Comprehensive Design System Analysis & Decisions

Current State Analysis

Portfolio Website System

Theme: Dark, Tech-forward, Bioinformatics-inspired **Fonts**: - Inter (300,400) - Body text - Space Grotesk (500) - Headlines
- JetBrains Mono (400,600) - Code/technical - IBM Plex Sans (300,500) - Accents

Colors: - Background: `#0b0b0b` (deep black) - Text: `white` - Accent system: (needs documentation)

Layout Grid: - Anchor Points: 8%, 16.5%, 25%, 50%, 75%, 92% - Zones: Left (8-25%), Main (25-75%), Right (75-92%) - Center Spine: 50%

Interactive Elements: - DNA shader background (Three.js) - Advanced debug overlay (Ctrl+G, Ctrl+B) - Smooth scrolling sections

---

Research Reports System

Theme: Light, Scientific, Academic **Fonts**: - Inter (300-800) - Complete weight range - Typography hierarchy with semantic scaling

Colors: - Background: `white` - Primary text: `#1f2937` (dark gray) - Secondary text: `#374151` (medium gray)
- Headers: `#111827` (near black)

Layout: - CSS Custom Properties spacing system - Scientific content optimized - Chart-first design

Interactive Elements: - D3.js visualizations - Custom buttons and hover states - Data-driven interactions

---

The Integration Challenge

Current Conflicts:

1. **Color Systems**: Dark vs Light themes
2. **Font Hierarchies**: Different font families & weights

3. **Layout Systems**: Portfolio grid vs Report flow
4. **CSS Specificity**: Global styles overriding each other
5. **Interaction Systems**: Three.js vs D3.js co-existence

Design System Decisions

1. Font Hierarchy System

Portfolio Context (Dark Theme)

```
/* Headlines & Navigation */
font-family: "Space Grotesk", sans-serif;
font-weight: 500;
letter-spacing: -0.02em;

/* Body Text */
font-family: "Inter", sans-serif;
font-weight: 300; /* Light for dark backgrounds */

/* Technical/Code */
font-family: "JetBrains Mono", monospace;
font-weight: 400-600;
```

Report Context (Light Theme)

```
/* Headers */
font-family: "Inter", sans-serif;
font-weight: 600-700; /* Heavier for light backgrounds */
color: #111827;

/* Body */
font-family: "Inter", sans-serif;
font-weight: 400; /* Standard weight */
color: #374151;

/* Data/Statistics */
font-family: "Inter", sans-serif;
font-weight: 500-600;
color: #1f2937;
```

2. Color System Standardization

Portfolio Palette

- Primary BG: `#0b0b0b`
- Text: `white`

- Accent Blue: `#3b82f6`
- Accent Green: `#10b981`
- Warning: `#f59e0b`

Report Palette

- Primary BG: `white`
- Text Primary: `#1f2937`
- Text Secondary: `#374151`
- Headers: `#111827`
- Accent Blue: `#2563eb`
- Success Green: `#059669`
- Chart colors: D3 Category10

3. Spacing System Unification

Base Scale (for both contexts)

```
:root {
  --space-3xs: 0.25rem;  /* 4px */
  --space-2xs: 0.5rem;   /* 8px */
  --space-xs: 0.75rem;   /* 12px */
  --space-sm: 1rem;      /* 16px */
  --space-md: 1.5rem;    /* 24px */
  --space-lg: 2rem;      /* 32px */
  --space-xl: 2.5rem;    /* 40px */
  --space-2xl: 3rem;     /* 48px */
  --space-3xl: 4rem;     /* 64px */
}
```

Shader Background Integration Options

Option A: Contextual Themes

- **Portfolio sections**: Full DNA shader (current)
- **Report sections**: Subtle shader overlay with light theme adaptation

Option B: Adaptive Shader

- Dynamic shader that shifts color/intensity based on content
- Dark → Light gradient transitions
- Maintain DNA structure but adjust materials

Option C: Layered System

- Background DNA at reduced opacity for reports
- Foreground content with proper contrast
- Interactive zones that reveal more shader detail

Option D: Report-Specific Shader

- Scientific visualization shader for reports
- Data-inspired patterns instead of DNA
- Molecular diagrams or network graphs

