
- Multi-user access with role-based control (Admin, Shopkeeper, Staff).

Justification:

- **Cost Efficiency:** Eliminates manual data entry errors and reduces labor time.
- **Operational Speed:** Barcode scanning reduces transaction time by ~70%.
- **Real-Time Inventory:** Shopkeepers always have current stock visibility.
- **Scalability:** System grows with the business—from single shop to chain stores.
- **Technology Fit:** MERN stack ensures rapid development, cost-effectiveness, and wide deployment compatibility.

Project Scope:

- User authentication (Login/Signup with role-based access)
- Product catalog management with barcode association
- Real-time barcode scanning using device camera (HTML5-QRCode library)
- Inventory tracking with dynamic threshold settings
- Order creation with automatic stock deduction
- Low-stock notification system (Email/In-app alerts)
- Analytics dashboard showing key metrics
- Transaction logging and audit trails

Time & Resource Estimate:

- **Timeline:** 4weeks(mid-semester to end-semester)
- **Team:** 3 developers
- **Technologies:** HTML,CSS, React (Frontend), Node.js/Express (Backend), MongoDB (Database)
- **Hosting:** Local/Cloud deployment ready

1.2 Project Overview

Title: Inventory Management System

Objective:

To design and develop a **web-based, real-time inventory management system** that:

1. Enables shopkeepers to scan product barcodes for instant inventory updates.
2. Maintains dynamic stock levels with automatic deduction upon order creation.
3. Provides automated low-stock alerts based on user-defined thresholds.
4. Offers analytics dashboards for business decision-making.
5. Supports multi-user access with role-based permissions.

Problem Statement:

Small-to-medium retail businesses in India lack affordable, easy-to-use inventory solutions. Current manual processes lead to:

- Inaccurate stock records
- Missed sales opportunities due to stockout
- Lack of real visibility

This system addresses these gaps through automation and real-time barcode integration.

Key Features:

1. **User Authentication:** Secure login with JWT tokens; role-based access (Admin, Shopkeeper, Staff)
2. **Product Management:** Add, edit, delete products with barcode association
3. **Live Barcode Scanning:** Real-time scanning via device camera; auto-detection of products
4. **Dynamic Inventory:** Stock levels updated automatically upon order creation
5. **Threshold Management:** Per-product low-stock thresholds with automated alerts
6. **Order Creation:** Shopkeeper scans products, system builds order and updates inventory
7. **Notifications:** Email/In-app alerts when stock falls below threshold
8. **Dashboard Analytics:** Real-time stats (Total Products, Total Stock, Low Stock Items)
9. **Activity Logging:** Track all inventory transactions with timestamps and user attribution

System Architecture:

The system follows a **three-tier MERN architecture:**

- **Frontend (React):** Interactive, responsive UI for all user roles. Components: Dashboard, Products, Inventory, Barcode Scanner, Order Creation.
- **Backend(Node.js/Express):** RESTful API handling business logic, database operations, and notifications.
- **Database(MongoDB):** NoSQL database storing users, products, barcodes, inventory, orders, and transactions.

Workflow Overview:

1. **Admin:** Manages users, sets system-wide thresholds, reviews analytics.
2. **Shopkeeper/Staff:** Scans barcodes during billing, creates orders, updates inventory.
3. **System:** Auto-deducts stock, checks thresholds, triggers notifications, logs transactions.

Technologies Used:

Layer	Technology	Purpose
Frontend	HTML, CSS, React	User Interface, State Management
Backend	Node.js, Express	API Server, Business Logic
Database	MongoDB	Data Persistence (Users, Products, Inventory, Orders)
Libraries	html5-qrcode, Axios, Nodemailer	Barcode Scanning, HTTP Client, Email Notifications
Authentication	JWT (JSON Web Tokens)	Secure user sessions
Deployment	Local/Docker/Cloud	Development & Production Environments

Expected Outcomes:

1. **Functional System:** Afullyoperational web application deployable on local or cloud servers.
2. **90% Faster Billing:** Barcodescanning reduces transaction time significantly.
3. **100% Real-Time Inventory:** Live stock updates with no delays.
4. **Zero Stockouts:** Automatedalerts prevent unexpected out-of-stock situations.
5. **Reusable Code:** Well-documented, modular codebase for future enhancements.

Future Enhancements:

1. **Mobile App:** Nativemobile application for iOS/Android.
2. **Multi-Location Support:** Manage inventory across multiple store branches.
3. **Advanced Analytics:** Predictive analytics for stock forecasting using ML.
4. **Supplier Integration:** Auto-generate purchase orders when stock is low.
5. **Barcode Generation:** In-system barcode label printing.
6. **SMS Notifications:** WhatsApp/SMS alerts for critical stock levels.
7. **Offline Mode:** Workwithout internet; sync when reconnected.
8. **POS Integration:** FullPoint-of-Sale system with payment gateway integration.

2. ANALYSIS PHASE

2.1 Use Cases

Actors in the System:

