SIRS Technical Deep Dive

Everything You Need to Know About the Codebase

For internal consumption only - Ernest Moyo, 7Square Inc. Document version: 1.0 | February 2026

TABLE OF CONTENTS

---

1. PROJECT OVERVIEW & ARCHITECTURE

What SIRS Is

SIRS is a **frontend prototype** of a regional disaster risk intelligence platform. It is a **single-page application** (SPA) built with Next.js that demonstrates the user interface and user experience that SADC duty officers, disaster managers, and UN agencies would use to monitor risk, run scenarios, and coordinate responses.

What SIRS Is NOT (Yet)

- There is **no backend server** processing data
- There is **no database** storing risk scores
- There is **no real-time data feed** from meteorological services
- There are **no real API calls** to INFORM, KoboToolbox, or Google Earth Engine
- There is **no authentication system** (the login page is cosmetic)

Everything runs in the browser. All data is hardcoded in TypeScript files.

Architecture Diagram

```
User's Browser
    |
    v
Next.js App (served as static HTML + JS)
    |
    +-- Layout Layer (AppShell, Sidebar, Header)
    |       |
    |       +-- Sidebar.tsx    (navigation)
    |       +-- Header.tsx     (search, alerts, profile)
    |       +-- AppShell.tsx   (wraps everything, provides sidebar context)
    |
    +-- Page Layer (Next.js App Router)
    |       |
    |       +-- page.tsx           --> Dashboard (/)
    |       +-- risk-monitor/      --> Risk Monitor
    |       +-- scenarios/         --> Scenario Builder
    |       +-- triggers/          --> Trigger Layer
    |       +-- sectors/           --> Sector Lenses
    |       +-- data-collection/   --> Kobo Integration
    |       +-- collaboration/     --> Workspace
    |       +-- settings/          --> Configuration
    |       +-- login/             --> Login screen
    |
    +-- Component Layer
    |       |
    |       +-- dashboard/    (7 components for the main dashboard)
    |       +-- scenarios/    (4 components for scenario builder)
    |       +-- triggers/     (3 components for trigger system)
    |       +-- sectors/      (4 components, one per sector lens)
    |       +-- ui/           (4 reusable UI primitives)
    |       +-- layout/       (3 layout components)
    |
    +-- Data Layer
    |       |
    |       +-- mock-data.ts        (ALL the data lives here)
    |       +-- utils.ts            (risk calculation & formatting functions)
```

```
|       +-- demo-data-readme.ts (documentation of data sources)
|
+-- External Services (loaded from CDN, not our servers)
        |
        +-- OpenStreetMap / CARTO tiles (map backgrounds)
        +-- Google Fonts (Inter typeface)
```

How Data Flows

```
mock-data.ts  --->  Component imports data  --->  Component renders UI
                    (e.g., sadcCountries)         (tables, charts, maps)
```

There is no state management library (no Redux, no Zustand). Each component imports what it needs directly from `mock-data.ts`. Interactive state (filters, selected items, expanded panels) is managed with React's `useState` hook locally in each component.

---

2. TECH STACK EXPLAINED

Next.js 16 (App Router)

What it is: A React framework that handles routing, server-side rendering, and build optimization.

Why we use it: It's the industry standard for React applications. The "App Router" means each folder in `src/app/` becomes a URL route:

Folder	URL
`src/app/page.tsx`	`/`
`src/app/scenarios/page.tsx`	`/scenarios`
`src/app/sectors/[lens]/page.tsx`	`/sectors/health-wash` , `/sectors/food-security` , etc.

The `[lens]` folder with brackets means it's a **dynamic route** - the URL segment becomes a variable that the page can read.

TypeScript

What it is: JavaScript with type annotations.

Why it matters: Every piece of data has a defined shape (interface). For example:

```
export interface CountryRisk {
  id: string;           // "moz"
  name: string;         // "Mozambique"
  riskScore: number;    // 0.82 (0-1 scale)
  hazardScore: number;  // 0.88
  // ...etc
}
```

This means if you try to access `country.riskScor` (typo), TypeScript catches it at build time, not when a user is demoing the product.

Tailwind CSS

What it is: A utility-first CSS framework. Instead of writing CSS files, you add classes directly to HTML elements.

