COMMUNITY BASED INTELLIGENCE

Frontend Implementation Guide

Health Officer Dashboard

Next.js 14 + TypeScript + Tailwind CSS + shadcn/ui

Version 1.0 - January 2026

1. Overview & Technology Stack

1.1 Dashboard Purpose

The Health Officer Dashboard provides a real-time interface for monitoring, investigating, and responding to health incidents reported through the CBI system. It enables health officers to view reports, track outbreaks, and coordinate response efforts.

1.2 Key Features

- Real-time notifications for critical health alerts with sound
- Interactive map showing geographic distribution of cases
- Report management with filtering, search, and status updates
- Analytics dashboard with trend charts and disease distribution
- Conversation history viewer for each report
- Mobile-responsive design for field access

1.3 Technology Stack

Technology	Version	Purpose
Next.js	14.1+	React framework with App Router
TypeScript	5.3+	Type-safe JavaScript
Tailwind CSS	3.4+	Utility-first CSS
shadcn/ui	latest	Accessible components
React Query	5.17+	Server state management
Zustand	4.5+	Client state management
Socket.io	4.7+	Real-time WebSocket
Recharts	2.10+	Data visualization
React Leaflet	4.2+	Interactive maps
Lucide React	0.309+	Icons

2. Project Setup

2.1 Create Next.js Project

```bash
# Create new Next.js project
npx create-next-app@latest dashboard --typescript --tailwind --eslint --app --src-dir

cd dashboard

# Install dependencies
npm install @tanstack/react-query zustand socket.io-client
npm install recharts react-leaflet leaflet lucide-react
npm install date-fns react-hook-form @hookform/resolvers zod
npm install -D @types/leaflet
```

2.2 Install shadcn/ui Components

```bash
# Initialize shadcn/ui
npx shadcn-ui@latest init

# Install components
npx shadcn-ui@latest add button card input label badge
npx shadcn-ui@latest add dropdown-menu dialog sheet tabs
npx shadcn-ui@latest add select table avatar separator
npx shadcn-ui@latest add alert toast sonner skeleton form
```

3. Project Structure

```
Directory
dashboard/
├── src/
│   ├── app/
│   │   ├── (auth)/
│   │   │   ├── login/page.tsx
│   │   │   └── layout.tsx
│   │   ├── (dashboard)/
│   │   │   ├── layout.tsx          # Protected layout
│   │   │   ├── page.tsx            # Dashboard home
│   │   │   ├── reports/
│   │   │   │   ├── page.tsx        # Reports list
│   │   │   │   └── [id]/page.tsx   # Report detail
│   │   │   ├── map/page.tsx        # Full map view
│   │   │   ├── analytics/page.tsx  # Analytics
│   │   │   └── settings/page.tsx   # Settings
│   │   ├── layout.tsx              # Root layout
│   │   ├── globals.css
│   │   └── providers.tsx           # React Query provider
│   ├── components/
│   │   ├── ui/                     # shadcn components
│   │   ├── layout/
│   │   │   ├── Sidebar.tsx
│   │   │   ├── Header.tsx
│   │   │   └── MobileNav.tsx
│   │   ├── dashboard/
│   │   │   ├── StatsCard.tsx
│   │   │   └── RecentAlerts.tsx
│   │   ├── reports/
│   │   │   ├── ReportTable.tsx
│   │   │   ├── ReportFilters.tsx
│   │   │   ├── ReportDetail.tsx
│   │   │   └── ConversationView.tsx
│   │   ├── map/
│   │   │   ├── IncidentMap.tsx
│   │   │   └── MapMarker.tsx
│   │   ├── charts/
│   │   │   ├── CasesTrend.tsx
│   │   │   └── DiseaseDistribution.tsx
│   │   └── notifications/
│   │       ├── NotificationBell.tsx
│   │       └── NotificationList.tsx
│   ├── hooks/
│   │   ├── useReports.ts
│   │   ├── useAnalytics.ts
│   │   └── useRealtime.ts
│   ├── lib/
│   │   ├── api.ts                  # API client
│   │   └── utils.ts
│   ├── stores/
│   │   ├── authStore.ts
│   │   └── notificationStore.ts
│   └── types/
│       └── index.ts
├── public/
│   └── sounds/alert.mp3
├── .env.local
└── package.json
```