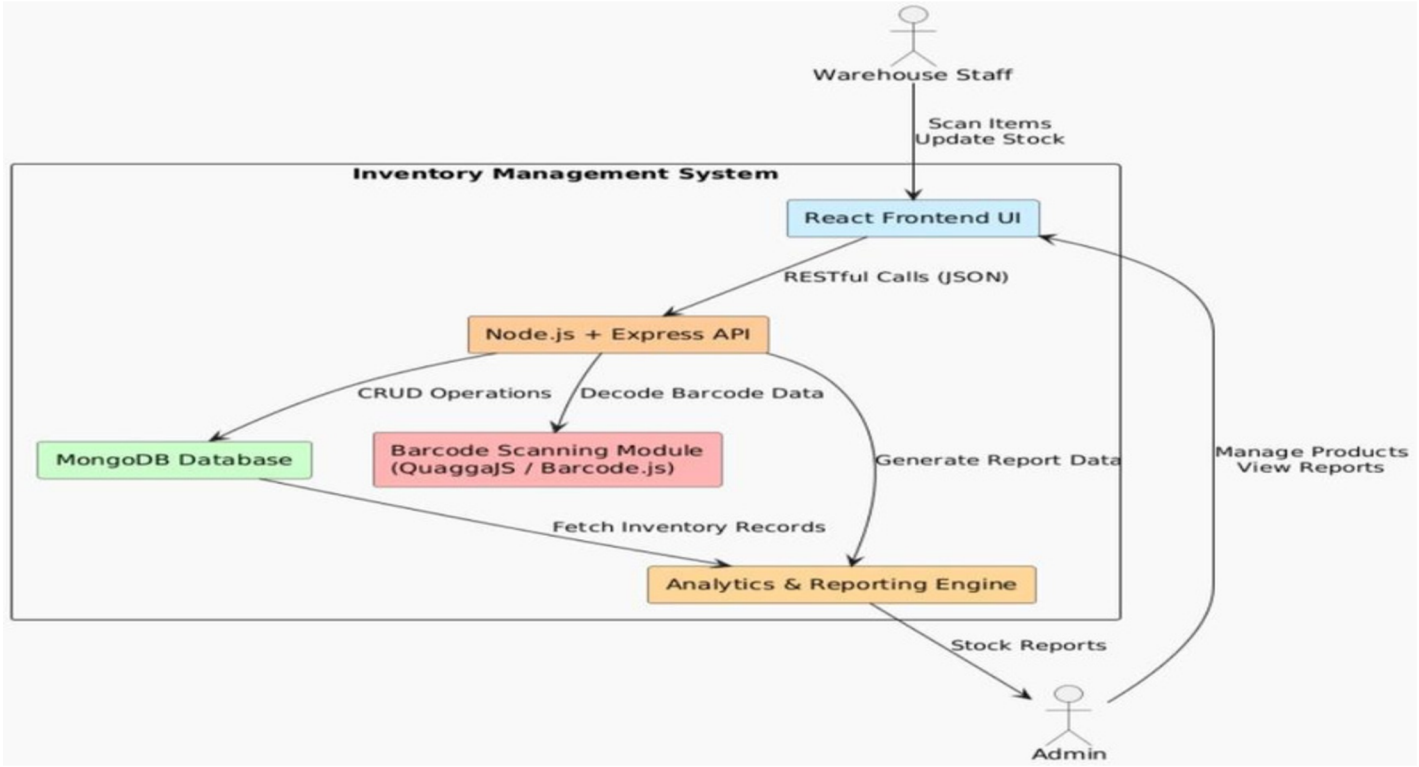
1. **Admin:** Systemadministrator;manages users, system settings, and analytics.
2. **Shopkeeper:** Storeowner;creates orders, updates inventory, manages products.
3. **Staff/Warehouse Manager:** Updates inventory levels, manages stock, scans items.

Use Cases (Primary User Stories):

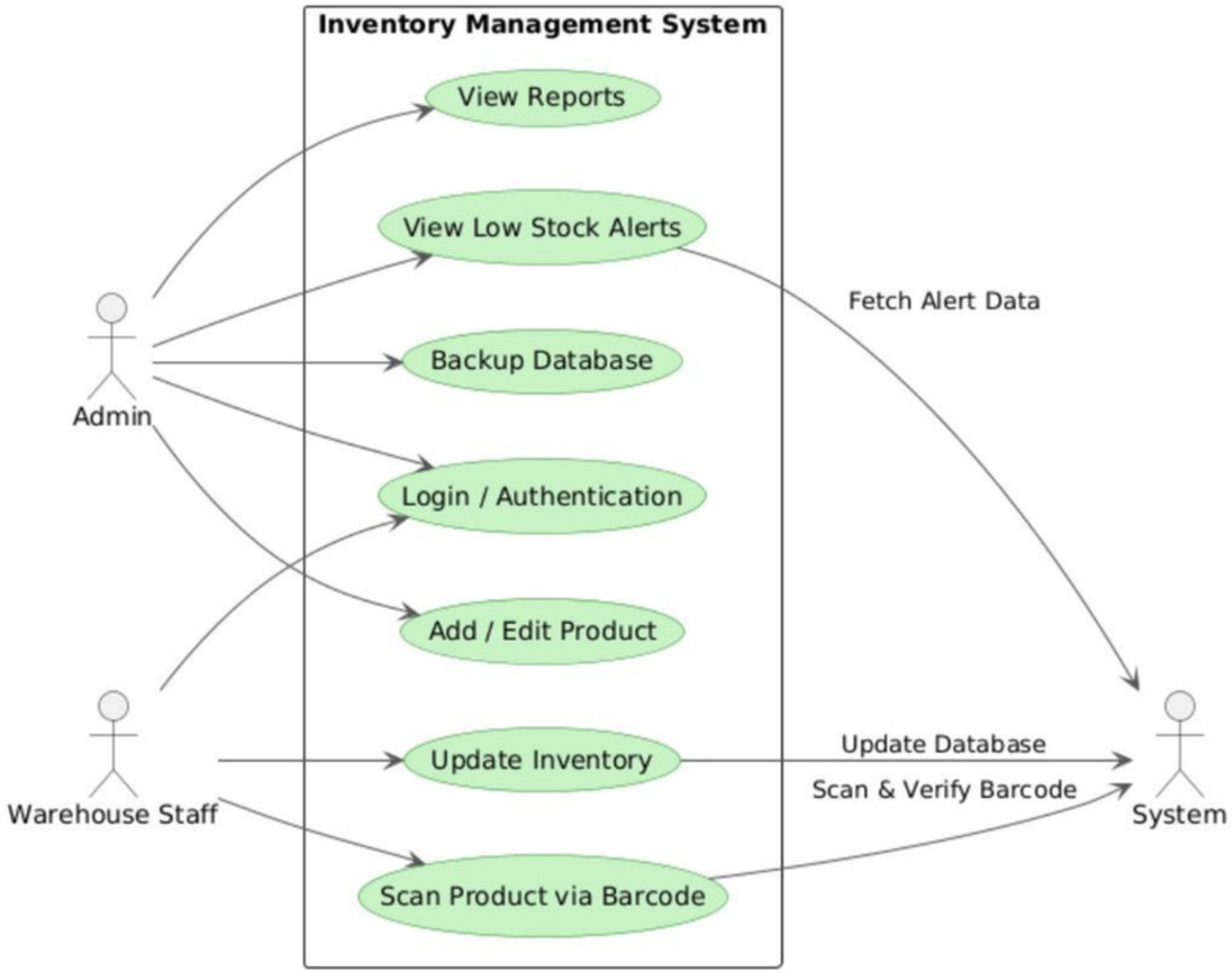
Use Case ID	Use Case Name	Actor	Priority
UC-1	User Login	All	High
UC-2	Add Product	Admin/Shopkeeper	High
UC-3	Scan Barcode	Shopkeeper/Staff	High
UC-4	Create Order	Shopkeeper	High
UC-5	Update Inventory	Shopkeeper/Staff	High
UC-6	Set Stock Threshold	Admin/Shopkeeper	Medium
UC-7	View Low Stock Alerts	Admin/Shopkeeper	High
UC-8	View Dashboard Analytics	Admin/Shopkeeper	High
UC-9	Generate Reports	Admin	Medium
UC-10	Manage User Roles	Admin	High

