Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the implementation details for **PulseCare - Real-Time Patient Dashboard**. PulseCare is a futuristic, research-oriented healthcare dashboard for hospitals, providing real-time patient data visualization, appointment management, and alerts, with role-based UI for doctors, nurses, and admins. The frontend leverages React, Redux Toolkit, TypeScript, and Tailwind CSS, integrating with REST APIs for live data.

---

1. System Components

Component	Description / Responsibility
App Shell	Layout, routing, authentication, role-based rendering
PatientDashboard	Displays patient vitals, history, and alerts
AppointmentScheduler	Manages and displays appointments
AlertsPanel	Shows real-time alerts and notifications
DynamicForm	Renders and validates dynamic forms (e.g., patient intake)
Redux Store	Centralized state management (patients, appointments, user)
API Service Layer	Handles REST API calls and error handling
RoleBasedAccess	Controls UI access per user role

---

2. Class/Interface Overview

Name	Type	Key Methods / Attributes	Relationships
`Patient`	Interface	id, name, vitals, history, alerts	Used in Redux state, UI
`Appointment`	Interface	id, patientId, doctorId, time, status	Used in Redux state, UI
`User`	Interface	id, name, role	Used for access control
`ApiService`	Class	getPatients(), getAppointments(), postAlert(), …	Used by Redux thunks
`RootState`	Interface	patients, appointments, user, alerts	Redux store root state
`DynamicFormProps`	Interface	schema, onSubmit, initialValues	Used by DynamicForm component

Example: Patient Interface

```
interface Patient {
  id: string;
```

```
  name: string;
  vitals: Vitals;
  history: HistoryEntry[];
  alerts: Alert[];
}
```

---

3. Data Structure Overview

Model	Fields
Patient	id, name, vitals {hr, bp, temp}, history[], alerts[]
Appointment	id, patientId, doctorId, time, status
Alert	id, patientId, type, message, timestamp, severity
User	id, name, role (‘doctor’

---

4. Algorithms / Logic

Role-Based UI Rendering

```
function renderComponentByRole(userRole, componentMap) {
  return componentMap[userRole] || <Unauthorized />;
}
```

Real-Time Alert Handling (Pseudocode)

```
onAlertReceived(alert):
  if alert.patientId in currentView:
    update AlertsPanel with alert
    highlight patient row
  else:
    increment unseenAlertCount
```

Dynamic Form Validation (Pseudocode)

```
for each field in formSchema:
  if field.required and value is empty:
    show error
  if field.type == 'number' and value is not number:
    show error
```

---

5. Error Handling

Scenario	Handling Approach

API failure (network/server)	Show error toast, retry option, log error
Unauthorized access	Redirect to login, show “Access Denied”
Invalid form input	Inline validation errors, prevent submission
Data not found	Show “No Data” state in UI
Real-time connection lost	Show warning banner, attempt reconnection

---

End of Document