4. Authentication System

4.1 Auth Store (Zustand)

TypeScript
```typescript
// src/stores/authStore.ts

import { create } from 'zustand';
import { persist } from 'zustand/middleware';

interface Officer {
  id: string;
  email: string;
  fullName: string;
  assignedRegions: string[];
}

interface AuthState {
  officer: Officer | null;
  accessToken: string | null;
  isAuthenticated: boolean;
  login: (email: string, password: string) => Promise<void>;
  logout: () => void;
}

export const useAuthStore = create<AuthState>()(
  persist(
    (set) => ({
      officer: null,
      accessToken: null,
      isAuthenticated: false,

      login: async (email, password) => {
        const res = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/v1/auth/login`, {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ email, password }),
        });
        if (!res.ok) throw new Error('Invalid credentials');
        const data = await res.json();
        set({
          officer: data.officer,
          accessToken: data.accessToken,
          isAuthenticated: true,
        });
      },

      logout: () => set({
        officer: null,
        accessToken: null,
        isAuthenticated: false,
      }),
    }),
    { name: 'cbi-auth' }
  )
);
```

4.2 Login Page

```typescript
// src/app/(auth)/login/page.tsx

'use client';

import { useState } from 'react';
import { useRouter } from 'next/navigation';
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Label } from '@/components/ui/label';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Alert, AlertDescription } from '@/components/ui/alert';
import { useAuthStore } from '@/stores/authStore';
import { AlertCircle, Loader2 } from 'lucide-react';

export default function LoginPage() {
  const router = useRouter();
  const { login } = useAuthStore();
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setError('');
    setLoading(true);
    try {
      await login(email, password);
      router.push('/');
    } catch {
      setError('Invalid email or password');
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="min-h-screen flex items-center justify-center bg-slate-50 p-4">
      <Card className="w-full max-w-md">
        <CardHeader className="text-center">
          <div className="mx-auto mb-4 h-12 w-12 rounded-full bg-primary/10 flex items-center
justify-center">
            <span className="text-2xl">🏥</span>
          </div>
          <CardTitle>CBI Dashboard</CardTitle>
        </CardHeader>
        <CardContent>
          <form onSubmit={handleSubmit} className="space-y-4">
            {error && (
              <Alert variant="destructive">
                <AlertCircle className="h-4 w-4" />
                <AlertDescription>{error}</AlertDescription>
              </Alert>
            )}
            <div className="space-y-2">
              <Label htmlFor="email">Email</Label>
              <Input
                id="email"
                type="email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                required
              />
            </div>
            <div className="space-y-2">
              <Label htmlFor="password">Password</Label>
              <Input
                id="password"
                type="password"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
                required
              />
            </div>
            <Button type="submit" className="w-full" disabled={loading}>
              {loading ? <Loader2 className="h-4 w-4 animate-spin" /> : 'Sign In'}
            </Button>
          </form>
        </CardContent>
      </Card>
    </div>
  );
}
```

5. Layout & Navigation

5.1 Sidebar Component

TypeScript

```
// src/components/layout/Sidebar.tsx

'use client';

import Link from 'next/link';
import { usePathname } from 'next/navigation';
import { LayoutDashboard, FileText, Map, BarChart3, Settings, LogOut, AlertTriangle } from
'lucide-react';
import { cn } from '@/lib/utils';
import { useAuthStore } from '@/stores/authStore';
import { Button } from '@/components/ui/button';

const navigation = [
  { name: 'Dashboard', href: '/', icon: LayoutDashboard },
  { name: 'Reports', href: '/reports', icon: FileText },
  { name: 'Map', href: '/map', icon: Map },
  { name: 'Analytics', href: '/analytics', icon: BarChart3 },
  { name: 'Settings', href: '/settings', icon: Settings },
];

export function Sidebar() {
  const pathname = usePathname();
  const { officer, logout } = useAuthStore();

  return (
    <aside className="hidden lg:fixed lg:inset-y-0 lg:flex lg:w-64 lg:flex-col">
      <div className="flex flex-1 flex-col bg-slate-900 px-4 py-6">
        {/* Logo */}
        <div className="flex items-center gap-3 px-2 mb-8">
          <div className="h-10 w-10 rounded-lg bg-primary flex items-center justify-center">
            <AlertTriangle className="h-6 w-6 text-white" />
          </div>
          <div>
            <h1 className="text-lg font-bold text-white">CBI</h1>
            <p className="text-xs text-slate-400">Health Surveillance</p>
          </div>
        </div>

        {/* Navigation */}
        <nav className="flex-1 space-y-1">
          {navigation.map((item) => {
            const isActive = pathname === item.href ||
              (item.href !== '/' && pathname.startsWith(item.href));
            return (
              <Link
                key={item.name}
                href={item.href}
                className={cn(
                  'flex items-center gap-3 px-3 py-2 rounded-lg text-sm font-medium
transition-colors',
                  isActive
                    ? 'bg-primary text-white'
                    : 'text-slate-300 hover:bg-slate-800 hover:text-white'
                )}
              >
                <item.icon className="h-5 w-5" />
                {item.name}
              </Link>
            );
          })}
        </nav>

        {/* User */}
        <div className="border-t border-slate-700 pt-4">
          <div className="flex items-center gap-3 px-2 mb-3">
            <div className="h-9 w-9 rounded-full bg-primary/20 flex items-center justify-
center">
              <span className="text-sm font-medium text-primary">
                {officer?.fullName?.charAt(0)}
              </span>
            </div>
            <div className="flex-1 min-w-0">
              <p className="text-sm font-medium text-white truncate">{officer?.fullName}</p>
              <p className="text-xs text-slate-400 truncate">{officer?.email}</p>
            </div>
          </div>
          <Button variant="ghost" className="w-full justify-start text-slate-300"
onClick={logout}>
            <LogOut className="h-5 w-5 mr-3" />
            Sign out
          </Button>
        </div>
      </div>
    </aside>
  );
}
```