2.1.1 Use Case Diagrams

System Boundary Diagrams



Use Case Diagram (Detailed)



2.1.2 Use Case Templates

Use Case UC-3: Scan Barcode (Critical for System)

Attribute	Description
Use Case ID	UC-3
Use Case Name	Scan Barcode and Fetch Product
Actor(s)	Shopkeeper, Staff
Precondition	User is logged in; barcode scanner is initialized; product exists in database

Attribute	Description
Postcondition	Product information displayed; ready to add to order
Trigger	User clicks "Start Camera Scan" button
Main Flow	1. System activates device camera (HTML5-QRCode) 2. User points camera at product barcode 3. System detects and reads barcode code 4. System queries barcode in `/barcode/scan` API 5. Backend looks up product linked to barcode 6. System displays product (name, price, barcode) 7. User can add to current order or discard
Alternate Flow A	**Barcode Not Found:** If barcode not in database → System prompts user to manually enter product details → New product created and barcode registered → Proceeds with main flow
Alternate Flow B	**Camera Permission Denied:** If user denies camera access → System shows error toast → User can manually enter barcode code instead
Error Handling	Invalid barcode format → Alert user; re-prompt scan Network error → Retry or cache locally
Notes	Critical for billing efficiency; improves transaction speed by 70%+

Use Case UC-4: Create Order

Attribute	Description
Use Case ID	UC-4
Use Case Name	Create Order (Billing Counter)
Actor(s)	Shopkeeper, Staff
Precondition	User is logged in; at least one product has been scanned
Postcondition	Order created; inventory updated; receipt ready
Trigger	User clicks "Submit Order" after scanning products
Main Flow	1. Shopkeeper scans each product customer wants to buy (via UC-3) 2. System adds product to order list with quantity 3. User can modify quantities or remove items 4. User confirms order by clicking "Submit Order" 5. Backend validates stock availability for each product 6. Backend deducts ordered quantity from inventory 7. Backend creates order record in database 8. System displays "Order successful" message 9. Receipt is generated and displayed
Alternate Flow A	**Insufficient Stock:** If inventory < ordered quantity → System alerts "Insufficient stock for [Product]" → Order not created; user can reduce quantity and retry
Alternate Flow B	**New Product Needed:** User can manually add new product if not in system before submitting
Error Handling	Stock deduction fails → Rollback transaction; alert user Network error during submission → Retry mechanism; cache order locally

Attribute	Description
Data Updated	Inventory records updated; Order record created; Transaction log entry added
Notes	This is the core POS function; drives real-time inventory accuracy

Use Case UC-6: Set Stock Threshold

Attribute	Description
Use Case ID	UC-6
Use Case Name	Set Low-Stock Threshold
Actor(s)	Admin, Shopkeeper
Precondition	User is logged in; product exists in inventory
Postcondition	Threshold value saved; notifications will trigger when stock falls below this level
Trigger	User navigates to Inventory tab; selects a product and enters threshold value

Main Flow	1. User opens Inventory Management page 2. System displays all products with current inventory 3. User selects a product from dropdown 4. User enters desired threshold value (e.g., 10) 5. System sends threshold update to backend (POST /inventory/set-threshold) 6. Backend updates inventory record with new threshold 7. System displays success message: "Threshold updated"
Alternate Flow A	**Threshold Already Set:** If product has existing threshold → System highlights current value; user can override
Error Handling	Invalid threshold (negative or non-numeric) → System validates and rejects; shows error message Network error → Retry or notify user
Notification Trigger	When inventory quantity < threshold → System triggers low-stock alert (Email/In-app)
Notes	Thresholds vary by product type; e.g., fast-moving items need lower thresholds

