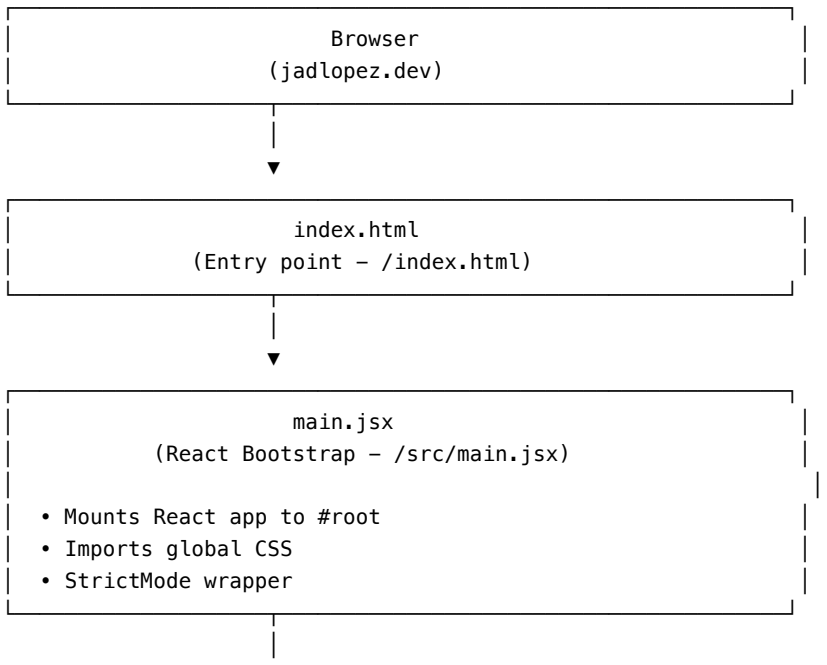