Example:

```
<!-- Traditional CSS approach -->
<div class="risk-card">...</div>
<!-- In a separate .css file: .risk-card { background: ...; border: ...; } -->

<!-- Tailwind approach (what we use) -->
<div class="rounded-xl border border-slate-700/50 bg-slate-800/50 p-5">...</div>
```

Every class maps to a single CSS property:

- `rounded-xl` = `border-radius: 12px`
- `border` = `border-width: 1px`
- `border-slate-700/50` = `border-color: rgb(51 65 85 / 0.5)`
- `bg-slate-800/50` = `background: rgb(30 41 59 / 0.5)`
- `p-5` = `padding: 20px`

Leaflet + React Leaflet

What it is: An open-source JavaScript library for interactive maps.

How it works in SIRS:

1. We load map tiles (background images) from CARTO's dark theme tile server
2. We overlay circle markers for each SADC country, sized and colored by risk
3. We draw polylines for the cyclone track (historical + projected)
4. Clicking a marker shows a popup with country details

Recharts

What it is: A React charting library built on D3.js.

Charts we use:

- `AreaChart` - risk trend lines over time (dashboard)
- `BarChart` - scenario comparisons, food security IPC phases
- `PieChart/Donut` - health facility status distribution
- `RadarChart` - risk decomposition (hazard/exposure/vulnerability/coping)
- `LineChart` - trigger value over time

Lucide React

What it is: An icon library with 1000+ clean, consistent SVG icons.

Usage: `<ShieldAlert className="h-4 w-4 text-red-400" />`

---

3. FILE-BY-FILE CODE WALKTHROUGH

`src/lib/utils.ts` - The Brain of Risk Display

This file contains 5 functions that control how risk is displayed everywhere:

`cn(...inputs)` - Class Name Merger

```
cn("bg-red-500", isActive && "text-white", !isActive && "text-gray-500")
// Merges CSS classes intelligently, resolving conflicts
```

Used everywhere to conditionally apply CSS classes.

`formatNumber(num)` - Human-Readable Numbers

```
formatNumber(2_450_000) // returns "2.5M"
formatNumber(45_000)    // returns "45.0K"
formatNumber(500)       // returns "500"
```

Used for population figures, affected counts, etc.

`getRiskColor(score)` - The Color Formula

```
Score >= 0.8  -->  #ef4444  (Red)      = "Critical"
Score >= 0.6  -->  #f97316  (Orange)   = "High"
Score >= 0.4  -->  #eab308  (Yellow)   = "Moderate"
Score >= 0.2  -->  #22c55e  (Green)    = "Low"
Score <  0.2  -->  #3b82f6  (Blue)     = "Minimal"
```

This is the single most important function in the UI. It determines the color of:

- Map markers
- Table cells
- Risk badges
- Chart lines
- Progress bars
- Trigger status indicators

`getRiskLevel(score)` - Text Labels

Same thresholds as above, but returns text: "Critical", "High", "Moderate", "Low", "Minimal"

`getRiskBadgeClass(score)` - Badge Styling

Returns Tailwind classes for colored badges:

- Critical: semi-transparent red background, red text, red border
- High: semi-transparent orange background, orange text, orange border
- etc.

`src/lib/mock-data.ts` - All The Data

This is the most important file to understand. Every number you see in the UI comes from here.

7 data interfaces define the shape of all data:

- `CountryRisk` - national-level risk profiles

- `DistrictRisk` - subnational risk data
- `ActiveEvent` - ongoing disasters
- `TriggerConfig` - early action trigger rules
- `Scenario` - impact scenario definitions
- `Assessment` - field data collection forms
- (plus `riskTimeSeries` and `recentActivities` as plain arrays)

6 exported datasets:

- `sadcCountries` - 16 SADC member states
- `activeEvents` - 4 current disaster events
- `triggers` - 6 partner trigger configurations
- `scenarios` - 4 impact scenarios
- `assessments` - 4 field assessment forms
- `riskTimeSeries` - 10 days of risk data for 4 countries
- `recentActivities` - 7 recent platform actions

`src/app/layout.tsx` - The Root

Every page in the app is wrapped by this file. It:

1. Loads the Inter font from Google Fonts
2. Sets the page `<html>` to dark mode (`className="dark"`)
3. Wraps all content in `<AppShell>` (which provides sidebar + header)

`src/app/page.tsx` - The Dashboard

The simplest page - it just composes 6 components in a grid:

```
Row 1: [RiskOverviewCards] ..... (full width, 4 KPI cards)
Row 2: [RegionalRiskMap (2/3)] [ActiveEventsPanel (1/3)]
Row 3: [RiskTrendChart] ..... (full width)
Row 4: [CountryRiskTable (2/3)] [RecentActivityFeed (1/3)]
```

`src/components/layout/AppShell.tsx` - The Shell

The most structurally important component. It:

1. Creates a React Context (`SidebarContext`) that stores sidebar state
2. The context has: `collapsed` (boolean), `mobileOpen` (boolean), and setters