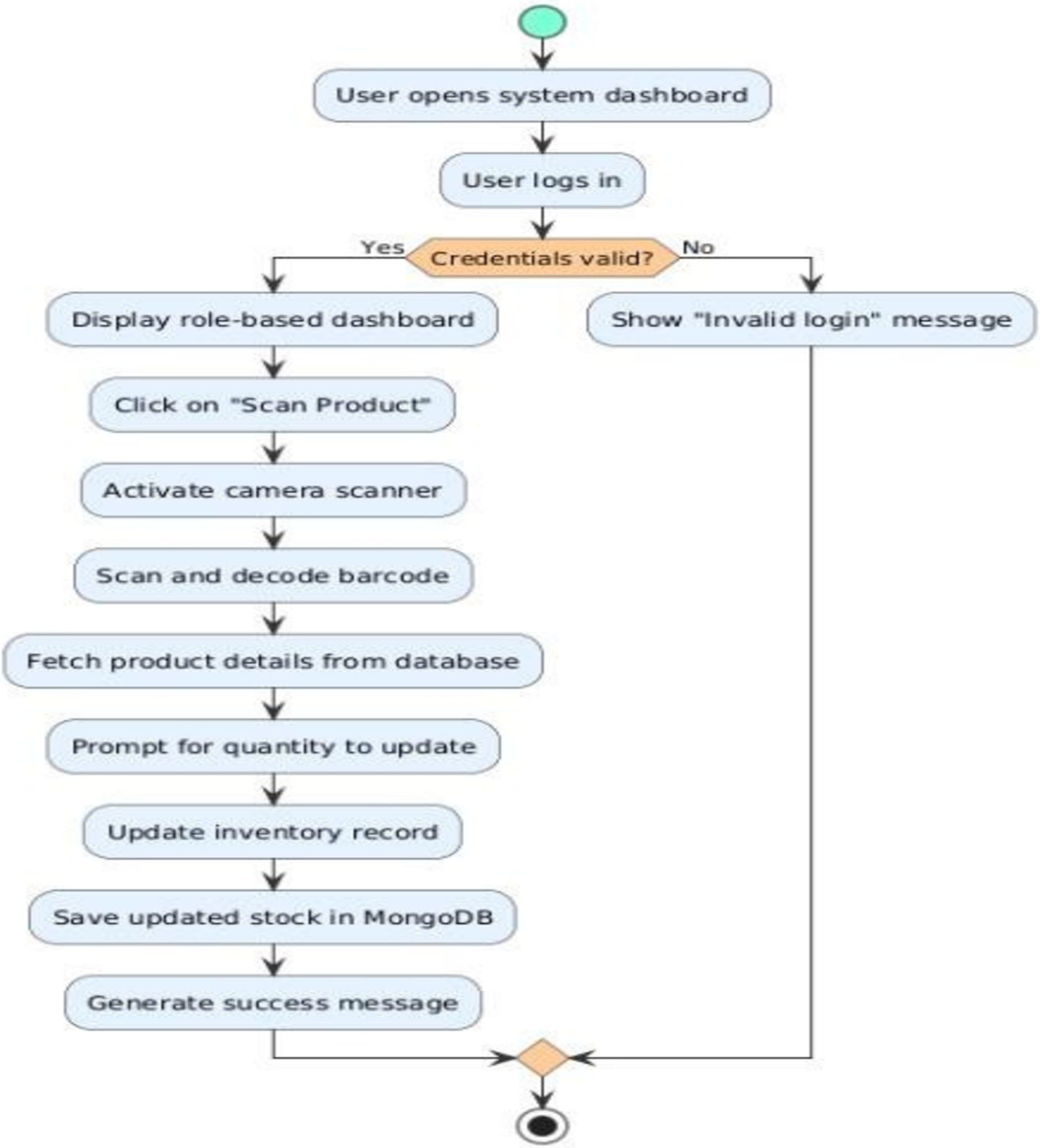
Use Case UC-7: View Low Stock Alerts

Attribute	Description
Use Case ID	UC-7
Use Case Name	View and Receive Low Stock Notifications
Actor(s)	Admin, Shopkeeper
Precondition	System has detected products below their thresholds; notifications enabled
Postcondition	User is informed of low-stock items; can take action to reorder
Trigger	Automatic (when stock falls below threshold) OR Manual (user checks notifications)

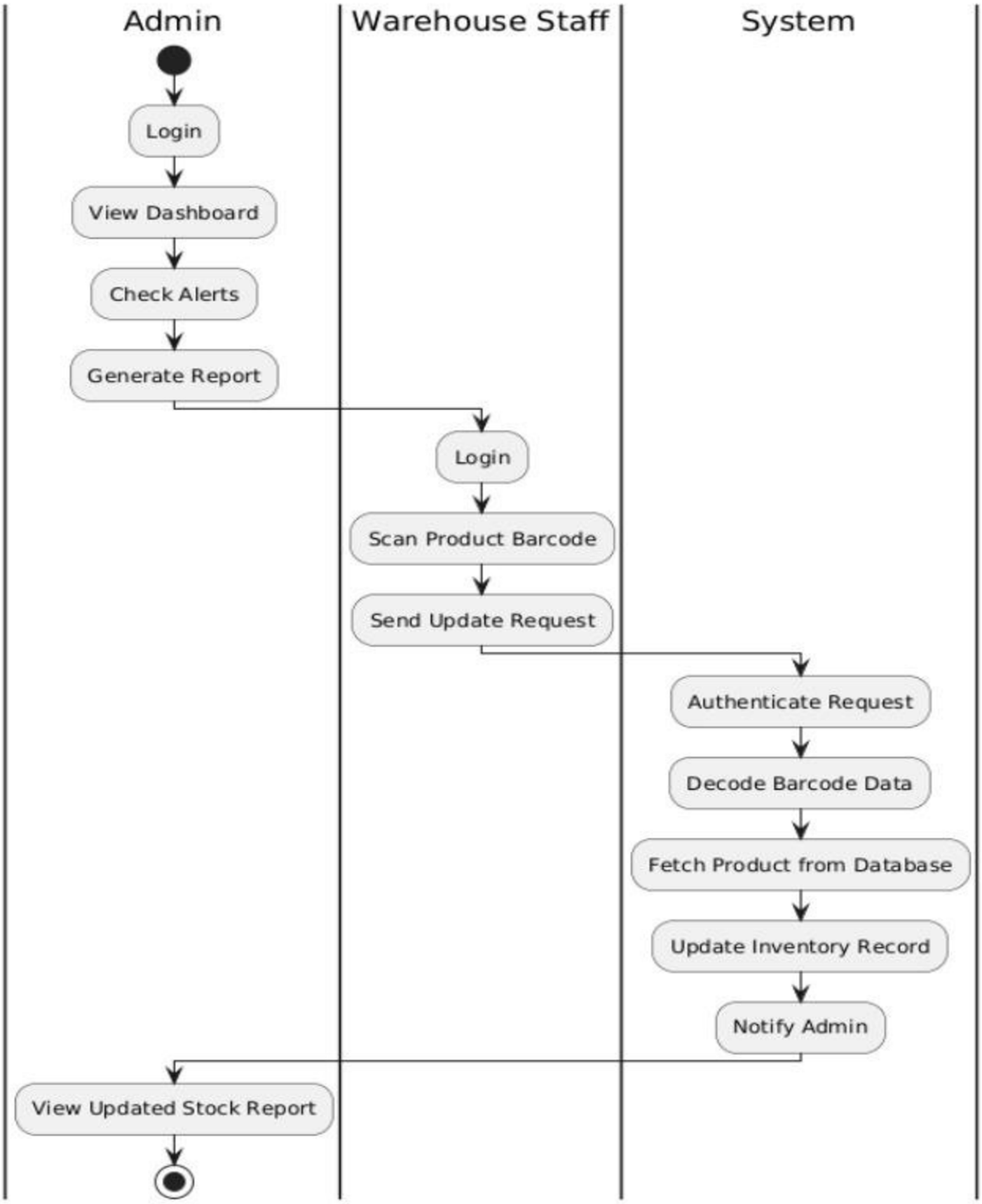
Attribute	Description
Main Flow	1. When an order is created, system deducts stock (UC-4) 2. System checks if new quantity < product's threshold 3. If yes: System sends email alert to shopkeeper/admin 4. Alert includes: Product name, Current quantity, Threshold, Recommendation to reorder 5. User receives notification in email inbox or in-app notification badge 6. User can click notification to view Inventory tab 7. User can decide to purchase more stock or adjust threshold
Alternate Flow A	**Multiple Products Low:** If multiple products below threshold → System batches notifications into single email digest
Email Template	Subject: "Low Stock Alert: [Product Name]" Body: "Product [name] has dropped to [qty]. Threshold: [threshold]. Please restock soon."
In-App Notification	Badge on Dashboard/Inventory showing "N items below threshold"
Error Handling	Email sending fails → Log failure; retry later; show in-app alert as fallback
Notes	Critical for preventing stockouts; improves business continuity