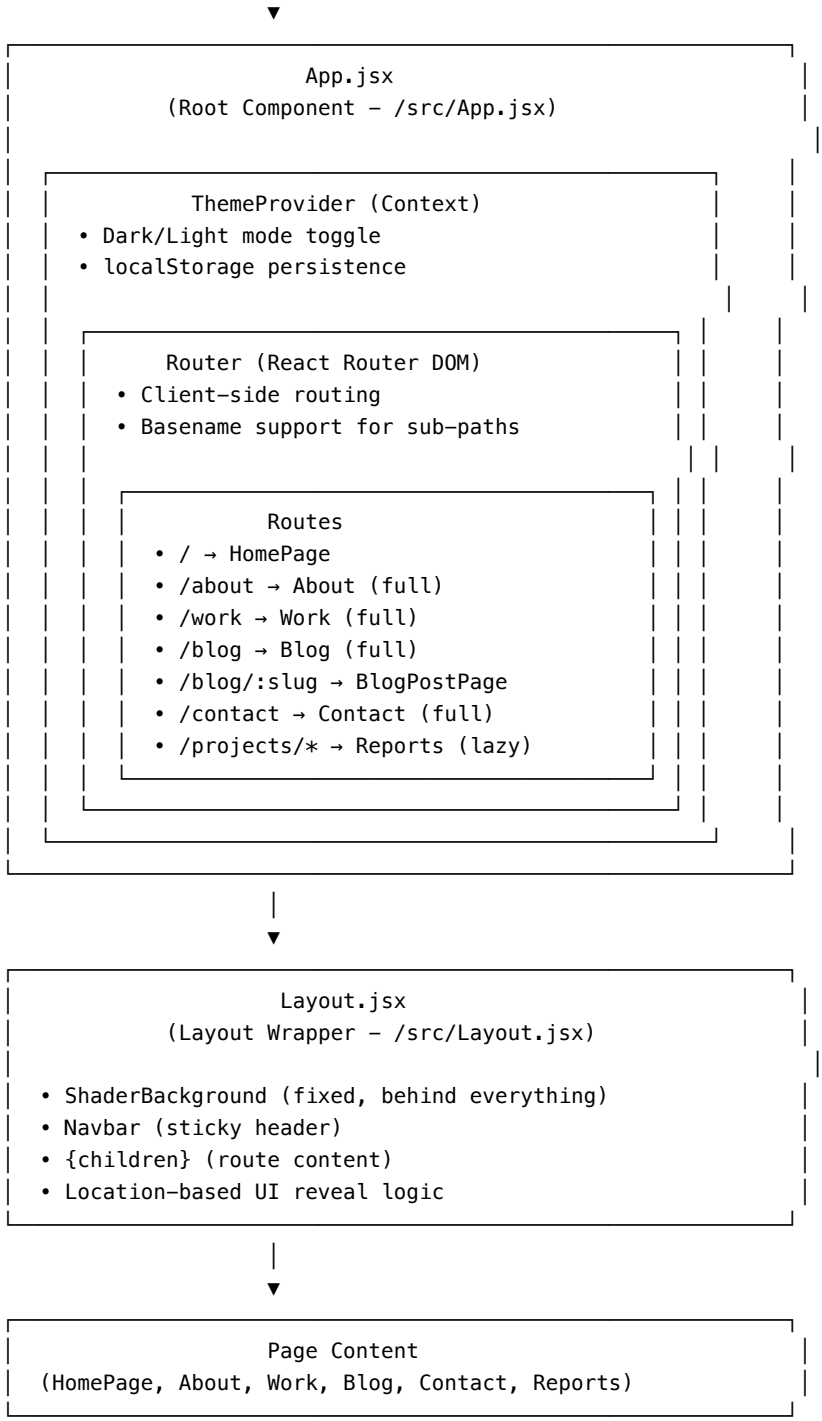
Portfolio Architecture