3. Renders `<Sidebar />` + `<Header />` + `{children}` (page content)
4. Handles keyboard shortcut `Ctrl+B` to toggle sidebar
5. Closes mobile sidebar on route change and window resize

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

The sidebar has these sections (top to bottom):

1. **Logo area** - SIRS logo + "7Square" text
2. **Active events indicator** - Red pulsing badge showing "4 Active Events"
3. **Navigation links** - 7 main routes with icons
4. **Settings link** - At bottom
5. **User area** - Avatar + name + role
6. **Collapse toggle** - Arrow button to shrink sidebar to icons only

Key behavior:

- Active route gets a blue highlight + left border indicator
- When collapsed, hovering shows a tooltip with the page name
- On mobile (<1024px), sidebar slides in as an overlay
- Trigger badge shows count of active triggers (hardcoded to 3)

`src/components/layout/Header.tsx` - Top Bar

Shows:

- **Page title** - Changes based on URL (uses a lookup table)
- **Search bar** - Decorative (doesn't search anything yet)
- **Live clock** - Updates every second showing current time
- **Notification bell** - Dropdown with 4 mock alerts
- **User profile** - Dropdown with name, role, sign-out button

Dashboard Components (7 files)

`RiskOverviewCards.tsx` - KPI Cards

Computes 4 summary numbers from the data:

- Total Affected Population: `sum of all country.affectedPopulation` = ~9.8M
- Active Events: `count where status === "active"` = 3
- Triggered Alerts: `count where trigger.status === "red"` = 2
- Countries at High Risk: `count where riskScore >= 0.6` = 6

The "change" values (+12.4%, +2, +1, -1) are **hardcoded** in the component. In production, these would be computed by comparing current vs. previous period.

`RegionalRiskMap.tsx` + `RegionalRiskMapInner.tsx` - The Map

Split into two files because **Leaflet cannot render on the server** (it needs `window` and `document`). So:

1. `RegionalRiskMap.tsx` uses `dynamic(() => import("./RegionalRiskMapInner"), { ssr: false })` to load the map only in the browser
2. `RegionalRiskMapInner.tsx` contains the actual Leaflet map

Map markers - how size and color are computed:

```
const color = getRiskColor(country.riskScore);  // e.g., red for 0.82
const radius = Math.max(8, country.riskScore * 28);  // e.g., 23px for 0.82
```

So higher risk = bigger circle AND redder color. Minimum size is 8px (for very low-risk countries like Seychelles).

Cyclone track visualization:

- Solid purple line: historical track (where the cyclone has been)
- Dashed purple line: projected track (where it's heading)
- Purple circle: current cyclone position

The coordinates are hardcoded to represent a realistic path from the Indian Ocean toward Sofala province, Mozambique.

Map popup on click: Shows country name, risk level badge, risk score (scaled to 0-10), affected population, active alerts, and a visual breakdown bar of hazard/exposure/vulnerability.

`RiskTrendChart.tsx` - Area Chart

Plots risk scores for 4 countries over 10 days (Feb 1-10).

- Each country has a unique color (Mozambique=red, Malawi=orange, Madagascar=yellow, Tanzania=blue)
- Gradient fill under each line for visual depth
- Custom dark-themed tooltip

The Y-axis displays values as 0-10 (by multiplying the 0-1 scores by 10).

`ActiveEventsPanel.tsx` - Event Cards

Lists 4 active events with:

- Type icon (cyclone=wind, flood=water, drought=sun)
- Severity badge (critical/high/moderate)

- Status dot (active=red, watch=amber)
- Click to expand and see description

`CountryRiskTable.tsx` - Sortable Table

Displays all 16 SADC countries in a sortable table. Click any column header to sort ascending/descending. Default sort: risk score descending.

The risk score column shows both a colored progress bar AND the numeric value (scaled 0-10).

`RecentActivityFeed.tsx` - Timeline

Shows 7 recent actions with a vertical timeline. Each type has its own icon and color:

- Scenario = purple zap icon
- Trigger = red bell icon
- Assessment = blue file icon
- Advisory = amber megaphone icon
- Data = green database icon

Trigger Components (3 files)

`TriggerCard.tsx`

Each trigger gets a card showing:

- Status indicator (large colored circle with glow/pulse animation)
- Organization name and country
- Current value vs. threshold as a progress bar
- Sparkline (small inline chart) showing recent trend
- Percentage of threshold reached
- Action buttons (Configure, History, Acknowledge)

The pulse animation: Red triggers get `animate-pulse-red` which creates a radiating ring effect. Amber triggers get a slower pulse. Green triggers have no animation.