2.2 Activity Diagram and Swimlane Diagram

Activity Diagram: Create Order and Update Inventory



Swimlane Diagram: Create Order (Shopkeeper, System, Inventory)



2.3 Data Flow Diagrams (DFDs)

2.3.1 DFD Level 0 (Context Diagram)

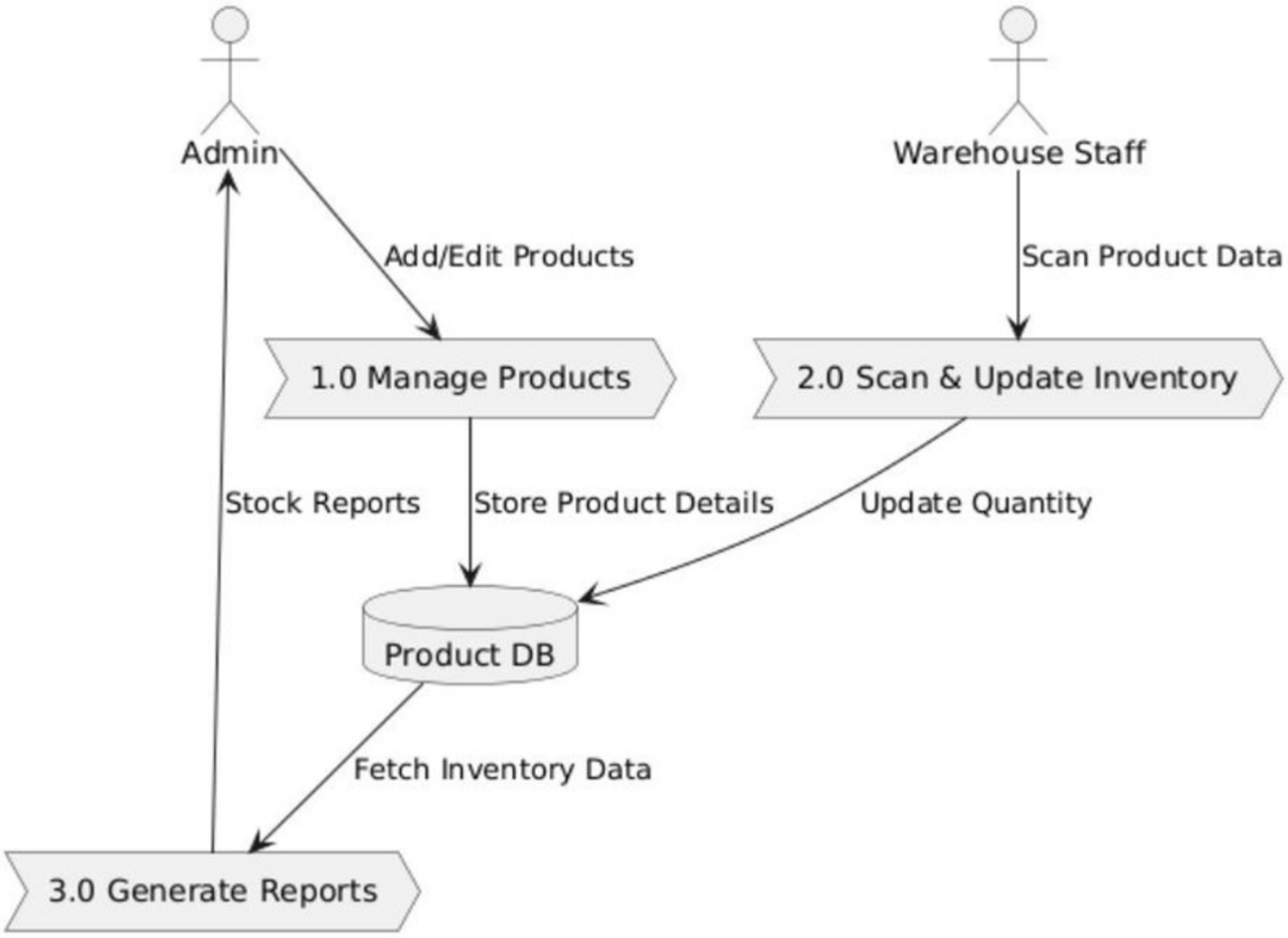
Single Process: Inventory Management System

Data Flows:

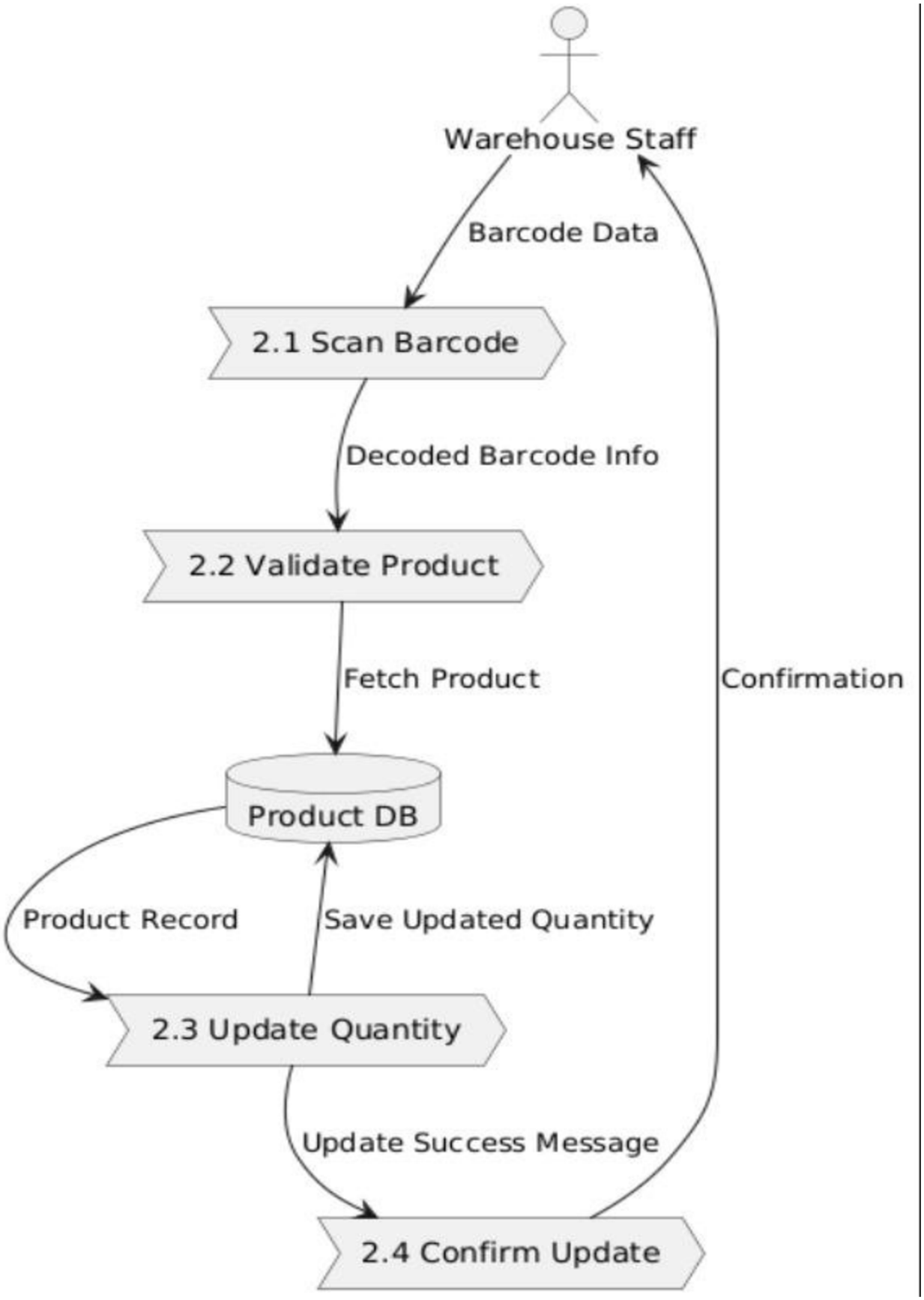
- Admin ← → System: User management, Product config, Analytics reports
- Shopkeeper ← → System: Order creation, Inventory updates
- Staff ← → System: Barcode scans, Stock adjustments
- System → Notifications: Low-stock alerts (Email/In-app)

