Introduction:

Project Title:  Store Manager-Keep track of inventory

Team Leader:Fahmiya M

Team Member:Divya M

Team Member:Gokila K

Team Member:Iniya M

Project overview:

Purpose: The purpose of this project is to help store managers efficiently track inventory, monitor stock levels, and manage product details through an easy-to-use web application.

Features:  
  
1. Product Management  
   - Add, edit, and delete product details (name, SKU, price, quantity, supplier).  
   - Bulk upload and bulk edit for faster updates.  
   - Image support for better product identification.  
  
2. Inventory Tracking  
   - Real-time stock level updates.  
   - Batch/lot tracking with expiration date (useful for perishable goods).  
   - Stock adjustment logs for audit trails.  
  
3. Alerts & Notifications  
   - Low-stock and out-of-stock alerts (real-time popups & email integration).  
   - Custom threshold settings for each product.  
   - Expiry date alerts for time-sensitive inventory.  
  
4. Search & Filtering  
   - Full-text search across products, suppliers, and categories.  
   - Advanced filters (price range, availability, category, supplier).  
  
5. Dashboard & Reports  
   - Interactive dashboard with charts (stock trends, sales vs. restock).  
   - Exportable reports (CSV, PDF).  
   - Historical inventory analysis for demand forecasting.  
  
6. User Management & Roles  
   - Role-based access control (Admin, Manager, Staff).  
   - Authentication with JWT/Session tokens.  
   - Audit logs of user activities.  
  
7. Integration & Extensibility  
   - API endpoints for third-party integrations.  
   - Future compatibility with barcode scanners and supplier APIs.  
   - Mobile-first design for tablets and smartphones.

Architecture:

Component Structure:  
- App.js – Root component with routing  
- Navbar – Navigation bar for app pages  
- Dashboard – Overview of inventory stats  
- ProductList – Displays all products in a table format  
- ProductForm – Add/Edit product details  
- InventoryAlerts – Low-stock notifications  
- Reports – Basic inventory reports

State Management:  
- Redux Toolkit for global state (inventory, users, alerts)  
- Local state with useState for component-level interactions

Routing:  
- Implemented with React Router v6  
  - /dashboard – Summary view  
  - /products – Inventory listing  
  - /products/add – Add new product  
  - /products/edit/:id – Edit product

Setup Instructions:

Prerequisites:  
- Node.js (v18+)  
- npm or yarn

Installation:  
git clone <repository-url>  
cd store-manager-frontend  
npm install  
npm start

Folder Structure

client/  
 ├── src/  
 │   ├── components/  
 │   │   ├── Navbar.js  
 │   │   ├── ProductList.js  
 │   │   ├── ProductForm.js  
 │   │   ├── InventoryAlerts.js  
 │   │   └── Dashboard.js  
 │   ├── pages/  
 │   │   ├── ProductsPage.js  
 │   │   ├── DashboardPage.js  
 │   │   └── ReportsPage.js  
 │   ├── redux/  
 │   │   ├── store.js  
 │   │   ├── productSlice.js  
 │   │   └── alertSlice.js  
 │   ├── assets/  
 │   │   ├── styles/  
 │   │   └── images/  
 │   ├── App.js  
 │   └── index.js

Running the Application

npm start

Component Documentation

Key Components:  
- ProductList: Displays products in a sortable, filterable table. Props: products, onDelete, onEdit.  
- ProductForm: Used for adding and editing products. Props: onSubmit, initialValues.  
- InventoryAlerts: Shows notifications for low-stock products. Props: alerts.

Reusable Components:  
- Button (custom styled)  
- Modal (confirmation dialogs)  
- InputField (form input component)

State Management

Global State (Redux):  
- products – stores all product details  
- alerts – stores low-stock warning messages  
- user – authentication and roles

Local State:  
- Form input handling in ProductForm  
- UI states like modal open/close

User Interface

Dashboard: Cards showing total products, low stock count, and recent activity  
Inventory List: Searchable, filterable product table  
Add/Edit Product: Form with validation  
Alerts: Red-highlighted items for low stock

