**Project Documentation**

**Store Manager Keep Track In Inventory**

1. **Introduction**
   * **Project Title**: STORE MANAGER KEEP TRACK IN INVENTORY
   * **Team ID** : NM2025TMID47718
   * **Team leader &ID** : S.PRABA [& prabasankar199@gamil.com](mailto:%26%20prabasankar199@gamil.com)

# Team Member:

1. S. POOVIZHI & [spoovizhi8107@gmail.com](mailto:spoovizhi8107@gmail.com)
2. N. PRAVESHIKA [&npraveshika2007@gmail.com](mailto:%26npraveshika2007@gmail.com)
3. S.RAMYA SRI & [ramyasrineet23@gmail.com](mailto:ramyasrineet23@gmail.com)

# Project Overview

* **Purpose:** In a **Store Manager (Inventory Management System)**, the **purpose** is to help a store (retail, wholesale, or warehouse) run smoothly by keeping track of goods, stock levels, and sales.

# FEATURES:

* + Inventory Management
  + Stock Movement Tracking
  + Real-Time Reporting
  + Purchase Order Management

# Client Setup

* bash
* cd client
* npm install
* This installs all necessary dependencies for the frontend (React, Vue, etc.).

# Server Setup

* bash
* cd ../server
* npm install
* This installs backend dependencies (Express, Mongoose, etc.).
* If you want, I can help you add more setup steps like running the server, connecting to a database, or even writing a README file. Just let me know what tech stack you're using (Node.js, MongoDB, etc.) and what you'd like to include.

# Folder Structure

Code

SB-Works/

├── client/ # React frontend

│ ├── components/ # Reusable UI components

│ └── pages/ # Page-level components (e.g., Home, Dashboard)

│

└── server/ # Node.js backend

├── routes/ # API route definitions

├── models/ # Mongoose/ORM models for DB schema

└── controllers/ # Logic for handling requests

### Suggestions for Expansion

If you're planning to scale this project, consider adding:

* client/assets/ for images, fonts, and styles
* server/middleware/ for auth, logging, error handling
* server/config/ for environment variables and DB setup
* README.md at the root for setup instructions and documentation.

# Running the Application

Frontend (React) bash

cd client npm start

Starts the React development server. Default port: [http://localhost:3000](http://localhost:3000/) Backend (Node.js)

bash

cd server npm start

* + Launches the Node.js server (Express or similar).
  + Make sure your backend is configured to run on a different port (e.g., 5000) to avoid conflicts with the frontend.

### Access the App

Open your browser and visit: [http://localhost:3000](http://localhost:3000/) This loads the frontend, which communicates with the backend via API calls.

# Access:

### User:

* /api/user/register
* /api/user/login

### Project :

* /api/projects/create
* /api/projects/:id • Applications: /api/apply

### Chart:

* /api/chart/send
* /api/chat/:userId

# Authentication

### JWT-Based Authentication

**JSON Web Tokens (JWT)** are a compact, URL-safe way of representing claims between two parties. They're commonly used for authentication in web apps.

### Login Flow:

* + - User submits credentials (e.g., email and password).
    - Server verifies credentials and generates a JWT.
    - JWT is sent back to the client and stored (usually in localStorage or a cookie).
    - For subsequent requests, the client includes the JWT in the Authorization header.

### Token Structure:

* + - **Header**: Specifies the signing algorithm (e.g., HS256).
    - **Payload**: Contains user data and claims (e.g., user ID, role).
    - **Signature**: Ensures the token hasn’t been tampered with.

### Security Tips:

* + - Use HTTPS to prevent token interception.
    - Set token expiration (exp) to limit session duration.
    - Consider using refresh tokens for long-lived sessions.

### Middleware for Route Protection

Middleware acts as a gatekeeper for your routes, verifying the JWT before allowing access.

### How It Works:

* + - Middleware intercepts incoming requests to protected routes.
    - It checks for the presence and validity of the JWT.
    - If valid, it allows the request to proceed.
    - If invalid or missing, it returns an error (e.g., 401 Unauthorized).