`TriggerTimeline.tsx`

Full-screen modal showing a trigger's detailed history:

- Large Recharts LineChart with the value over time
- Threshold drawn as a horizontal dashed line
- Color zones (green below readiness, amber in readiness, red above activation)
- Data table of all historical values

`TriggerConfigForm.tsx`

A form for creating new triggers:

- Select organization, country, metric
- Set threshold with a slider
- Configure readiness/activation/stand-down levels
- Multi-step preview

Scenario Components (4 files)

`ScenarioBuilder.tsx` - Creation Wizard

4-step wizard:

1. Select hazard type (cyclone/flood/drought) with visual cards
2. Configure parameters (wind speed, rainfall, etc.) with sliders
3. Select geography (country, districts)
4. Review and run

`ScenarioCard.tsx` - Scenario Display

Shows each saved scenario with type icon, status badge, key metrics, and action buttons (View, Compare, Export).

`ScenarioComparison.tsx` - Side-by-Side

Compares two scenarios showing:

- Affected population comparison (bar chart)
- Facilities at risk comparison
- District-level impact differences

`ScenarioResults.tsx` - Results View

Shows scenario output: summary cards, district impact table, facility breakdown, export buttons.

Sector Lens Components (4 files)

Each sector lens provides a domain-specific view of the risk data:

`HealthWashLens.tsx` (accent: emerald)

- Health facility status: operational / at-risk / non-functional
- Donut chart of facility status distribution
- WASH infrastructure impact
- Patient populations affected

`FoodSecurityLens.tsx` (accent: amber)

- IPC phase classification (phases 1-5 with severity colors)
- Stacked bar chart of affected population by IPC phase
- Crop loss estimates by district

- Horizontal bar chart of crop loss percentages

`LogisticsLens.tsx` **(accent: cyan)**

- Road network status (accessible / restricted / cut-off)
- Bridge status indicators
- Supply corridor bar chart
- Transport infrastructure summary

`SocialProtectionLens.tsx` **(accent: purple)**

- Eligible households for cash transfers
- Beneficiary coverage donut chart
- Transfer recommendations by district
- Coverage gap bar chart

---

4. THE RISK MODEL: HOW SCORES WORK

The INFORM Framework (Real Methodology)

SIRS is built on the **INFORM Subnational Risk Index** methodology. In the real INFORM model:

```
Risk = Hazard & Exposure x Vulnerability x Lack of Coping Capacity
```

Each dimension is scored 0-10 (we normalize to 0-1 internally):

- **Hazard & Exposure**: How likely is a disaster AND how many people/assets are exposed?
  - Natural hazards: earthquakes, floods, cyclones, drought
  - Human hazards: conflict, displacement
- **Vulnerability**: How susceptible is the population?
  - Socio-economic: poverty, inequality, development
  - Vulnerable groups: uprooted people, children, elderly
- **Lack of Coping Capacity**: How unable is the country/region to cope?
  - Institutional: governance, DRR infrastructure
  - Infrastructure: health system, communications, physical

How We Simulate Scores

In our prototype, scores are **manually assigned** based on general knowledge of SADC countries' risk profiles. They are NOT computed from a formula.

Why Mozambique has 0.82 (Critical):

- Cyclone-prone coastline (high hazard)
- Large exposed population in coastal lowlands (high exposure)
- High poverty rates, limited infrastructure (high vulnerability)
- Limited disaster management capacity (low coping)

Why Botswana has 0.31 (Low):

- Inland, no cyclone exposure (moderate hazard - mainly drought)
- Small population, less concentrated (lower exposure)
- Higher GDP per capita (lower vulnerability)
- Better institutional capacity (higher coping)

Score-to-Color Mapping

```
Score Range   Level      Color    Hex       Meaning
-----------   --------   ------   -------   --------------------------------
0.80 - 1.00   Critical   Red      #ef4444   Immediate action required
0.60 - 0.79   High       Orange   #f97316   Significant risk, active monitoring
0.40 - 0.59   Moderate   Yellow   #eab308   Elevated risk, preparedness needed
0.20 - 0.39   Low        Green    #22c55e   Below-average risk
0.00 - 0.19   Minimal    Blue     #3b82f6   Very low risk
```

Display Scale

Internally, all scores are 0-1. When displayed to users, they're multiplied by 10 to show as 0-10 (matching INFORM's public scale):

```
(country.riskScore * 10).toFixed(1)  // 0.82 displays as "8.2"
```

Map Marker Size Formula

```
const radius = Math.max(8, country.riskScore * 28);
```