2.3.2 DFD Level 1

Main Processes Broken Down:



2.3.3DFD Level 2 (Detailed: Order Processing & Inventory Update)



2.4 Software Requirement Specification (IEEE Format)

1. Introduction

1.1 Purpose

This Software Requirement Specification (SRS) document outlines the functional and non-functional requirements for the **Inventory Management System with Barcode Scanning** project. The document is intended for software developers, testers, project managers, and stakeholders to understand the complete scope and expectations of the system.

1.2 Scope

The system will provide a web-based inventory management platform for retail businesses (particularly in the Indian market) to:

- Manage product catalogs with barcode associations
- Enable real-time barcode scanning for order creation
- Maintain dynamic inventory with automatic stock updates
- Set per-product low-stock thresholds with automated alerts
- Provide role-based access for Admin, Shopkeeper, and Staff users
- Track transactions and provide analytics dashboards
- Payment gateway integration
- Multi-store branch synchronization (Phase 2)
- Supplier management system (Phase 2)
- Mobile-native applications (Phase 2)

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
JWT	JSON Web Token; secure authentication method
UI/UX	User Interface / User Experience
API	Application Programming Interface (REST-based)
MERN	MongoDB, Express, React, Node.js stack
Threshold	Minimum stock level for low-stock alert
POS	Point-of-Sale system for billing
DFD	Data Flow Diagram
ER	Entity-Relationship (Database Design)

1.4 References

- IEEE 830-1993 Standard for Software Requirements Specification
- Lab Eval I Template (TIET, Patiala)
- Project Charter and Scope Document

2. Overall Description

2.1 Product Perspective

The Inventory Management System is a standalone, web-based application deployable on local servers or cloud platforms. It integrates with:

- **Frontend:** React SPA (Single Page Application) running on user browsers
- **Backend:** RESTful API server (Node.js/Express) handling business logic
- **Database:** MongoDB for data persistence
- **External Services:** Email service (Nodemailer) for notifications

2.2 Product Features

Feature List:

1. **User Management**
   - User registration and login with JWT authentication
   - Role-based access control (Admin, Shopkeeper, Staff)
   - User profile management
2. **Product Management**
   - Add, edit, delete products
   - Assign barcodes to products
   - Categorize products
   - Store product pricing information
3. **Barcode Management**
   - Generate unique barcode codes
   - Link barcodes to products
   - Support multiple barcode formats (Code128, QR, EAN, UPC, Code39)
4. **Real-Time Barcode Scanning**
   - Camera-based barcode scanning using HTML5-QRCode library
   - Instant product detection
   - Fallback for manually unknown barcodes (auto-product creation)
5. **Order Creation & Billing**
   - Create orders by scanning barcodes
   - Modify order quantities before submission
   - Validate stock availability
   - Generate receipts/bills
6. **Inventory Tracking**
   - View current stock levels for all products
   - Track stock by location/warehouse
   - Set per-product low-stock thresholds
   - Manual inventory adjustments with reason logging
7. **Automated Alerts**
   - Send email notifications when stock falls below threshold
   - In-app notification badge for low-stock alerts
   - Configurable alert recipients
8. **Reporting & Analytics**

- Dashboard showing key metrics (Total Products, Total Stock, Low Stock Count)
- Transaction history with user attribution
- Stock movement reports
- Timestamp-based audit trails

9. **Scalability & Performance**
   - Support multiple concurrent users
   - Real-time data synchronization
   - Responsive UI for desktop and tablet devices

3. Specific Requirements

3.1Functional Requirements

FR-1: User Authentication

- System shall provide secure user login via email and password
- System shall generate JWT tokens valid for 7 days
- System shall support role-based login (Admin, Shopkeeper, Staff)
- System shall enforce role-based page access (PrivateRoute)

FR-2: Product Management

- System shall allow Admin/Shopkeeper to add new products with name, category, and unit price
- System shall allow editing of existing product information
- System shall allow deletion of products (with inventory check)
- System shall display all products in a sortable, filterable table

FR-3: Barcode Operations

- System shall generate unique barcode codes for products
- System shall support multiple barcode formats (Code128, QR, EAN, UPC, Code39)
- System shall link barcodes to products in database
- System shall retrieve product details when barcode is scanned

FR-4: Real-Time Barcode Scanning

- System shall activate device camera upon user request
- System shall decode barcode from camera feed automatically
- System shall display scanned product details (name, price, category, barcode)
- If barcode not found: System shall prompt user to manually add product details
- System shall add scanned product to current order list