Last Updated: January 13, 2026 **Related Docs:** ROUTING.md | STATE-MANAGEMENT.md | MASTER-OVERVIEW

Table of Contents

High-Level Architecture




App.jsx
(Root Component – /src/App.jsx)
ThemeProvider (Context)
• Dark/Light mode toggle
• localStorage persistence
Router (React Router DOM)
• Client-side routing
• Basename support for sub-paths
Routes
• / → HomePage
• /about → About (full)
• /work → Work (full)
• /blog → Blog (full)
• /blog/:slug → BlogPostPage
• /contact → Contact (full)
• /projects/* → Reports (lazy)
Layout.jsx
(Layout Wrapper – /src/Layout.jsx)
• ShaderBackground (fixed, behind everything)
• Navbar (sticky header)
• {children} (route content)
• Location-based UI reveal logic
Page Content
(HomePage, About, Work, Blog, Contact, Reports)

Component Hierarchy

Full Application Tree

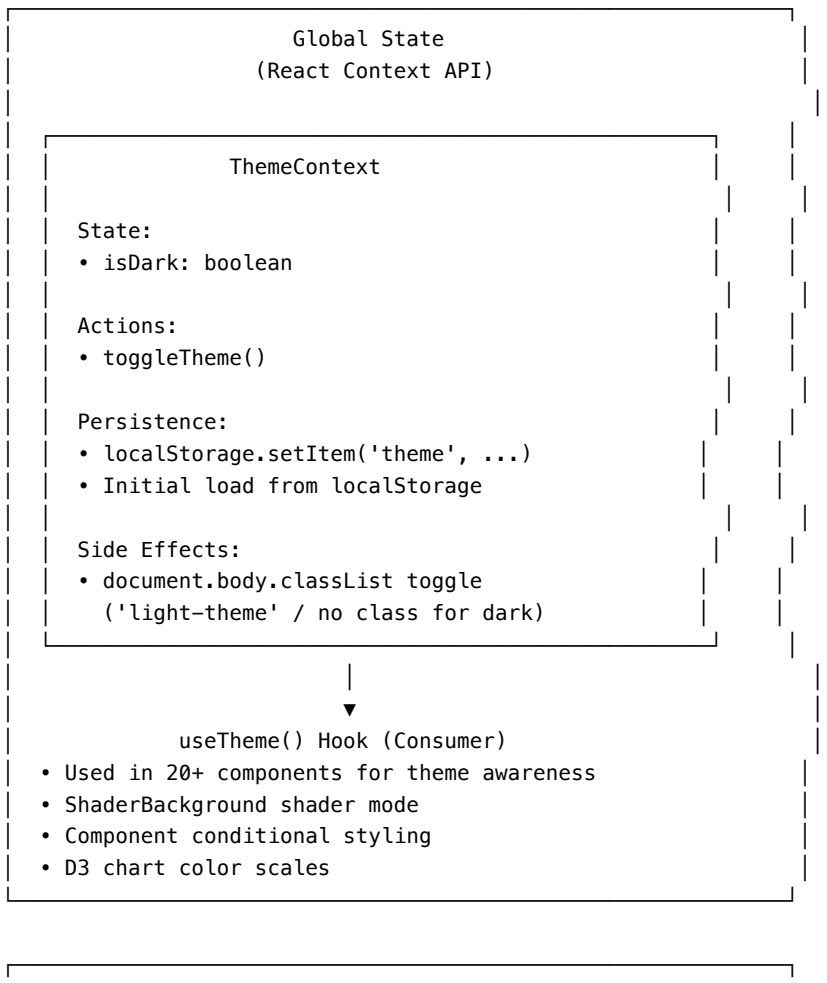
```
App (Root)
│
├── ThemeProvider (Context)
│   │
│   └── Router (React Router)
│       │
│       └── Routes
│           │
│           ├── Route: / (Home)
│           │   └── Layout
│           │       ├── ShaderBackground
│           │       │   ├── Canvas (React Three Fiber)
│           │       │   └── TopographicCytoplasmPlane
│           │       │       └── FloatingParticles (if dark mode)
│           │       │
│           │       ├── Navbar
│           │       │   ├── Logo/Brand
│           │       │   ├── Nav Links
│           │       │   └── ThemeToggle
│           │       │
│           │       └── HomePage
│           │           ├── Hero Section
│           │           │   ├── DNAHelixScene (Three.js)
│           │           │   │   └── DNA Component
│           │           │   ├── HeroContent
│           │           │   └── InfoPanels
│           │           │
│           │           ├── About Section (Merged)
│           │           │   ├── AboutIntroMerged
│           │           │   │   ├── AboutIntro (Hola)
│           │           │   │   │   ├── AnimatedSignature
│           │           │   │   │   └── OrbitingIcons
│           │           │   │   └── SkillsBannerGlass
│           │           │   │
│           │           │   └── AcademicJourney Section
│           │           │       ├── Timeline (Desktop)
│           │           │       │   ├── Timeline Dots
│           │           │       │   ├── Timeline Spine
│           │           │       │   └── Timeline Cards (5)
```