Country	Risk Score	Radius (px)	Visual
Mozambique	0.82	23	Large

Seychelles	0.25	8 (minimum)	Small
DR Congo	0.78	22	Large
Botswana	0.31	9	Small

---

5. DATA SIMULATION: WHAT'S FAKE AND WHY

Country Data (`sadcCountries`)

All 16 SADC member states are listed with:

Field	Source	Accuracy
`name , code`	Real	100% accurate
`lat , lng`	Real	Approximate country centroids
`population`	Real-ish	Based on recent estimates, not exact
`riskScore`	Simulated	Informed by general INFORM trends
`hazardScore`	Simulated	Reflects known hazard exposure
`exposureScore`	Simulated	Reflects population distribution
`vulnerabilityScore`	Simulated	Reflects development indicators
`copingCapacity`	Simulated	Reflects institutional strength
`affectedPopulation`	Simulated	Realistic for current season
`activeAlerts`	Simulated	Arbitrary for demo
`districts`	Simulated	Only 3-4 countries have district data

Why only some countries have district data: Only Mozambique (4 districts), Malawi (3 districts), Madagascar (2 districts), and Tanzania (1 district) have sub-national data. These are the "pilot countries" mentioned in the concept note. In production, all 16 countries would have subnational data.

District Data

Districts are real administrative areas with approximately correct coordinates:

- **Sofala, Mozambique**: Highest risk (0.91) because it's the Beira cyclone corridor
- **Nsanje, Malawi**: High risk (0.88) because it's in the Lower Shire flood zone
- **Dar es Salaam, Tanzania**: Urban flood risk due to informal settlements

Facility counts (health clinics, WASH points, schools) are **entirely simulated** but proportional to district populations.

Active Events

The 4 events are **fictional but realistic**:

1. **Tropical Cyclone Batsirai II** - Named after the real Cyclone Batsirai (2022). "Batsirai II" is fictional. The Category 3 intensity, Mozambique Channel path, and Sofala landfall are all realistic scenarios.
2. **Zambezi Basin Flooding** - Multi-country flooding is a recurring reality. The Zambezi basin regularly floods affecting Malawi, Mozambique, Zambia, Zimbabwe.
3. **Southern Madagascar Drought** - The Grand Sud drought is a real, ongoing crisis. IPC Phase 3 conditions are realistic for this region.
4. **Dar es Salaam Urban Flooding** - Dar es Salaam floods regularly during heavy rains due to inadequate drainage in informal settlements.

Trigger Data

The 6 triggers represent **realistic partner configurations:**

Trigger	Organization	Why It's Realistic
WFP AA Malawi	WFP	WFP runs anticipatory action programs in Malawi
IFRC DREF Mozambique	IFRC	IFRC issues DREF allocations for cyclones
NDMA Readiness Zambezia	NDMA Moz	National agencies set readiness levels
WFP Food Security Madagascar	WFP	WFP monitors food insecurity in Madagascar
IFRC Volunteer Tanzania	IFRC	Red Cross deploys volunteers before events
SADC Standby Regional	SADC	SADC has regional standby mechanisms

The **threshold values** (0.55 to 0.75) and **current values** are simulated to demonstrate all three states: green (below), amber (approaching), red (exceeded).

Risk Time Series

The 10-day trend data (Feb 1-10) shows a **rising pattern** for all 4 countries, simulating the approach of a cyclone season event. This is realistic - risk scores would increase as a cyclone approaches and rainfall intensifies.

Assessments

The 4 assessments represent typical KoboToolbox deployments:

- Rapid damage assessments (post-cyclone)
- Needs assessments (flood response)
- Facility status checks (health system monitoring)
- Coping capacity surveys (baseline building)

Response counts and targets are realistic for their type.

---

6. THE MAP SYSTEM: LEAFLET INTEGRATION

Why Two Files?

Leaflet uses `window` and `document` objects that don't exist during server-side rendering. Next.js tries to render pages on the server first. Solution:

```
// RegionalRiskMap.tsx (this file CAN render on server)
const MapInner = dynamic(() => import("./RegionalRiskMapInner"), {
  ssr: false,  // <-- "Don't try to render this on the server"
  loading: () => <LoadingSpinner />,  // Show while loading
});
```

Map Tiles

We use CARTO's dark-themed tiles:

```
https://{s}.basemaps.cartocdn.com/dark_nolabels/{z}/{x}/{y}{r}.png
https://{s}.basemaps.cartocdn.com/dark_only_labels/{z}/{x}/{y}{r}.png
```

Two layers: one for the geography (dark), one for labels on top. This gives us a clean dark look that matches the dashboard theme.

