Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the architecture and core components for **RetailIQ - Smart Retail Analytics Dashboard**. RetailIQ is a React + TypeScript web application providing advanced retail analytics, inventory management, and secure user authentication, with responsive design and state management via Redux Toolkit.

---

1. System Components

Component	Description / Responsibility
UI Layer	React components for dashboard, inventory, auth, etc.
State Management	Redux Toolkit for global state and async actions
Data Visualization	Chart components for analytics (e.g., sales, trends)
Inventory Module	Manage/view inventory data
Auth Module	User login, registration, session management
API Layer	Fetches data from backend REST APIs
Styling	Tailwind CSS for responsive, modern UI

---

2. Class/Interface Overview

Name	Type	Key Methods / Attributes	Relationships
`Dashboard`	Component	Renders analytics, charts	Uses `Chart` , `Inventory`
`Inventory`	Component	List, add, edit, delete inventory items	Uses `InventoryItem`
`AuthForm`	Component	Handles login/register forms	Uses `AuthService`
`RootState`	Interface	Global Redux state shape	Used by Redux
`InventoryItem`	Interface	id, name, qty, price, status	Used in inventory slice
`User`	Interface	id, name, email, role, token	Used in auth slice
`inventorySlice`	Redux Slice	reducers: add, update, remove, fetch	Uses `InventoryItem`
`authSlice`	Redux Slice	reducers: login, logout, register	Uses `User`
`AuthService`	Service	login(), register(), logout()	Used by `authSlice`
`apiClient`	Utility	get(), post(), put(), delete()	Used by slices/services

Key Methods Example:

```
interface InventoryItem {
  id: string;
  name: string;
  quantity: number;
  price: number;
  status: 'in_stock' | 'out_of_stock';
}

function login(email: string, password: string): Promise<User>
```

---

3. Data Structure Overview

Model	Fields
User	id, name, email, role, token
InventoryItem	id, name, quantity, price, status
AnalyticsData	date, sales, revenue, topProducts[], inventoryLevels[]

---

4. Algorithms / Logic

User Authentication Flow:

```
On login/register:
  - Validate input fields
  - Call AuthService.login/register()
  - On success: store user/token in Redux state
  - On failure: show error message
```

Inventory Update Flow:

```
On add/edit/delete inventory:
  - Dispatch Redux action
  - Call API via apiClient
  - On success: update Redux state
  - On error: show notification
```

Data Visualization:

```
On dashboard load:
  - Fetch analytics data from API
  - Transform data for chart components
  - Render charts responsively
```

---

5. Error Handling

Scenario	Handling Approach
API request failure	Show error toast, retry option, log error
Auth failure (invalid creds)	Display inline error, prevent navigation
Form validation error	Highlight fields, show validation messages
Unauthorized access	Redirect to login, clear user state
Data parsing/transform error	Fallback to empty state, log error

---

End of Document