Product Requirements & Specification Document

Project Name

PulseCare - Real-Time Patient Dashboard

1. Overview

PulseCare is a futuristic, research-driven healthcare dashboard for hospitals, providing real-time visualization of patient data, appointment schedules, and alerts. The platform features advanced React architecture, Redux Toolkit state management, dynamic forms, responsive design with Tailwind CSS, REST API integration, and role-based UI for doctors, nurses, and admins.

2. Objectives

- Deliver a real-time, unified dashboard for hospital staff.
- Visualize patient data, appointments, and alerts efficiently.
- Ensure secure, role-based access and responsive design.
- Integrate seamlessly with hospital REST APIs.

3. Core Features

Feature	Description	Priority
Patient Data	Real-time display of vitals, history, and demographics.	High
Appointment Schedules	Calendar view, filtering, and management of appointments.	High
Real-Time Alerts	Immediate notifications for critical patient events.	High
Role-Based UI	Custom dashboards and permissions for doctors, nurses, and admins.	High
Dynamic Forms	Configurable forms for patient intake and updates.	Medium
Responsive Design	Optimized for desktop, tablet, and mobile using Tailwind CSS.	High
REST API Integration	Live data sync with hospital backend systems.	High
Audit Logging	Track user actions for compliance and security.	Medium

4. User Roles & Permissions

Role	Access Level
Doctor	View/edit patient data, manage appointments, receive alerts.
Nurse	View patient data, update vitals, receive alerts.

Admin	Full access: manage users, settings, audit logs, and all dashboard features.

5. Technical Specifications

5.1 Architecture

- **Frontend:** React (TypeScript), Redux Toolkit, Tailwind CSS
- **State Management:** Redux Toolkit (slices for patients, appointments, alerts, user)
- **API Integration:** RESTful endpoints (secured with JWT/OAuth)
- **Responsive Design:** Tailwind CSS breakpoints and utility classes
- **Testing:** Jest, React Testing Library

5.2 Component Structure

```
/src
  /components
    /Dashboard
    /PatientCard
    /AppointmentCalendar
    /AlertBanner
    /DynamicForm
    /RoleBasedLayout
  /redux
    /slices
  /api
  /utils
```

5.3 Data Flow

- **Redux Slices:** Patients, Appointments, Alerts, User
- **API Calls:** Thunks for async data fetching and updates
- **WebSocket (optional):** For real-time alerts

5.4 Security

- JWT/OAuth authentication
- Role-based route protection
- Input validation and sanitization

6. UI/UX Requirements

Requirement	Specification
Theme	Futuristic, clean, high-contrast, research-oriented
Navigation	Sidebar with quick access to dashboard sections
Accessibility	WCAG 2.1 AA compliance, keyboard navigation, ARIA labels
Responsiveness	Mobile-first, adaptive layouts for all devices

Customization	User profile settings, theme toggling

7. Integration & APIs

API Endpoint	Method	Description
/api/patients	GET	Fetch patient list
/api/patients/:id	GET	Fetch patient details
/api/appointments	GET	Fetch appointments
/api/alerts	GET	Fetch real-time alerts
/api/forms	GET	Fetch dynamic form definitions
/api/auth	POST	User authentication

8. Non-Functional Requirements

Requirement	Specification
Performance	<2s load time, real-time updates <1s latency
Scalability	Support 1000+ concurrent users
Security	HIPAA-compliant data handling
Maintainability	Modular, well-documented codebase
Localization	Ready for multi-language support

9. Milestones & Deliverables

Milestone	Deliverable	Timeline
UI/UX Design	Wireframes, prototypes	Week 2
Core Dashboard	Patient, appointment, alert modules	Week 4
Role-Based Access	Auth, permissions, protected routes	Week 5
API Integration	Live data sync, error handling	Week 6
Testing & QA	Unit/integration tests, bug fixes	Week 7
Deployment	Production-ready build, documentation	Week 8

10. Acceptance Criteria

- All core features implemented and tested
- Real-time data and alerts function as specified
- Role-based UI and permissions enforced

- Responsive and accessible on all devices
- Security and compliance requirements met

---

11. Appendix

Sample Redux Slice (Pseudocode)

```
// patientsSlice.ts
createSlice({
  name: 'patients',
  initialState: [],
  reducers: {
    setPatients(state, action) { ... },
    updatePatient(state, action) { ... }
  },
  extraReducers: (builder) => {
    builder.addCase(fetchPatients.fulfilled, (state, action) => { ... });
  }
});
```

End of Document