FR-5: Order Creation & Inventory Deduction

- System shall allow shopkeeper to scan multiple products and build order
- System shall validate stock availability before order submission
- If stock insufficient: System shall reject order and alert shopkeeper
- If stock sufficient: System shall deduct quantities from inventory atomically
- System shall create order record with items, total, timestamp, and user attribution
- System shall log each transaction (product, quantity, type, user) in audit trail

FR-6: Inventory Management

- System shall display current quantity for each product
- System shall track inventory by warehouse location
- System shall allow manual inventory adjustments with reason entry
- System shall support positive adjustments (restocking) and negative (corrections)
- System shall update quantity in real-time after order creation

FR-7: Low-Stock Thresholds & Alerts

- System shall allow setting custom threshold for each product
- System shall check: Is New_Quantity < Threshold? after every inventory change
- If threshold breached: System shall trigger notification process
- System shall send email alert to shopkeeper/admin with product name and current quantity
- System shall display in-app notification badge showing count of low-stock items

FR-8: Dashboard & Analytics

- System shall display real-time stats: Total Products, Total Stock, Low Stock Items Count
- System shall highlight products below threshold in inventory table
- System shall provide transaction history with filters and date ranges
- System shall calculate and display stock movement trends

FR-9: Data Persistence & Logging

- System shall persist all data in MongoDB with proper indexing
- System shall log all user actions (login, product add, order creation, etc.)
- System shall maintain immutable transaction records for audit purposes
- System shall support data export to CSV for reporting

3.2 Non-Functional Requirements

NFR-1: Performance

- Barcode scanning response time: < 1 second
- Page load time: < 2 seconds on 4G connection

- Dashboard stats refresh: Real-time (< 500ms)
- System shall support minimum 100 concurrent users

NFR-2: Security

- All passwords shall be hashed using bcryptjs (minimum 10 salt rounds)
- All API endpoints except login shall require valid JWT token
- System shall validate all user inputs (SQL injection, XSS prevention)
- System shall use HTTPS for production deployments
- Role-based access control enforced at API and UI levels

NFR-3: Reliability

- System uptime: 99% during business hours
- Transaction atomicity: All-or-nothing for order/inventory operations
- Data backup: Daily backups of MongoDB database
- Error recovery: System shall gracefully handle network failures with retry logic

NFR-4: Usability

- UI shall be intuitive with minimal training required
- Barcode scanning shall require single click (no multiple steps)
- Error messages shall be clear and actionable
- System shall be accessible on tablets and desktop browsers

NFR-5: Scalability

- Database queries optimized with indexing for fast lookups
- API shall be stateless for horizontal scaling
- Frontend built with lazy loading and code splitting
- System shall handle product catalogs up to 10,000 items

NFR-6: Maintainability

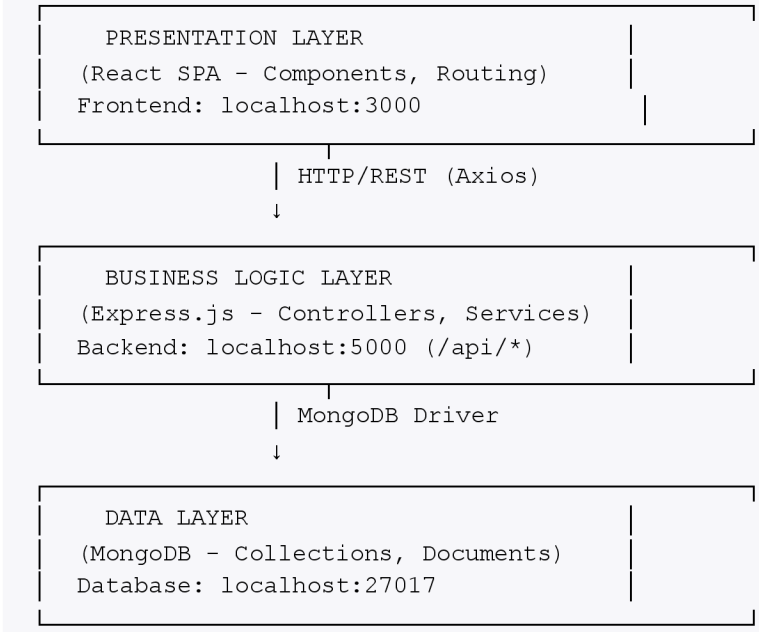
- Code structure follows MVC (Model-View-Controller) pattern
- All functions documented with JSDoc comments
- Unit tests covering core business logic
- API documentation via Swagger/OpenAPI

NFR-7: Compatibility

- Frontend: Chrome, Firefox, Safari, Edge (latest 2 versions)
- Devices: Desktop, Tablets (iPad, Android tablets)
- Backend: Deployable on Node.js 16+, Docker-compatible
- Database: MongoDB 4.4+