Map Center & Zoom

```
center={[-15, 30]}  // Southern Africa
zoom={4}            // Shows the whole SADC region
```

Scroll Behavior

```
map.scrollWheelZoom.disable();  // Prevent accidental zooming
map.on("focus", () => map.scrollWheelZoom.enable());  // Enable when clicked
```

Cyclone Track Coordinates

The track follows a realistic path for an Indian Ocean cyclone:

```
Historical:  (-14.0, 48.0) --> (-17.5, 42.3)   [East to West, into Channel]
Projected:   (-17.5, 42.3) --> (-19.2, 35.8)   [Channel to Sofala coast]
```

The dashed line (`dashArray: "8, 8"`) visually distinguishes the forecast from observed positions.

Map Popups (Leaflet Dark Theme)

The default Leaflet popups are white. We override them in `globals.css` :

```
.leaflet-popup-content-wrapper {
  background: #1e293b !important;  /* Dark background */
  color: #e2e8f0 !important;       /* Light text */
  border: 1px solid rgba(100, 116, 139, 0.3);
}
```

---

7. CHARTS & VISUALIZATIONS: RECHARTS

Risk Trend Area Chart (Dashboard)

```
<AreaChart data={riskTimeSeries}>
  <Area dataKey="mozambique" stroke="#ef4444" fill="url(#gradient-mozambique)"
/>
  <Area dataKey="malawi"     stroke="#f97316" fill="url(#gradient-malawi)" />
```

```
  // ... etc
</AreaChart>
```

Gradient fills: Each country line has a gradient that goes from 25% opacity at the top to 0% at the bottom, creating a subtle "glow under the line" effect.

Custom tooltip: Instead of Recharts' default tooltip, we render our own dark-themed tooltip component that shows country names and scores.

Trigger Sparklines (Trigger Cards)

Small inline charts using Recharts `LineChart` with no axes, no grid, no labels. Just a tiny line showing the trend. Red fills for activated triggers.

Sector Lens Charts

Each lens uses different chart types:

- **Health**: `PieChart` (donut) for facility status distribution
- **Food Security**: `BarChart` (stacked) for IPC phase populations
- **Logistics**: `BarChart` for supply corridor status
- **Social Protection**: `PieChart` (donut) + `BarChart` for coverage

Radar Chart (Risk Monitor)

The risk monitor page uses a `RadarChart` to show the 4 dimensions of risk for a selected country:

```
Axes: Hazard, Exposure, Vulnerability, Coping Capacity
Shape: Filled polygon showing relative strengths/weaknesses
```

---

8. THE TRIGGER SYSTEM: TRAFFIC LIGHTS FOR DISASTER RESPONSE

Concept

The trigger system implements a simple state machine:

```
GREEN  --[value approaches threshold]-->  AMBER  --[value exceeds threshold]-->
RED
```

```
RED    --[value drops below threshold]-->  AMBER  --[value drops well below]--->
GREEN
```

How Status is Determined

```
interface TriggerConfig {
  threshold: number;      // e.g., 0.70
  currentValue: number;   // e.g., 0.83
  status: "green" | "amber" | "red";
}
```

In the current prototype, status is **hardcoded** in the mock data.

In production, the status would be computed:

```
If currentValue >= threshold:           RED    (threshold exceeded)
If currentValue >= threshold * 0.85:    AMBER  (within 15% of threshold)
Else:                                   GREEN  (below threshold)
```

Visual Indicators

- **RED**: Pulsing red circle with radiating ring animation (`animate-pulse-red`)
- **AMBER**: Slower pulsing amber circle (`animate-pulse-amber`)
- **GREEN**: Solid green circle, no animation

Partner-Specific Triggers

Each partner defines their own rules:

- **WFP**: "Activate anticipatory cash transfers when flood risk > 0.70 for 2+ days"
- **IFRC**: "Deploy DREF when cyclone impact > 0.65"
- **NDMA**: "Enter readiness mode when composite risk > 0.60"

The trigger system allows multiple partners to monitor the SAME underlying risk data but with DIFFERENT thresholds, enabling coordinated but independent decision-making.

---

9. SECTOR LENSES: DOMAIN-SPECIFIC VIEWS

Concept

The base risk picture is the same for everyone. Sector lenses add specialized analysis for specific domains. Each lens has:

1. **Its own accent color** (emerald, amber, cyan, purple)
2. **Domain-specific metrics** (health facilities, crop loss, road status, etc.)
3. **Relevant data overlays** (facility registries, IPC classifications, etc.)

