High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **PulseCare - Real-Time Patient Monitoring Dashboard**. PulseCare is a futuristic healthcare dashboard enabling real-time patient monitoring with dynamic data visualization, responsive UI, and advanced form handling. The system leverages React, TypeScript, Tailwind CSS, Redux Toolkit, and REST API integration for live updates.

1. System Architecture Overview

Architecture Summary:
PulseCare is a single-page application (SPA) built with a modular frontend architecture. It interacts with external REST APIs for real-time patient data and manages complex state using Redux Toolkit.

Module/Component	Role/Responsibility
UI Layer (React)	Renders dashboard, visualizations, and forms
State Management (Redux)	Manages global app state and synchronizes live data
API Integration Layer	Handles REST API requests/responses for patient data
Data Visualization	Displays real-time charts and metrics
Form Handling	Manages patient data input and validation
Styling (Tailwind CSS)	Provides responsive, futuristic UI design

2. Component Interactions

Interaction Sequence
1. UI Layer dispatches actions (e.g., fetch patient data, submit form)
2. Redux Middleware triggers API Integration Layer to call REST endpoints
3. API responses update Redux state
4. UI Layer subscribes to Redux state, re-renders visualizations and forms with latest data
5. Data Visualization components receive updated data and refresh charts in real time

3. Data Flow Overview

Source	Flow Direction	Destination	Data Type/Content
REST API	→	API Integration	Patient vitals, alerts, metadata

API Integration	→	Redux Store	Normalized patient data
Redux Store	→	UI Components	State slices for display
UI Components	→	Form Handling	User input, validation feedback
Form Handling	→	API Integration	Form submissions (updates)

---

4. Technology Stack

Layer/Function	Technology/Framework
UI Framework	React (with TypeScript)
State Management	Redux Toolkit
Styling/UI	Tailwind CSS, HTML, CSS
Data Visualization	React charting libraries
API Communication	REST (fetch/axios)
Form Handling	React Hook Form / Custom

---

5. Scalability & Reliability

- **Scalability:** Modular React components and Redux slices enable easy extension for new features or patient types. REST API integration supports horizontal scaling.
- **Reliability:** Redux ensures consistent state; error boundaries and API error handling provide resilience. Responsive design ensures usability across devices.
- **Security:** Follows best practices for data privacy, secure API communication (HTTPS), and input validation.

End of Document