Project Documentation

1. **Introduction**
   - **Project Title**: Store manager: keep track of inventory
   - **Team ID**: NM2025TMID47782
   - **Team Leader**: GOKULNATH S (gokulnaths.831@gmail.com)
   - **Team Members**:
     1. GOBINATH V (velusamygobinath@gmail.com)
     2. HARI M (h898745@gmail.com)
     3. HARISHANKAR S (harishankarsengodan@gmail.com)

   Roles and Responsibilities :

   Team Leader – GOKULNATH S
   Oversees project progress and task allocation.
   Coordinates team communication and final integration.
   Ensures timely submission and quality of deliverables.
   Team Member – GOBINATH V
   Designs and develops UI/UX.
   Implements store dashboard and inventory features.
   Integrates frontend with backend APIs.
   Team Member – HARI M
   Develops server-side logic and database.
   Creates APIs for user, store and inventory.
   Manages authentication and data flow.
   Team Member – HARISHANKAR S
   Conducts testing and debugging.
   Ensures application quality and performance.
   Prepares research inputs and documentation.

2. **Project Overview**

   Purpose :

   - The purpose of the Store Manager Inventory Management System is to provide a digital solution for managing stock and sales in a store. It helps store owners and staff track inventory in real-time, avoid understocking or overstocking, maintain supplier and customer details, and generate reports for better business decisions. This reduces manual errors, saves time, and improves efficiency.

   Features :

   User Management – Admin and staff login with different access levels.
   Product Management – Add, update, delete, and categorize products.
   Stock Tracking – Real-time updates on stock levels aftersales/purchases.
   Low Stock Alerts – Notifications when items reach minimum stock level. Supplier & Customer Records – Maintain details for smooth transactions. Sales & Purchase Management – Store and retrieve transaction history.

Reports Generation – Daily, monthly, and yearly reports for stock and sales. Search & Filter – Quickly find products by name, category, or ID.

3. **Architecture**

   - **Component Structure**:The application follows a modular React component architecture, promoting reusability and separation of concerns. Major components include:

     Header & Navigation – Provides site-wide navigation and access to user profile/settings.

     Inventory List – Displays a grid or list of all products in the store with filtering and search options.

     Product Detail – Shows detailed information about a selected product, including quantity, price, and supplier details.

   - **State Management**: The application uses Context API for global state management. Key state domains include:

     User Authentication & Profile Data

     Inventory Collection & Stock Levels

     Supplier Information

     Sales & Transaction Records

Context Providers wrap the main application tree, allowing child components to access and update shared state without prop drilling. Local component state is used for UI-specific interactions (e.g., modal visibility, form inputs).

   - **Routing**:Routing is the process of defining how the server responds to different HTTP requests (like GET, POST, etc.) at specific URLs.

Using Express.js:

Express is a popular framework built on Node.js that simplifies routing. Example:

Ex :

```
 const express = require('express');

 const app = express();

// Route to display all products in inventory

app.get('/inventory', (req, res) => {

  res.send('Displaying all products in inventory');

});
```

```
// Route to add a new product to inventory
app.post('/inventory/add', (req, res) => {
 res.send('Product added to inventory');
});
// Route to update product details
app.put('/inventory/update/:id', (req, res) => {
  res.send(`Product with ID ${req.params.id} updated`);
});
// Route to delete a product from inventory
app.delete('/inventory/delete/:id', (req, res) => {
  res.send(`Product with ID ${req.params.id} deleted`);
});
app.listen(3000, () => {
  console.log('Server running on port 3000');
};
```

4. **Setup Instructions**
   - **Prerequisites**: Node.js, MongoDB, Git, React.js, Express.js, Visual Studio Code
   - **Installation**:

     ```
     # Clone the repository
     # Install client dependencies → cd client && npm install
     # Install server dependencies → cd ../server && npm install
     ```