Health/WASH Lens Example

The Health lens shows health facility data from the mock district data:

```
// Aggregating facility data across all districts
facilities: {
  health: 145,       // Total health facilities
  healthAtRisk: 67,  // Facilities in flood/cyclone zone
  wash: 312,         // WASH infrastructure points
  washAtRisk: 134,   // WASH points at risk
}
```

The donut chart divides facilities into:

- Operational (total - atRisk)
- At Risk (atRisk number)
- Non-functional (a percentage of atRisk, estimated)

Dynamic Routing

The sector hub (`/sectors`) shows 4 cards. Clicking "Enter Lens" navigates to `/sectors/health-wash` , `/sectors/food-security` , etc.

The `[lens]` dynamic route reads the URL parameter:

```
// In /sectors/[lens]/page.tsx
export default function LensPage({ params }: { params: { lens: string } }) {
  // params.lens = "health-wash" or "food-security" etc.
}
```

---

10. LAYOUT SYSTEM: SIDEBAR, HEADER, ROUTING

The AppShell Pattern

Sidebar Context

The sidebar state is shared via React Context:

```
const SidebarContext = createContext({
  collapsed: false,        // Desktop: narrow icon-only mode
  setCollapsed: () => {},
  mobileOpen: false,       // Mobile: overlay sidebar
  setMobileOpen: () => {},
});
```

Any component can read `useSidebar()` to know the sidebar state and react to it.

Responsive Behavior

Screen Width	Sidebar Behavior
< 1024px (mobile/tablet)	Hidden. Hamburger button opens as overlay
>= 1024px, not collapsed	Fixed 256px wide sidebar
>= 1024px, collapsed	Fixed 68px wide icon-only sidebar

The main content area adjusts with:

```
className={collapsed ? "lg:ml-[68px]" : "lg:ml-64"}
```

Page Title System

The header reads the current URL and looks up the title:

```
const pageTitles = {
  "/": "Dashboard",
  "/risk-monitor": "Risk Monitor",
  "/scenarios": "Scenarios",
  // ...
};
const title = pageTitles[pathname] || "Dashboard";
```

---

11. STYLING SYSTEM: TAILWIND + CSS ARCHITECTURE

CSS Variables (globals.css)

We define custom properties for the SIRS brand:

```
:root {
  --sirs-bg: #020617;           /* Near-black background */
  --sirs-card: rgba(30, 41, 59, 0.5);  /* Semi-transparent card */
  --sirs-primary: #3b82f6;      /* Blue accent */
  --sirs-risk-critical: #ef4444; /* Red */
  --sirs-risk-high: #f97316;    /* Orange */
  /* ...etc */
}
```

Custom Animations

6 custom animations defined in CSS:

- `sirs-fade-in` : Slide up + fade in (for page transitions)
- `sirs-slide-in-right/left` : Horizontal slide for panels

- `sirs-scale-in` : Zoom-in effect
- `sirs-pulse-ring` : Radiating ring (for critical status)
- `sirs-shimmer` : Loading skeleton effect
- `sirs-glow` : Pulsing box shadow

Plus trigger-specific animations:

- `pulse-red` : Radiating red ring for active triggers
- `pulse-amber` : Slower amber pulse for warnings
- `glow-red/glow-amber` : Drop shadow pulse effects

The Glass Effect

Many cards use a "glassmorphism" style:

```
.glass {
  background: rgba(15, 23, 42, 0.6);  /* Semi-transparent */
  backdrop-filter: blur(12px);          /* Blurs content behind */
  border: 1px solid rgba(51, 65, 85, 0.3);  /* Subtle border */
}
```

Reusable CSS Classes

```
.sirs-card              /* Standard card with backdrop blur */
.sirs-card-interactive  /* Card with hover effects */
.risk-critical/high/moderate/low  /* Pre-styled risk badges */
.gradient-text-brand    /* Blue-to-purple gradient text */
```

---

12. WHAT'S REAL VS WHAT WOULD NEED TO BE BUILT FOR PRODUCTION

Currently Real / Working

Feature	Status

Full UI/UX with all pages	Working
Interactive Leaflet map	Working (real OSM tiles)
Charts and visualizations	Working (with mock data)
Responsive layout (mobile + desktop)	Working
Sidebar navigation + routing	Working
Trigger traffic light system	Working (visual only)
Scenario builder wizard	Working (UI only, no computation)
Dark theme + brand styling	Working