5.2 Header with Notifications

```TypeScript
// src/components/layout/Header.tsx

'use client';

import { useState } from 'react';
import { Menu, Search } from 'lucide-react';
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { NotificationBell } from '@/components/notifications/NotificationBell';
import { MobileNav } from './MobileNav';

export function Header() {
  const [mobileNavOpen, setMobileNavOpen] = useState(false);
  const [search, setSearch] = useState('');

  return (
    <>
      <header className="sticky top-0 z-40 bg-white border-b border-slate-200">
        <div className="flex items-center justify-between h-16 px-4 sm:px-6">
          <Button
            variant="ghost"
            size="icon"
            className="lg:hidden"
            onClick={() => setMobileNavOpen(true)}
          >
            <Menu className="h-6 w-6" />
          </Button>

          <div className="flex-1 max-w-md mx-4">
            <div className="relative">
              <Search className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-
400" />
              <Input
                type="search"
                placeholder="Search reports..."
                className="pl-10"
                value={search}
                onChange={(e) => setSearch(e.target.value)}
              />
            </div>
          </div>

          <NotificationBell />
        </div>
      </header>
      <MobileNav open={mobileNavOpen} onClose={() => setMobileNavOpen(false)} />
    </>
  );
}
```

5.3 Protected Dashboard Layout

```typescript
// src/app/(dashboard)/layout.tsx

'use client';

import { useEffect } from 'react';
import { useRouter } from 'next/navigation';
import { useAuthStore } from '@/stores/authStore';
import { Sidebar } from '@/components/layout/Sidebar';
import { Header } from '@/components/layout/Header';
import { useRealtime } from '@/hooks/useRealtime';

export default function DashboardLayout({ children }: { children: React.ReactNode }) {
  const router = useRouter();
  const { isAuthenticated } = useAuthStore();

  // Initialize real-time WebSocket connection
  useRealtime();

  useEffect(() => {
    if (!isAuthenticated) {
      router.push('/login');
    }
  }, [isAuthenticated, router]);

  if (!isAuthenticated) {
    return (
      <div className="min-h-screen flex items-center justify-center">
        <div className="animate-spin h-8 w-8 border-4 border-primary border-t-transparent
rounded-full" />
      </div>
    );
  }

  return (
    <div className="min-h-screen bg-slate-50">
      <Sidebar />
      <div className="lg:pl-64">
        <Header />
        <main className="p-6">{children}</main>
      </div>
    </div>
  );
}
```