### Example (Node.js/Express):

js

const jwt = require('jsonwebtoken');

function authenticateToken(req, res, next) {

const token = req.headers['authorization']?.split(' ')[1]; if (!token) return res.sendStatus(401);

jwt.verify(token, process.env.JWT\_SECRET, (err, user) => { if (err) return res.sendStatus(403);

req.user = user; next();

});

}

// Usage

app.get('/private', authenticateToken, (req, res) => { res.send('This is a protected route');

});

# User Interface

### Landing Page

**Purpose:** First impression, marketing, and user conversion **Key Features:**

* + Eye-catching hero section with a clear value proposition
  + Call-to-action buttons (e.g., “Join as Freelancer” / “Post a Project”)
  + Testimonials or success stories
  + Overview of platform features
  + Navigation to login/signup

### Freelancer Dashboard

**Purpose:** Workspace for freelancers to manage their profile and projects **Key Features:**

* + Profile overview and edit options
  + Active, pending, and completed projects
  + Messaging center for client communication
  + Earnings and withdrawal section
  + Notifications and alerts

### Admin Panel

**Purpose:** Control center for platform management **Key Features:**

* + User management (freelancers and clients)
  + Project moderation and approval
  + Financial tracking and analytics
  + Reports and insights dashboard
  + Settings for platform rules, fees, and content

### Project Details Page

**Purpose:** Showcase individual project info and facilitate bidding or collaboration **Key Features:**

* + Project title, description, budget, and deadline
  + Client profile and rating
  + Proposal submission form
  + Comments or Q&A section
  + Tags or categories for filtering

# Testing

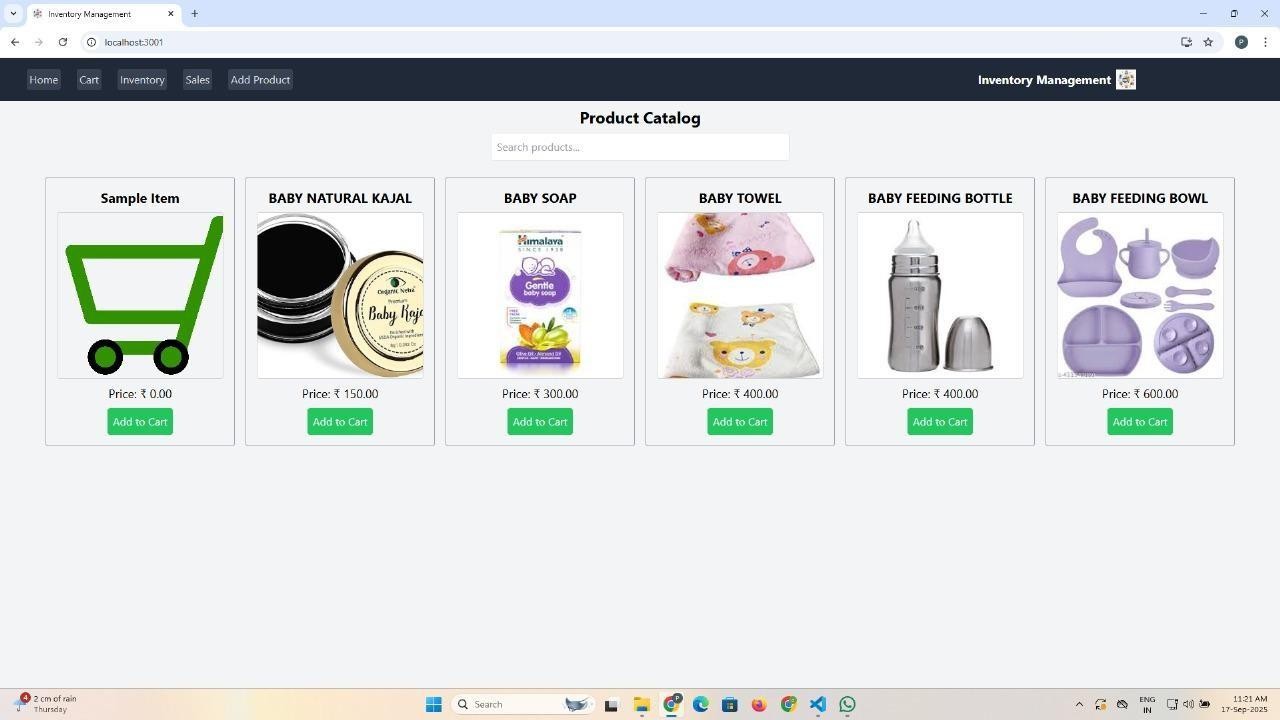
* + Conducted **manual testing** at key development milestones to ensure feature stability and performance
  + Utilized tools such as **Postman** for API validation and **Chrome DevTools** for front- end debugging and performance analysis