Styling

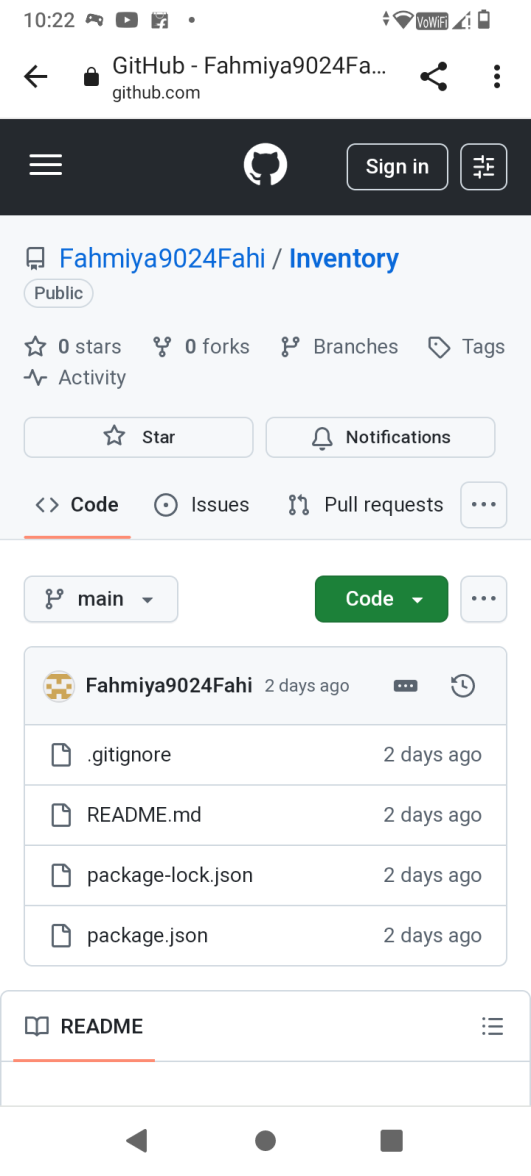
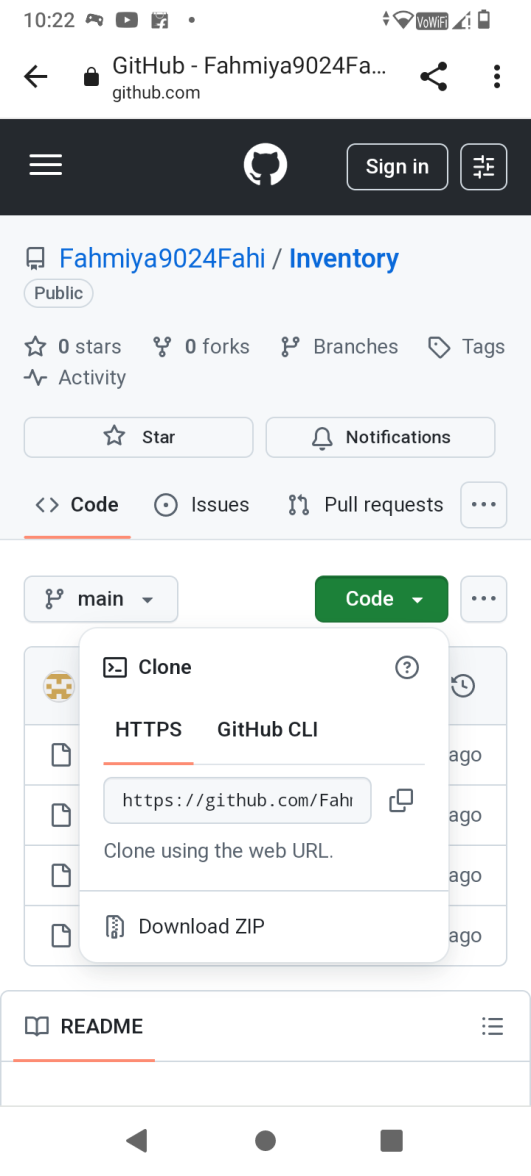
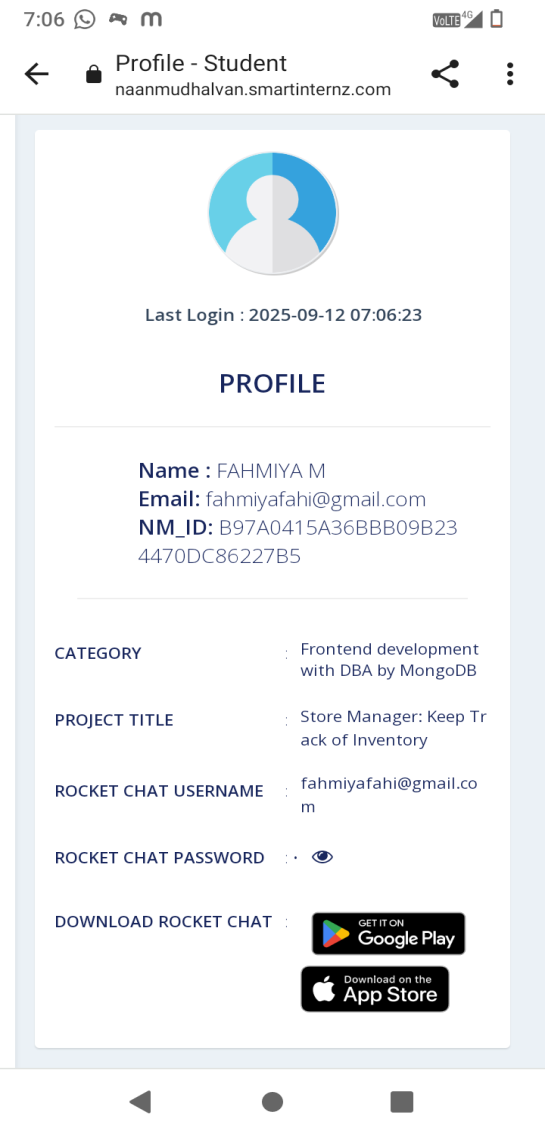
Framework: Tailwind CSS for responsive UI  
Custom Theming: Store branding with color palette

Testing

Strategy:  
- Unit tests with Jest & React Testing Library  
- Integration tests for Redux store  
- E2E tests with Cypress (future enhancement)

Code Coverage:  
- Achieved with Jest coverage reports

Screenshots or Demo:



Future Enhancements

• Barcode scanner support  
• Export reports (CSV/PDF)  
• Multi-store inventory support  
• Dark mode UI  
• Mobile app version for store managers  
• Integration with suppliers (auto re-order)  
• Analytics (top-selling products, seasonal demand)