6. Dashboard Overview Page

TypeScript
```
// src/app/(dashboard)/page.tsx

'use client';

import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { AlertTriangle, Activity, MapPin, Users } from 'lucide-react';
import { StatsCard } from '@/components/dashboard/StatsCard';
import { RecentAlerts } from '@/components/dashboard/RecentAlerts';
import { CasesTrend } from '@/components/charts/CasesTrend';
import { DiseaseDistribution } from '@/components/charts/DiseaseDistribution';
import { useAnalytics } from '@/hooks/useAnalytics';

export default function DashboardPage() {
  const { data: analytics, isLoading } = useAnalytics();

  return (
    <div className="space-y-6">
      <div>
        <h1 className="text-2xl font-bold text-slate-900">Dashboard</h1>
        <p className="text-slate-500">Overview of health incident reports</p>
      </div>

      {/* Stats Cards */}
      <div className="grid gap-4 md:grid-cols-2 lg:grid-cols-4">
        <StatsCard
          title="Critical Alerts"
          value={analytics?.criticalAlerts ?? 0}
          icon={<AlertTriangle className="h-5 w-5" />}
          iconColor="text-red-500"
          iconBg="bg-red-50"
          trend={analytics?.criticalTrend}
        />
        <StatsCard
          title="Active Cases"
          value={analytics?.activeCases ?? 0}
          icon={<Activity className="h-5 w-5" />}
          iconColor="text-amber-500"
          iconBg="bg-amber-50"
          trend={analytics?.casesTrend}
        />
        <StatsCard
          title="Affected Regions"
          value={analytics?.affectedRegions ?? 0}
          icon={<MapPin className="h-5 w-5" />}
          iconColor="text-blue-500"
          iconBg="bg-blue-50"
        />
        <StatsCard
          title="Reports Today"
          value={analytics?.reportsToday ?? 0}
          icon={<Users className="h-5 w-5" />}
          iconColor="text-green-500"
          iconBg="bg-green-50"
        />
      </div>
      {/* Charts */}
      <div className="grid gap-6 lg:grid-cols-2">
        <Card>
          <CardHeader>
            <CardTitle className="text-lg">Cases Trend (7 Days)</CardTitle>
          </CardHeader>
          <CardContent>
            <CasesTrend data={analytics?.trendData ?? []} />
          </CardContent>
        </Card>
        <Card>
          <CardHeader>
            <CardTitle className="text-lg">Disease Distribution</CardTitle>
          </CardHeader>
          <CardContent>
            <DiseaseDistribution data={analytics?.diseaseData ?? []} />
          </CardContent>
        </Card>
      </div>

      {/* Recent Alerts */}
      <Card>
        <CardHeader>
          <CardTitle className="text-lg">Recent Alerts</CardTitle>
        </CardHeader>
        <CardContent>
          <RecentAlerts />
        </CardContent>
      </Card>
    </div>
  );
}
```

6.1 Stats Card Component

```TypeScript
// src/components/dashboard/StatsCard.tsx

import { Card, CardContent } from '@/components/ui/card';
import { TrendingUp, TrendingDown } from 'lucide-react';
import { cn } from '@/lib/utils';

interface StatsCardProps {
  title: string;
  value: number;
  icon: React.ReactNode;
  iconColor: string;
  iconBg: string;
  trend?: number;
}

export function StatsCard({ title, value, icon, iconColor, iconBg, trend }: StatsCardProps) {
  return (
    <Card>
      <CardContent className="p-6">
        <div className="flex items-start justify-between">
          <div>
            <p className="text-sm font-medium text-slate-500">{title}</p>
            <p className="text-3xl font-bold text-slate-900 mt-1">{value.toLocaleString()}</p>
            {trend !== undefined && (
              <div className="flex items-center mt-2 text-sm">
                {trend > 0 ? (
                  <TrendingUp className="h-4 w-4 text-red-500 mr-1" />
                ) : trend < 0 ? (
                  <TrendingDown className="h-4 w-4 text-green-500 mr-1" />
                ) : null}
                <span className={cn(
                  'font-medium',
                  trend > 0 && 'text-red-500',
                  trend < 0 && 'text-green-500'
                )}>
                  {trend > 0 && '+'}{trend}%
                </span>
                <span className="text-slate-400 ml-1">vs last week</span>
              </div>
            )}
          </div>
          <div className={cn('p-3 rounded-lg', iconBg)}>
            <div className={iconColor}>{icon}</div>
          </div>
        </div>
      </CardContent>
    </Card>
  );
}
```