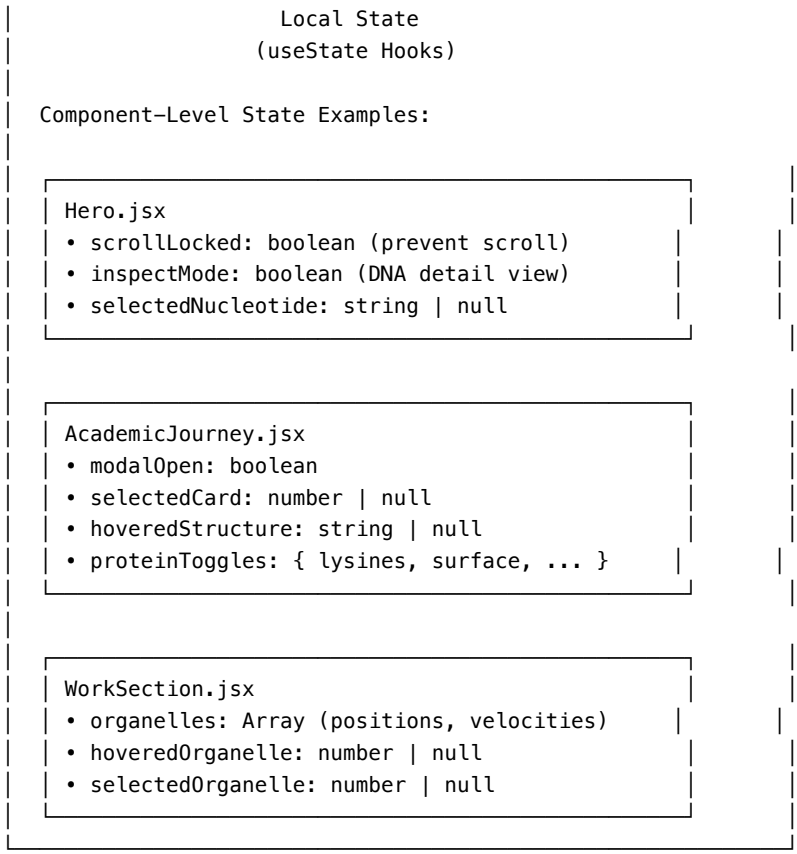
```
│           │           │       │       └── ProteinViewer (Three.js)
│           │           │       │
│           │           │       └── Modal (ReactDOM.createPortal)
│           │           │           ├── ProteinViewer (fullscreen)
│           │           │           └── ProteinControls
│           │           │
│           │           ├── Work Section
│           │           │   ├── WorkSection Container
│           │           │   ├── OrganicBlob (membrane)
│           │           │   ├── Nucleus (center)
│           │           │   └── OrganelleNew (4 projects)
│           │           │       └── Physics Engine
│           │           │
│           │           ├── Blog Section
│           │           │   ├── BlogSection Container
│           │           │   ├── BlogItem (cards)
│           │           │   └── PlaceholderCard (coming soon)
│           │           │
│           │           ├── Contact Section
│           │           │   └── Contact Form
│           │           │
│           │           └── Footer
│           │
│           ├── Route: /blog/:slug
│           │   └── Layout
│           │       └── BlogPostPage
│           │
│           ├── Route: /projects/hearing-loss-diabetes
│           │   └── Layout
│           │       └── Suspense
│           │           └── HearingLossReport (lazy)
│           │               ├── HeroSection
│           │               ├── MethodsSection
│           │               ├── DemographicsSection
│           │               ├── RiskFactorsSection
│           │               ├── ForestPlotSection
│           │               ├── ComparisonSection
│           │               └── ModelStatisticsSection
│           │
│           └── Route: /projects/melanoma-workshop
│               └── Layout
│                   └── Suspense
│                       └── MelanomaWorkshopReport (lazy)
│                           ├── HeroSection
│                           ├── StudyDesignSection
│                           ├── KnowledgeChangeSection
```

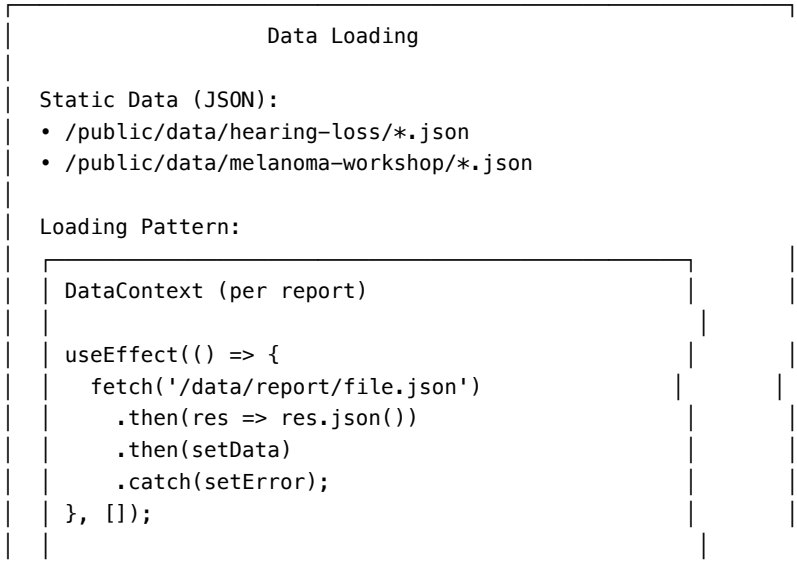
```
├── BaselineAttitudesSection
├── SkillsSection
├── SatisfactionSection
├── BarriersSection
├── StatisticalTestingSection
├── RecommendationsSection
└── ConclusionsSection
```