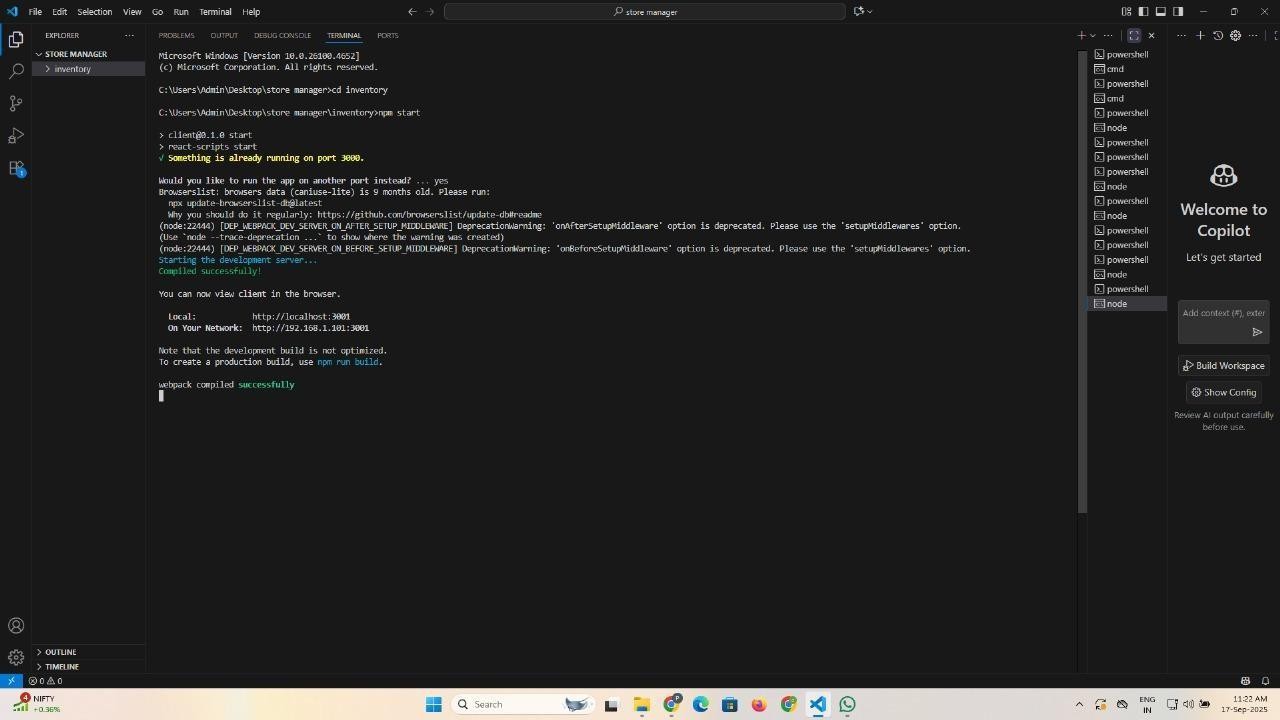
If you're compiling this for a resume or portfolio, I can help tailor it even further based on your role (QA, developer, etc.) or the project context. Want to expand this into a full section or keep refining other points?

**Existing solutions**

* + Git hub link [h t t p s : / / g i t h u b . c o m / p r a b a s a n k a r 1 9 9 - p n g / s t o r e -](https://github.com/prabasankar199-png/store-manager-keep-track-in-inventory) [m a n a g e r - k e e p - t r a c k - in - i n v e n t o r y](https://github.com/prabasankar199-png/store-manager-keep-track-in-inventory)

## Screenshots or Demo





**Known Issues Software or app issues (e.g., bugs in a program or system)?**

Project documentation (like a section in a report or release notes)? Product development (issues in a prototype or rollout)?

Website or platform problems?

If you're drafting a “Known Issues” sec

## Future Enhancements

1. Performance Improvements

* + Optimize system speed and responsiveness
  + Reduce latency or load times
  + Enhance scalability for larger user bases

## Feature Expansion

* + Introduce new functionalities based on user feedback
  + Integrate with third-party tools or platforms
  + Add multilingual support or accessibility features

## User Experience (UX) Upgrades

* + Refine interface design for better usability
  + Implement personalization or adaptive UI
  + Improve onboarding and help systems

## Security & Compliance

* + Strengthen data encryption and privacy protocols
  + Ensure compliance with evolving regulations
  + Add multi-factor authentication or biometric access

## AI & Automation

* + Use machine learning for predictive analytics
  + Automate routine tasks or workflows
  + Introduce smart assistants or chatbots

## Analytics & Reporting

* + Expand dashboard capabilities
  + Enable real-time data visualization
  + Add customizable reporting tools