7. Reports Management

7.1 Reports List Page

```typescript
// src/app/(dashboard)/reports/page.tsx

'use client';

import { useState } from 'react';
import { ReportFilters } from '@/components/reports/ReportFilters';
import { ReportTable } from '@/components/reports/ReportTable';
import { useReports } from '@/hooks/useReports';
import { Button } from '@/components/ui/button';
import { Download } from 'lucide-react';

export default function ReportsPage() {
  const [filters, setFilters] = useState({
    status: '',
    urgency: '',
    disease: '',
    search: '',
  });
  const [page, setPage] = useState(1);

  const { data, isLoading } = useReports({ ...filters, page, limit: 20 });

  return (
    <div className="space-y-6">
      <div className="flex items-center justify-between">
        <div>
          <h1 className="text-2xl font-bold text-slate-900">Reports</h1>
          <p className="text-slate-500">{data?.total ?? 0} total reports</p>
        </div>
        <Button variant="outline">
          <Download className="h-4 w-4 mr-2" />
          Export
        </Button>
      </div>

      <ReportFilters filters={filters} onFiltersChange={setFilters} />
      <ReportTable reports={data?.reports ?? []} isLoading={isLoading} />

      {data && data.total > 20 && (
        <div className="flex items-center justify-between">
          <p className="text-sm text-slate-500">
            Showing {((page - 1) * 20) + 1} to {Math.min(page * 20, data.total)} of
{data.total}
          </p>
          <div className="flex gap-2">
            <Button variant="outline" size="sm" disabled={page === 1} onClick={() => setPage(p
=> p - 1)}>
              Previous
            </Button>
            <Button variant="outline" size="sm" disabled={page * 20 >= data.total} onClick={()
=> setPage(p => p + 1)}>
              Next
            </Button>
          </div>
        </div>
      )}
    </div>
  );
}
```

7.2 Report Filters

```typescript
// src/components/reports/ReportFilters.tsx

'use client';

import { Search, X, Filter } from 'lucide-react';
import { Input } from '@/components/ui/input';
import { Button } from '@/components/ui/button';
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
'@/components/ui/select';
import { Badge } from '@/components/ui/badge';

interface Filters {
  status: string;
  urgency: string;
  disease: string;
  search: string;
}

interface Props {
  filters: Filters;
  onFiltersChange: (filters: Filters) => void;
}

export function ReportFilters({ filters, onFiltersChange }: Props) {
  const update = (key: keyof Filters, value: string) => {
    onFiltersChange({ ...filters, [key]: value });
  };

  const clear = () => {
    onFiltersChange({ status: '', urgency: '', disease: '', search: '' });
  };

  const activeCount = Object.values(filters).filter(Boolean).length;

  return (
    <div className="bg-white rounded-lg border border-slate-200 p-4 space-y-4">
      <div className="relative">
        <Search className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-400" />
        <Input
          placeholder="Search by location, symptoms..."
          className="pl-10"
          value={filters.search}
          onChange={(e) => update('search', e.target.value)}
        />
      </div>

      <div className="grid grid-cols-2 md:grid-cols-3 gap-3">
        <Select value={filters.status} onValueChange={(v) => update('status', v)}>
          <SelectTrigger><SelectValue placeholder="Status" /></SelectTrigger>
          <SelectContent>
            <SelectItem value="open">Open</SelectItem>
            <SelectItem value="investigating">Investigating</SelectItem>
            <SelectItem value="resolved">Resolved</SelectItem>
            <SelectItem value="false_alarm">False Alarm</SelectItem>
          </SelectContent>
        </Select>

        <Select value={filters.urgency} onValueChange={(v) => update('urgency', v)}>
          <SelectTrigger><SelectValue placeholder="Urgency" /></SelectTrigger>
          <SelectContent>
            <SelectItem value="critical">Critical</SelectItem>
            <SelectItem value="high">High</SelectItem>
            <SelectItem value="medium">Medium</SelectItem>
          </SelectContent>
        </Select>

        <Select value={filters.disease} onValueChange={(v) => update('disease', v)}>
          <SelectTrigger><SelectValue placeholder="Disease" /></SelectTrigger>
          <SelectContent>
            <SelectItem value="cholera">Cholera</SelectItem>
            <SelectItem value="dengue">Dengue</SelectItem>
            <SelectItem value="malaria">Malaria</SelectItem>
          </SelectContent>
        </Select>
      </div>

      {activeCount > 0 && (
        <div className="flex items-center gap-2">
          <Badge variant="secondary">
            <Filter className="h-3 w-3 mr-1" />
            {activeCount} filter{activeCount !== 1 && 's'}
          </Badge>
          <Button variant="ghost" size="sm" onClick={clear}>
            <X className="h-4 w-4 mr-1" /> Clear
          </Button>
        </div>
      )}
    </div>
  );
}
```

