Project Documentation: Inventory Management Frontend

# 1. Project Overview

The Inventory Management Frontend project is designed to provide a user-friendly interface for managing stock, tracking products, and monitoring inventory levels efficiently. The system aims to minimize manual effort, reduce errors, and enhance real-time visibility of inventory across an organization.

## Purpose:

* To allow users to easily add, view, update, and delete inventory items.
* To provide real-time stock status and generate alerts for low/out-of-stock items.
* To integrate with backend services for data persistence and reporting.

## Scope:

* Product listing with detailed information.
* CRUD (Create, Read, Update, Delete) operations on products.
* Search, filter, and sorting functionalities for faster navigation.
* Dashboard for summaries like low stock alerts, total inventory value, etc.
* Role-based access (admin, staff) for secure operations.

## Target Users:

* Warehouse managers.
* Retail shop owners.
* Supply chain and logistics staff

# 2. System Architecture & Technology Stack

## Architecture:

The frontend follows a component-based architecture to ensure modularity and reusability. It interacts with the backend APIs using REST calls.

## Technology Stack:

* Frontend Framework: React.js
* Programming Language: JavaScript / TypeScript
* Styling: Tailwind CSS / Material-UI
* State Management: Redux / Context API
* Routing: React Router
* API Communication: Axios / Fetch API
* Build Tools: Webpack / Vite
* Version Control: Git + GitHub/GitLab

# 3. Features

# Authentication & Authorization

* Login system for different user roles.
* Admin access to all operations; staff access limited.

## Dashboard

* Overview of inventory status.
* Key metrics like low stock alerts, out-of-stock items, total products, and overall stock value.

## Search & Filters

* Search by product name, category, or SKU.
* Filter by category, supplier, or stock levels.
* Sorting options (by price, stock quantity, etc.).

## 4.Reports & Exports

* Export inventory details to CSV/Excel.
* Generate stock level reports.

## 5. Responsiv.

* Optimized for desktop, tablet, and mobile devices.

# 4. Data Models

## Product Model Example:

{

"id": "P1234",

"name": "Laptop",

"sku": "LAP123",

"category": "Electronics",

"quantity": 25,

"price": 60000,

"supplier": "Tech Supplier Ltd",

"lastUpdated": "2025-09-18"

}

# 5. API Endpoints

GET /api/items → Fetch all products.

GET /api/items/:id → Fetch single product by ID.

POST /api/items → Add a new product.

PUT /api/items/:id → Update product details.

DELETE /api/items/:id → Delete a product.

GET /api/items?filter=low\_stock → Fetch items below threshold.

# 6. Installation & Setup

## Prerequisites:

* Node.js and npm installed.
* Git for version control.

## Steps:

* git clone <repository\_url>
* cd inventory-management-frontend
* npm install
* npm start

## Production Build:

* npm run build

# 7. Directory Structure

/src

/components → Reusable UI components (buttons, forms, tables)

/pages → Page-level components (Dashboard, ProductList, ItemForm)

/services → API calls (Axios services)

/store → State management (Redux/Context)

/assets → Images, icons, static files

/styles → CSS/Tailwind configurations

/utils → Helper functions

/public

package.json

README.md

.env

# 8. Testing

* Unit Testing: Components tested with Jest / React Testing Library.
* Integration Testing: API and UI flow tested with Cypress.
* Manual Testing: Form validation, CRUD operations, search/filter performance.

# 9. Deployment

* Hosting Platforms: Netlify, Vercel, or AWS S3 + CloudFront.
* CI/CD Pipeline: GitHub Actions for automated build and deploy.
* Environment Variables:
* REACT\_APP\_API\_URL=https://backend.api url.

# 10. Security Considerations

* JWT tokens for secure API calls.
* Input validation to prevent XSS and SQL injection.
* HTTPS enforced for secure communication.
* Role-based access control (admin vs staff).

# 11. Performance & Optimization

* Code splitting for faster load.
* Lazy loading of components.
* Caching API respondes.
* Image optimization.

# 12. Future Enhancements

* Barcode scanning for faster item entry.
* Bulk import/export via Excel.
* Advanced analytics dashboard.
* Multi-language support.
* Notifications via email/SMS for low stock.