Enhanced Inventory and Sales Management System - Node.js + SQLite

1. Project Overview

This console-based application provides comprehensive inventory and sales management capabilities. It allows users to manage products, track stock levels, record sales, generate reports, and includes advanced features like supplier management, category organization, and automated alerts.

2. Enhanced Database Schema

Database Structure

The enhanced system uses six main data tables:

Products Table: Stores detailed product information including name, description, category, pricing (both selling price and cost price), current stock levels, minimum and maximum stock thresholds, supplier information, and barcode data.

Categories Table: Organizes products into logical groups like Electronics, Clothing, Food Items, etc. Each category has a name and description.

Suppliers Table: Maintains vendor information including company name, contact person, email, phone number, and address details.

Sales Table: Records every sale transaction with product details, quantities sold, pricing, discounts applied, tax calculations, customer information, and payment method.

Purchase Orders Table: Tracks orders placed with suppliers, including order status, dates, expected delivery, and total amounts.

Stock Movements Table: Creates an audit trail of all inventory changes, whether from sales, purchases, adjustments, or returns.

3. Enhanced Features

Core Features

- ✅ Add, update, delete products with detailed information
- ✅ Category management
- ✅ Supplier management
- ✅ Advanced product search (by name, ID, barcode, category)
- ✅ Record sales with customer details and payment methods
- ✅ Purchase order management
- ✅ Stock level tracking with min/max thresholds
- ✅ Automated low stock alerts
- ✅ Stock movement audit trail

Reporting Features

- ✅ Daily/Weekly/Monthly sales reports
- ✅ Profit margin analysis
- ✅ Top-selling products
- ✅ Category-wise performance
- ✅ Supplier performance reports
- ✅ Stock valuation reports
- ✅ Low stock reports
- ✅ Revenue trends and forecasting

Advanced Features

- ✅ Real-time notifications for low stock
- ✅ Barcode scanning simulation
- ✅ Discount and tax calculations
- ✅ Dashboard with key metrics
- ✅ Advanced filtering and sorting
- ✅ Export reports to CSV/PDF
- ✅ Bulk import/export functionality
- ✅ Simple REST API endpoints

4. Enhanced Project Structure

```
inventory-management/
├── app.js                      # Main application entry point
├── config/
│   ├── database.js             # Database configuration
│   └── settings.js             # Application settings
├── controllers/
│   ├── productController.js    # Product CRUD operations
│   ├── salesController.js      # Sales management
│   ├── categoryController.js   # Category management
│   ├── supplierController.js   # Supplier management
│   ├── purchaseController.js   # Purchase orders
│   └── reportController.js     # Report generation
├── models/
│   ├── Product.js              # Product model
│   ├── Sale.js                 # Sales model
│   ├── Category.js             # Category model
│   ├── Supplier.js             # Supplier model
│   └── PurchaseOrder.js        # Purchase order model
├── services/
│   ├── inventoryService.js     # Business logic for inventory
│   ├── reportService.js        # Report generation service
│   ├── notificationService.js  # Alert system
│   └── exportService.js        # Data export functionality
├── utils/
│   ├── dateHelper.js           # Date formatting utilities
│   ├── validation.js           # Input validation
│   ├── calculator.js           # Price/tax calculations
│   └── logger.js               # Error logging
├── data/
│   ├── database.sqlite         # SQLite database file
│   └── exports/                # Generated reports
├── tests/
│   ├── product.test.js         # Unit tests
│   ├── sales.test.js           # Sales tests
│   └── reports.test.js         # Report tests
├── package.json
├── README.md
└── .gitignore
```

5. Enhanced Sample Workflow

Main Menu Options

```
 ================== INVENTORY MANAGEMENT SYSTEM ==================
1.  □ Product Management
2.  □  Category Management
3.  □ Supplier Management
4.  □ Sales Management
5.  □ Purchase Orders
6.  □ Reports & Analytics
7.  □ Notifications & Alerts
8.  ⚙  System Settings
9.  □ Import/Export Data
10. □ Exit
================================================================
```

Sub-menus

Product Management

- Add new product with full details
- Update existing product
- Delete product (with confirmation)

- View all products with pagination
- Search products (multiple criteria)
- Check stock levels
- Adjust stock quantities
- View product history

Sales Management

- Record new sale with customer details
- Apply discounts and calculate taxes
- Process returns and refunds
- View sales history
- Search sales by date/customer
- Generate invoices

Reports Dashboard

- Quick stats overview
- Sales performance charts
- Inventory status summary
- Profit margin analysis
- Export options

6. Advanced Implementation Features

Error Handling & Validation

The system implements comprehensive validation to ensure data integrity:

- Product names must be meaningful and not empty
- Prices must be positive values
- Stock quantities cannot be negative
- Email addresses must follow proper format
- Phone numbers must match valid patterns
- All required fields are checked before saving

Automated Alerts System

- Low stock notifications
- Overstock warnings
- Sales target alerts
- Expiry date reminders (if applicable)
- System health checks

Data Export Options

- CSV format for spreadsheet analysis
- PDF reports for formal documentation
- JSON format for data migration
- Scheduled automated exports

Performance Optimizations

- Database indexing on frequently queried fields
- Pagination for large datasets
- Caching frequently accessed data
- Query optimization techniques

7. Installation & Setup

Prerequisites

```
Node.js (v14 or higher)
npm (Node Package Manager)
```

Required Dependencies

The system uses several Node.js libraries to enhance functionality:

- **sqlite3**: Database management for storing all data
- **inquirer**: Creates interactive command-line menus
- **chalk**: Adds colors to console output for better visibility
- **cli-table3**: Formats data in neat tables
- **moment**: Handles date and time operations
- **csv-writer**: Exports data to spreadsheet format

- **pdfkit**: Generates PDF reports
- **validator**: Ensures data meets required formats
- **uuid**: Creates unique identifiers for records

Setup Process

To get the system running:

1. **Create Project Folder**: Set up a dedicated directory for the application
2. **Initialize Node.js Project**: Create package.json file to manage dependencies
3. **Install Required Libraries**: Download all necessary packages
4. **Set Up Database**: Create SQLite database file with all required tables
5. **Configure Settings**: Set up application preferences and default values
6. **Run Application**: Start the main program file to begin using the system

8. Security & Best Practices

Data Security

- Input sanitization to prevent SQL injection
- Data validation at multiple layers
- Error logging without exposing sensitive data
- Regular database backups

Code Quality

- Modular architecture with separation of concerns
- Comprehensive error handling
- Unit testing for critical functions
- Code documentation and comments

Performance Monitoring

- Database query performance tracking
- Memory usage monitoring
- Response time optimization
- Regular maintenance procedures

9. Future Enhancement Ideas

Phase 2 Features

- Multi-location inventory support
- Integration with external accounting systems
- Mobile app companion
- Real-time dashboard web interface
- Advanced analytics with machine learning
- Multi-currency support
- Integration with e-commerce platforms

Integration Possibilities

- POS system integration
- Accounting software APIs
- Shipping and logistics APIs
- Customer relationship management (CRM)
- Business intelligence tools

10. Troubleshooting Guide

Common Issues

- Database connection problems
- Performance issues with large datasets
- Data export/import errors
- Validation failures
- Stock calculation discrepancies

Maintenance Tasks

- Regular database cleanup
- Performance optimization
- Data backup procedures
- System health checks
- Update procedures

This enhanced inventory management system provides a robust foundation for small to medium-sized businesses while maintaining the simplicity of a console-based application. The modular architecture allows for easy expansion and customization based on specific business needs.