7.3 Reports Table

```typescript
// src/components/reports/ReportTable.tsx

'use client';

import Link from 'next/link';
import { formatDistanceToNow } from 'date-fns';
import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from
'@/components/ui/table';
import { Badge } from '@/components/ui/badge';
import { Skeleton } from '@/components/ui/skeleton';
import type { Report } from '@/types';
import { cn } from '@/lib/utils';

const urgencyColors = {
  critical: 'bg-red-100 text-red-800',
  high: 'bg-amber-100 text-amber-800',
  medium: 'bg-blue-100 text-blue-800',
  low: 'bg-slate-100 text-slate-800',
};

const statusColors = {
  open: 'bg-yellow-100 text-yellow-800',
  investigating: 'bg-blue-100 text-blue-800',
  resolved: 'bg-green-100 text-green-800',
  false_alarm: 'bg-slate-100 text-slate-800',
};

export function ReportTable({ reports, isLoading }: { reports: Report[]; isLoading: boolean })
{
  if (isLoading) {
    return (
      <div className="bg-white rounded-lg border">
        <Table>
          <TableHeader>
            <TableRow>
              <TableHead>Disease</TableHead>
              <TableHead>Location</TableHead>
              <TableHead>Cases</TableHead>
              <TableHead>Urgency</TableHead>
              <TableHead>Status</TableHead>
              <TableHead>Reported</TableHead>
            </TableRow>
          </TableHeader>
          <TableBody>
            {[...Array(5)].map((_, i) => (
              <TableRow key={i}>
                {[...Array(6)].map((_, j) => (
                  <TableCell key={j}><Skeleton className="h-5 w-full" /></TableCell>
                ))}
              </TableRow>
            ))}
          </TableBody>
        </Table>
      </div>
    );
  }
  return (
    <div className="bg-white rounded-lg border">
      <Table>
        <TableHeader>
          <TableRow>
            <TableHead>Disease</TableHead>
            <TableHead>Location</TableHead>
            <TableHead className="text-center">Cases</TableHead>
            <TableHead>Urgency</TableHead>
            <TableHead>Status</TableHead>
            <TableHead>Reported</TableHead>
          </TableRow>
        </TableHeader>
        <TableBody>
          {reports.map((report) => (
            <TableRow key={report.id} className="cursor-pointer hover:bg-slate-50">
              <TableCell>
                <Link href={`/reports/${report.id}`} className="font-medium capitalize">
                  {report.suspectedDisease || 'Unknown'}
                </Link>
              </TableCell>
              <TableCell className="text-slate-600">
                {report.locationNormalized || report.locationText || 'Unknown'}
              </TableCell>
              <TableCell className="text-center font-medium">{report.casesCount ?? '-
'}</TableCell>
              <TableCell>
                <Badge className={cn('capitalize', urgencyColors[report.urgency])}>
                  {report.urgency}
                </Badge>
              </TableCell>
              <TableCell>
                <Badge className={cn('capitalize', statusColors[report.status])}>
                  {report.status.replace('_', ' ')}
                </Badge>
              </TableCell>
              <TableCell className="text-slate-500">
                {formatDistanceToNow(new Date(report.createdAt), { addSuffix: true })}
              </TableCell>
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </div>
  );
}
```

8. Notification System

8.1 Notification Store

TypeScript
```typescript
// src/stores/notificationStore.ts

import { create } from 'zustand';

export interface Notification {
  id: string;
  title: string;
  body: string;
  urgency: 'critical' | 'high' | 'medium' | 'low';
  reportId?: string;
  timestamp: Date;
  read: boolean;
}

interface NotificationState {
  notifications: Notification[];
  unreadCount: number;
  addNotification: (n: Omit<Notification, 'read'>) => void;
  markAllRead: () => void;
}

export const useNotificationStore = create<NotificationState>((set) => ({
  notifications: [],
  unreadCount: 0,

  addNotification: (notification) => {
    set((state) => ({
      notifications: [{ ...notification, read: false }, ...state.notifications].slice(0, 50),
      unreadCount: state.unreadCount + 1,
    }));

    // Play sound for critical alerts
    if (notification.urgency === 'critical') {
      const audio = new Audio('/sounds/alert.mp3');
      audio.play().catch(() => {});
    }
  },

  markAllRead: () => {
    set((state) => ({
      notifications: state.notifications.map((n) => ({ ...n, read: true })),
      unreadCount: 0,
    }));
  },
}));
```