5. **Folder Structure**
   - **Client:** The React app is organized into clear folders:

     components/ – Reusable parts of the UI like buttons, headers, and inventory cards

     pages/ – Full screens like Home, Inventory, Orders, Suppliers, and Profile

     assets/ – Images, icons, and other static files

     styles/ – CSS files or styled-components for design

     routes/ – Handles navigation between pages

     context/ – Manages shared data like user info and inventory items

- **Utilities**: Helpful code is stored in:

  utils/ – Functions for filtering inventory, formatting product data, calculating stock levels, etc.

  hooks/ – Custom React hooks like useInventorySync or useAuth

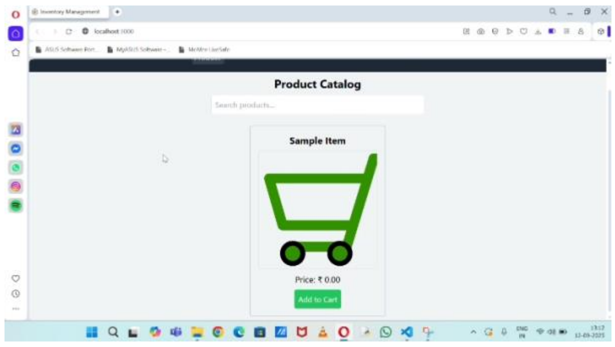
  services/ – API calls like fetching inventory, updating orders, or managing supplier data

6. **Running the Application**
   - Frontend: cd client && npm start
   - Backend: cd server && npm start
   - Access: Visit http://localhost:3000
7. **Component Documentation**
   - **Key Components**:
     - Inventory List–
       - Displays all available inventory items in the store.
       - Props: inventoryItems, on SelectItem
     - Inventory details –
       - Shows full details of a selected inventory item, including stock, supplier, and price.
       - Props:itemId
     - Store manager –
       - Allows users to view and update inventory stock levels.
       - Props: inventoryItems, on UpdateItem
     - Order plan –
       - Helps users manage orders, track deliveries, and generate purchase lists.
       - Props: plannedOrders, on AddOrder
     - User Profile –
       - Manages user settings and store-related preferences.
       - Props: userData, on UpdateProfile
   - **Reusable Components**:
     - Header – Top navigation bar with links and user information.
     - InventoryCard – Compact inventory item preview used in lists.
     - Button – Custom button with styling options.
     - Modal – Popup for forms or messages, such as updating stock or adding orders.
8. **Authentication**
   - JWT-based authentication for secure login. Middleware to protect user and admin routes.
9. **User Interface**

- Home Page – Displays featured inventory items, low-stock alerts, and navigation options.
- Inventory Detail Page – Shows detailed information about a selected inventory item, including quantity, supplier, and price.
- Orders Page – Allows users to view and manage current and past orders.
- Stock Planner – Helps users plan restocking, track incoming shipments, and manage purchase lists.
- Forms – Includes login, inventory addition/updating, order creation, and user profile update forms.

10. **Testing**
    - **Manual Testing** -Manual testing was performed at key development milestones to ensure core features like inventory browsing, stock updates, order management, and supplier tracking worked as expected.
    - Tools Used –
        - Postman – Used to test and verify API endpoints for recipe data, user authentication, and pantry management.
        - Chrome Dev Tools – Used for debugging UI components, inspecting network requests, and monitoring performance.

11. **Screenshots or Demo**



12. **Known Issues**

- Pantry sync delay -Updates to inventory items or stock levels may take a few seconds to reflect across all components.
- Mobile Responsiveness – Some UI elements may not display correctly on smaller screens and may require layout adjustments for proper visibility and usability.

13. **Future Enhancements**

- New Components: Add features like a supplier rating system, order history logs, and stock alerts.
- Enhanced Styling – Improve visual design with animations, transitions, and theme customization options.
- Mobile Optimization – Refine layout and responsiveness for better performance on smartphones and tablets.
- Notifications – Introduce alerts for low-stock items, upcoming orders, and supplier updates.