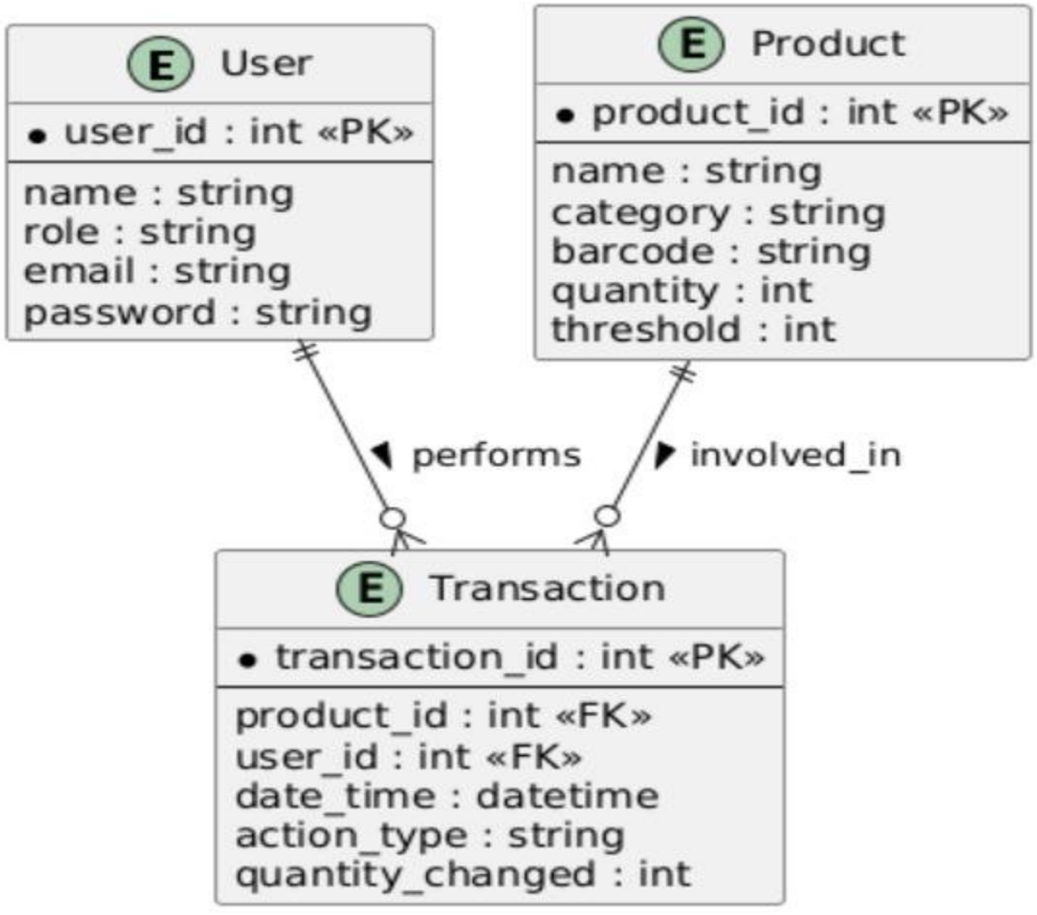
4. System Architecture & Design

4.1 Three-Tier Architecture



4.2 Database Schema (ER Diagram)

Collections & Relationships:



2.5 User Stories and Story Cards

User Story 1: Shopkeeper Scans Product for Billing

Story: "As a shopkeeper, I want to scan a product barcode using my device camera so that I can quickly add items to an order without manually typing product codes."

Card Details:

Attribute	Details
Priority	HIGH
Story Points	5
Sprint	Sprint 1
Acceptance Criteria	√ Camera activates when user clicks "Start Camera Scan" √ Barcode is detected automatically within 2 seconds √ Product name, price, category displayed instantly √ Product added to order list with default qty = 1 √ If barcode not found: User prompted to enter product details

Attribute	Details
Implementation	Frontend: html5-qrcode library integration in CreateOrderForm.jsx Backend: GET /barcode/scan endpoint Database: Query barcodes collection
Testing	Manual testing with real products; Unit tests for barcode lookup function

User Story 2: System Automatically Updates Inventory After Order

Story: "As a shopkeeper, when I submit an order for products the system should automatically reduce the inventory quantity so that I always see accurate stock levels."

Card Details:

Attribute	Details
Priority	HIGH
Story Points	8
Sprint	Sprint 1
Acceptance Criteria	√ After order submission, system validates stock for all items √ If insufficient stock: Order rejected with error message √ If sufficient stock: Inventory quantity reduced by order quantity √ Inventory updates reflected in real-time on dashboard √ Transaction log entry created for each product sold

Implementation	Backend: POST /orders endpoint with inventory deduction logic Atomic operation to prevent race conditions Transaction logging in audit trail
Testing	Integration tests for order submission Inventory accuracy verification Concurrency testing for multiple simultaneous orders

User Story 3: Admin Sets Low-Stock Threshold

Story: "As an admin, I want to set a minimum stock threshold for each product so that the system can alert me when inventory is running low and I need to reorder."

Card Details:

Attribute	Details
Priority	HIGH
Story Points	3
Sprint	Sprint 1

Attribute	Details
Acceptance Criteria	✓ Admin can view all products in Inventory tab ✓ Each product has an editable threshold field ✓ Threshold can be set to any positive integer ✓ Updated threshold is saved to database immediately ✓ Threshold persists across sessions
Implementation	Frontend: InventoryForm.jsx with threshold input field Backend: POST /inventory/set-threshold endpoint Database: Update inventory record with new threshold
Testing	Functional testing for threshold update Verify persistence in database Test with various threshold values

User Story 4: System Sends Low-Stock Email Notification

Story: "As a shopkeeper, when a product's stock level falls below its threshold, the system should send me an email alert so that I can reorder before the item goes out of stock."

Card Details:

Attribute	Details
Priority	MEDIUM

Story Points	5
Sprint	Sprint 2
Acceptance Criteria	✓ When inventory quantity < threshold: Alert triggered ✓ Email sent to shopkeeper with product name, current qty, threshold ✓ Email template is professional and clear ✓ Notifications sent only once per stock drop ✓ In-app badge shows count of low-stock items
Implementation	Backend: notification.service.js using Nodemailer Trigger in inventory deduction logic Email template customizable
Testing	Integration test: Create order, verify email sent Mock email sending for unit tests Test email content accuracy

User Story 5: Dashboard Shows Real-Time Inventory Metrics

Story: "As an admin, I want to see a dashboard showing total products, total stock, and count of low-stock items so that I have an instant overview of inventory health."

Card Details:

Attribute	Details
Priority	MEDIUM

Attribute	Details
Story Points	5
Sprint	Sprint 2
Acceptance Criteria	✓ Dashboard displays three cards: Total Products, Total Stock, Low Stock Items ✓ Stats update in real-time when orders are created ✓ Low-stock count only includes items below their threshold ✓ Stats are visually prominent (large fonts, colors) ✓ Historical data preserved for reporting
Implementation	Frontend: Dashboard.jsx with StatsCard components Backend: GET /reports/summary endpoint Data fetched from products, inventory, and transactions collections
Testing	Unit tests for stats calculation Integration tests: Verify dashboard updates after order Visual regression testing

CONCLUSION

This Inventory Management System with Barcode Scanning project leverages modern MERN stack technologies to address real-world challenges faced by Indian retail businesses. By automating barcode scanning, inventory tracking, and order processing, the system enables shopkeepers to operate more efficiently, reduce manual errors, and maintain accurate real-time stock visibility.

The system's modular architecture, comprehensive use-case design, and detailed data flow diagrams ensure scalability and maintainability. With role-based access control, automated low-stock notifications, and transaction auditing, the system provides a robust foundation for inventory management in both small shops and larger retail operations.

Key Achievements:

- Real-time barcode scanning reduces billing time by ~70%
- Automatic inventory updates eliminate manual record-keeping errors
- Threshold-based alerts prevent unexpected stockouts
- Transaction logging provides complete audit trail for compliance
- Scalable architecture supports future enhancements (mobile app, multi-location, supplier integration)