8.2 Notification Bell

TypeScript
```typescript
// src/components/notifications/NotificationBell.tsx

```

```tsx
'use client';

import { useState } from 'react';
import { Bell } from 'lucide-react';
import { Button } from '@/components/ui/button';
import { Popover, PopoverContent, PopoverTrigger } from '@/components/ui/popover';
import { useNotificationStore } from '@/stores/notificationStore';
import { NotificationList } from './NotificationList';
import { cn } from '@/lib/utils';

export function NotificationBell() {
  const [open, setOpen] = useState(false);
  const { notifications, unreadCount, markAllRead } = useNotificationStore();

  const handleOpen = (isOpen: boolean) => {
    setOpen(isOpen);
    if (isOpen && unreadCount > 0) {
      setTimeout(() => markAllRead(), 1000);
    }
  };

  return (
    <Popover open={open} onOpenChange={handleOpen}>
      <PopoverTrigger asChild>
        <Button variant="ghost" size="icon" className="relative">
          <Bell className="h-5 w-5" />
          {unreadCount > 0 && (
            <span className={cn(
              'absolute -top-1 -right-1 h-5 w-5 rounded-full',
              'bg-red-500 text-white text-xs font-bold',
              'flex items-center justify-center animate-pulse'
            )}>
              {unreadCount > 9 ? '9+' : unreadCount}
            </span>
          )}
        </Button>
      </PopoverTrigger>
      <PopoverContent align="end" className="w-80 p-0">
        <div className="p-4 border-b">
          <h3 className="font-semibold">Notifications</h3>
          <p className="text-sm text-slate-500">
            {unreadCount > 0 ? `${unreadCount} unread` : 'All caught up!'}
          </p>
        </div>
        <NotificationList notifications={notifications} onClose={() => setOpen(false)} />
      </PopoverContent>
    </Popover>
  );
}
```

9. Real-time Updates (WebSocket)

TypeScript

```
// src/hooks/useRealtime.ts

'use client';

import { useEffect, useRef } from 'react';
import { io, Socket } from 'socket.io-client';
import { useQueryClient } from '@tanstack/react-query';
import { useNotificationStore } from '@/stores/notificationStore';
import { useAuthStore } from '@/stores/authStore';

export function useRealtime() {
  const socketRef = useRef<Socket | null>(null);
  const queryClient = useQueryClient();
  const { addNotification } = useNotificationStore();
  const { accessToken, isAuthenticated } = useAuthStore();

  useEffect(() => {
    if (!isAuthenticated || !accessToken) return;

    // Connect to WebSocket server
    socketRef.current = io(process.env.NEXT_PUBLIC_WS_URL!, {
      auth: { token: accessToken },
      transports: ['websocket'],
    });

    const socket = socketRef.current;

    socket.on('connect', () => console.log('WebSocket connected'));

    // Handle new alerts
    socket.on('new_alert', (data) => {
      addNotification({
        id: data.id,
        title: data.title,
        body: data.body,
        urgency: data.urgency,
        reportId: data.reportId,
        timestamp: new Date(),
      });

      // Refresh data
      queryClient.invalidateQueries({ queryKey: ['reports'] });
      queryClient.invalidateQueries({ queryKey: ['analytics'] });
    });

    // Handle report updates
    socket.on('report_updated', (data) => {
      queryClient.invalidateQueries({ queryKey: ['reports', data.reportId] });
      queryClient.invalidateQueries({ queryKey: ['reports'] });
    });

    return () => { socket.disconnect(); };
  }, [isAuthenticated, accessToken, addNotification, queryClient]);

  return socketRef.current;
}
```

10. API Integration

10.1 API Client

```typescript
// src/lib/api.ts

import { useAuthStore } from '@/stores/authStore';

const API_URL = process.env.NEXT_PUBLIC_API_URL;

class APIClient {
  private getHeaders(): HeadersInit {
    const token = useAuthStore.getState().accessToken;
    return {
      'Content-Type': 'application/json',
      ...(token && { Authorization: `Bearer ${token}` }),
    };
  }

  async get<T>(path: string, params?: Record<string, any>): Promise<T> {
    const url = new URL(`${API_URL}${path}`);
    if (params) {
      Object.entries(params).forEach(([k, v]) => {
        if (v !== undefined && v !== '') url.searchParams.set(k, String(v));
      });
    }
    const res = await fetch(url.toString(), { headers: this.getHeaders() });
    if (!res.ok) throw new Error('Request failed');
    return res.json();
  }

  async post<T>(path: string, data?: any): Promise<T> {
    const res = await fetch(`${API_URL}${path}`, {
      method: 'POST',
      headers: this.getHeaders(),
      body: JSON.stringify(data),
    });
    if (!res.ok) throw new Error('Request failed');
    return res.json();
  }

  async patch<T>(path: string, data?: any): Promise<T> {
    const res = await fetch(`${API_URL}${path}`, {
      method: 'PATCH',
      headers: this.getHeaders(),
      body: JSON.stringify(data),
    });
    if (!res.ok) throw new Error('Request failed');
    return res.json();
  }
}

export const api = new APIClient();
```