Would Need to Be Built

Feature	Complexity	Priority
Backend API server (Node.js/Python)	High	Phase 1
Database (PostgreSQL + PostGIS)	High	Phase 1
INFORM API integration (baseline risk scores)	Medium	Phase 0
Meteorological data feeds (RSMC, national services)	High	Phase 1
Real risk computation engine (composite scores)	High	Phase 1
Scenario computation engine (hazard models)	Very High	Phase 1
KoboToolbox API integration (form deployment + data)	Medium	Phase 1
Google Earth Engine (satellite imagery)	High	Phase 1
Authentication system (SADC SSO, UN Identity)	Medium	Phase 1
Role-based access control	Medium	Phase 2
DHIS2 integration (health facility data)	Medium	Phase 2
OGC services (WMS/WFS for GIS interoperability)	Medium	Phase 2
Real-time WebSocket updates	Medium	Phase 1
PDF/PPTX export (report generation)	Medium	Phase 2
Notification system (email/SMS alerts)	Medium	Phase 1

---

13. HOW TO EXTEND: ADDING NEW FEATURES

Adding a New Page

1. Create `src/app/your-page/page.tsx`
2. Add a navigation item in `Sidebar.tsx` (in the `navItems` array)
3. Add title/description in `Header.tsx` (in `pageTitles` and `pageDescriptions`)

Adding New Mock Data

1. Define the interface in `mock-data.ts`
2. Export the data array
3. Import it in your component: `import { yourData } from "@/lib/mock-data"`

Adding a New Chart

1. Import from Recharts: `import { BarChart, Bar, XAxis, ... } from "recharts"`
2. Wrap in `<ResponsiveContainer width="100%" height={300}>`
3. Style with dark theme colors (see existing charts for patterns)

Changing Risk Thresholds

Edit `src/lib/utils.ts`:

```
export function getRiskColor(score: number): string {
  if (score >= 0.8) return "#ef4444";  // Change these thresholds
  if (score >= 0.6) return "#f97316";
  // ...
}
```

This single change affects the entire application's risk visualization.

Adding a New Sector Lens

1. Create `src/components/sectors/YourLens.tsx`
2. Add it to the switch statement in `src/app/sectors/[lens]/page.tsx`
3. Add a card for it on `src/app/sectors/page.tsx`

APPENDIX: COMPLETE FILE LIST

```
app/src/
  app/
    globals.css                          (420 lines - all custom CSS)
    layout.tsx                           (46 lines - root layout)
    page.tsx                             (51 lines - dashboard composition)
    collaboration/page.tsx               (collaboration workspace)
    data-collection/page.tsx             (Kobo integration)
    login/page.tsx                       (login screen)
    risk-monitor/page.tsx                (detailed risk profiles)
    scenarios/page.tsx                   (scenario builder hub)
    sectors/page.tsx                     (sector lens hub)
    sectors/[lens]/page.tsx              (individual lens detail)
    settings/page.tsx                    (configuration)
    triggers/page.tsx                    (trigger monitoring)

  components/
    layout/
      AppShell.tsx                       (84 lines - shell + context)
      Sidebar.tsx                        (312 lines - navigation)
      Header.tsx                         (380 lines - top bar)
    dashboard/
      RiskOverviewCards.tsx              (129 lines - KPI cards)
      RegionalRiskMap.tsx                (67 lines - map wrapper)
      RegionalRiskMapInner.tsx           (237 lines - Leaflet map)
      RiskTrendChart.tsx                 (158 lines - area chart)
      ActiveEventsPanel.tsx              (152 lines - event cards)
      CountryRiskTable.tsx               (206 lines - sortable table)
      RecentActivityFeed.tsx             (104 lines - activity timeline)
    scenarios/
      ScenarioBuilder.tsx                (scenario creation wizard)
      ScenarioCard.tsx                   (scenario display card)
      ScenarioComparison.tsx             (side-by-side comparison)
      ScenarioResults.tsx                (results display)
    triggers/
      TriggerCard.tsx                    (trigger monitoring card)
      TriggerTimeline.tsx                (trigger history modal)
      TriggerConfigForm.tsx              (trigger configuration form)
    sectors/
      HealthWashLens.tsx                 (health/WASH analytics)
      FoodSecurityLens.tsx               (food security analytics)
```

```
    LogisticsLens.tsx                (logistics analytics)
    SocialProtectionLens.tsx         (social protection analytics)
  ui/
    Badge.tsx                        (reusable badge)
    Button.tsx                       (reusable button)
    Card.tsx                         (reusable card)
    ProgressBar.tsx                  (reusable progress bar)

lib/
  mock-data.ts                       (382 lines - ALL data)
  utils.ts                           (36 lines - risk functions)
  demo-data-readme.ts                (125 lines - data docs)
```