Implementation Strategy

Phase 1: Design System Foundation

1. Create unified CSS custom properties
2. Establish component isolation patterns

3. Set up theme switching utilities

Phase 2: Shader Integration Experiments

1. Test shader opacity/color variations
2. Create theme-adaptive materials
3. Implement smooth transitions

Phase 3: Report Reconstruction

1. Use debug tools for perfect alignment
2. Rebuild components with new system
3. Test all interactions and hover states

Phase 4: Polish & Optimization

1. Performance optimization
2. Accessibility improvements

3. Mobile responsiveness

Component Architecture Evolution

Site-Level vs Project-Level Components

Site-Level Components (`/src/components/common/`)

- **Purpose**: Shared across entire portfolio website
- **Examples**: `InsightHighlight`, `SectionHeader`
- **Styling**: Global CSS with proper specificity
- **Use Case**: Hero sections, About page, Work page

Project-Level Components (`/src/components/projects/common/`)

- **Purpose**: Shared across scientific reports/projects
- **Examples**: `PipelineStepCard`, `MethodCard`, `StatCard`
- **Styling**: Dual-mode components (HL-specific + Generic)
- **Use Case**: Statistical reports, research presentations

Dual-Mode Component Pattern

Key Innovation: Components that detect CSS class prefixes to adapt their structure:

```
// Example: MethodCard component
if (className.includes('hl-method-card')) {
  // Use HL-specific structure with existing CSS
  return (
    <div className={`${className} ${isFlipped ? 'flipped' : ''}`}>
      <div className="hl-card-inner">
        <div className="hl-card-front">
          <div className="hl-method-header">
```

```
            <div className="hl-method-icon">{icon}</div>
            <h4>{title}</h4>
          </div>
        </div>
      </div>
    </div>
  );
}

// Otherwise use generic structure
return (
  <div className={`method-card ${isFlipped ? 'flipped' : ''} ${className}`}>
    <div className="card-inner">
      // Generic structure
    </div>
  </div>
);
```

Benefits: - **Backward Compatibility**: Existing components maintain exact appearance - **Reusability**: Same component works for future projects - **No CSS Conflicts**: Each mode uses its own CSS classes - **Type Safety**: TypeScript props with proper interfaces

---

Current Implementation Status

Completed Sections

1. **Hero Section**: Refactored with site-level components
   - Uses: `StatCard`, `ContentCard`, `DisclaimerCard`
   - Status: Visual fidelity maintained, reusable components created
2. **Methods Section**: Refactored with project-level components
   - Uses: `PipelineStepCard` (4 instances), `MethodCard` (6 instances)
   - Status: 409 lines → Clean component usage, dual-mode pattern established

In Progress

3. **Demographics Section**: Next target for refactoring
   - Current: 550+ lines with duplicated D3 logic
   - Opportunity: Extract D3 chart components
   - Challenge: Complex data visualization patterns

Pending Sections

4. **Risk Factors**: Placeholder section
5. **Comparison**: Placeholder section
6. **Forest Plot**: Placeholder section
7. **Model Statistics**: Placeholder section

Key Lessons & Patterns Established

Component Extraction Strategy

1. **Analyze existing patterns** - Look for repeated JSX structures
2. **Preserve visual fidelity** - No changes to appearance during refactoring
3. **Create dual-mode components** - Support existing CSS + generic usage
4. **Extract complex props** - Handle arrays, objects, code snippets
5. **Maintain state management** - Parent components handle flip states

CSS Architecture Lessons

- **CSS specificity battles** solved with component encapsulation
- **Dual-mode detection** prevents style conflicts
- **Project-level organization** separate from site-level components
- **Export barrel pattern** for clean imports

Development Workflow

- **Hot module replacement** enables rapid iteration
- **Component props** make complex content manageable
- **TypeScript interfaces** ensure prop consistency
- **Progress logging** critical for tracking architectural decisions

Next Steps

1. **Shader Experiments**: Create 3-4 variations to test with report
2. **Component Isolation**: Build bulletproof CSS scoping **SOLVED**
3. **Demographics D3 Components**: Extract chart visualizations
4. **Statistical Visualization Library**: Build reusable D3 patterns
5. **Debug-Driven Development**: Use existing tools for precision

---

This document will evolve as we make design decisions and implement the integration.