10.2 React Query Hooks

TypeScript

```typescript
// src/hooks/useReports.ts

import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import { api } from '@/lib/api';
import type { Report, ReportListResponse } from '@/types';

interface Filters {
  status?: string;
  urgency?: string;
  disease?: string;
  search?: string;
  page?: number;
  limit?: number;
}

export function useReports(filters: Filters = {}) {
  return useQuery({
    queryKey: ['reports', filters],
    queryFn: () => api.get<ReportListResponse>('/api/v1/reports', filters),
    staleTime: 30000,
  });
}

export function useReport(id: string) {
  return useQuery({
    queryKey: ['reports', id],
    queryFn: () => api.get<Report>(`/api/v1/reports/${id}`),
    enabled: !!id,
  });
}

export function useUpdateReport() {
  const queryClient = useQueryClient();
  return useMutation({
    mutationFn: ({ id, ...data }: { id: string } & Partial<Report>) =>
      api.patch<Report>(`/api/v1/reports/${id}`, data),
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ['reports'] });
    },
  });
}

// src/hooks/useAnalytics.ts

export function useAnalytics() {
  return useQuery({
    queryKey: ['analytics'],
    queryFn: () => api.get('/api/v1/analytics/summary'),
    staleTime: 60000,
  });
}
```

11. Type Definitions

```
TypeScript
// src/types/index.ts

export interface Report {
  id: string;
  conversationId: string;
  platform: 'telegram' | 'whatsapp';
  status: 'open' | 'investigating' | 'resolved' | 'false_alarm';
  createdAt: string;
  updatedAt: string;

  // MVS Data
  symptoms: string[];
  suspectedDisease: 'cholera' | 'dengue' | 'malaria' | 'unknown' | null;
  locationText: string | null;
  locationNormalized: string | null;
  locationCoords: { lat: number; lng: number } | null;
  onsetText: string | null;
  onsetDate: string | null;
  casesCount: number | null;
  deathsCount: number | null;

  // Classification
  dataCompleteness: number;
  urgency: 'critical' | 'high' | 'medium' | 'low';
  alertType: string | null;
  thresholdExceeded: boolean;

  // Investigation
  assignedOfficerId: string | null;
  investigationNotes: string | null;
  outcome: string | null;

  // Conversation
  rawConversation: {
    messages: Array<{
      role: 'user' | 'assistant';
      content: string;
      timestamp: string;
    }>;
  } | null;
}

export interface ReportListResponse {
  reports: Report[];
  total: number;
  limit: number;
  offset: number;
}

export interface AnalyticsSummary {
  criticalAlerts: number;
  criticalTrend: number;
  activeCases: number;
  casesTrend: number;
  affectedRegions: number;
  reportsToday: number;
  trendData: Array<{ date: string; cholera: number; dengue: number; malaria: number }>;
  diseaseData: Array<{ name: string; value: number }>;
}
```

12. Environment Configuration

```Environment
# .env.local

# API Configuration
NEXT_PUBLIC_API_URL=http://localhost:8000
NEXT_PUBLIC_WS_URL=http://localhost:8000
```

12.1 Provider Setup

```TypeScript
// src/app/providers.tsx

'use client';

import { QueryClient, QueryClientProvider } from '@tanstack/react-query';
import { useState } from 'react';
import { Toaster } from '@/components/ui/sonner';

export function Providers({ children }: { children: React.ReactNode }) {
  const [queryClient] = useState(() => new QueryClient({
    defaultOptions: {
      queries: { staleTime: 30000, retry: 1 },
    },
  }));

  return (
    <QueryClientProvider client={queryClient}>
      {children}
      <Toaster position="top-right" richColors />
    </QueryClientProvider>
  );
}

// src/app/layout.tsx

import './globals.css';
import { Providers } from './providers';

export default function RootLayout({ children }: { children: React.ReactNode }) {
  return (
    <html lang="en">
      <body>
        <Providers>{children}</Providers>
      </body>
    </html>
  );
}
```

— End of Frontend Guide —