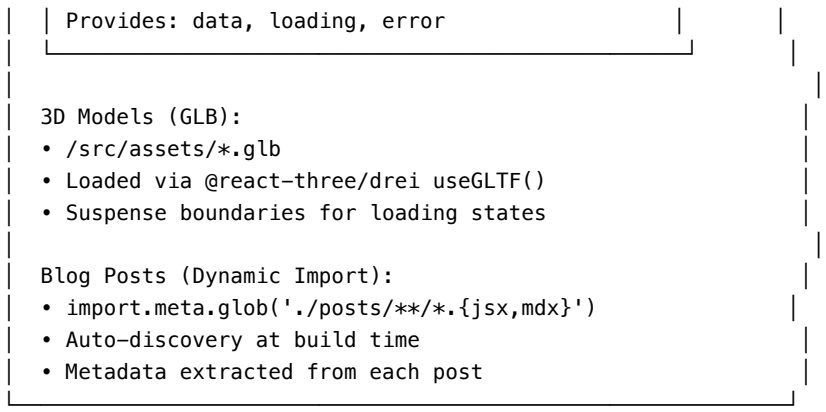
Data Flow

State Management Architecture

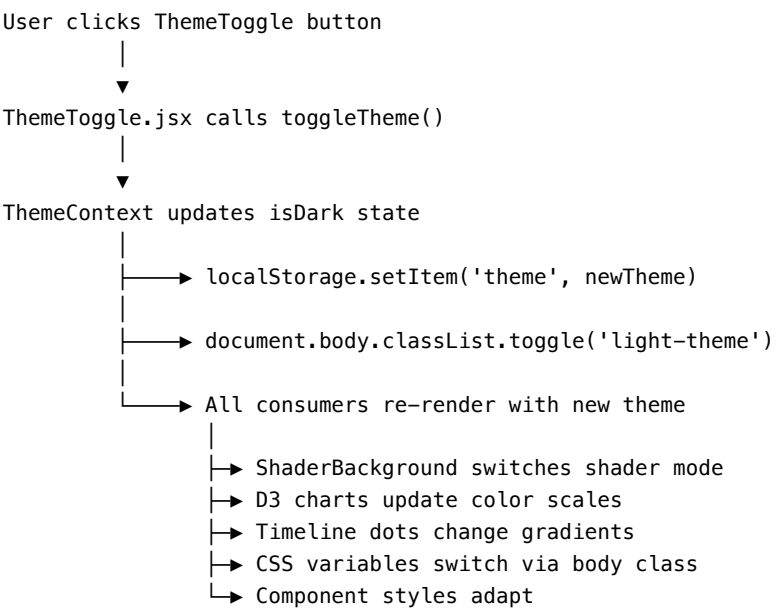



Local State
(useState Hooks)
Component-Level State Examples:
Hero.jsx
• scrollLocked: boolean (prevent scroll)
• inspectMode: boolean (DNA detail view)
• selectedNucleotide: string | null
AcademicJourney.jsx
• modalOpen: boolean
• selectedCard: number | null
• hoveredStructure: string | null
• proteinToggles: { lysines, surface, ... }
WorkSection.jsx
• organelles: Array (positions, velocities)
• hoveredOrganelle: number | null
• selectedOrganelle: number | null


Data Loading
Static Data (JSON):
• /public/data/hearing-loss/*.json
• /public/data/melanoma-workshop/*.json
Loading Pattern:
DataContext (per report)
useEffect(() => {
fetch('/data/report/file.json')
.then(res => res.json())
.then(setData)
.catch(setError);
}, []);



Data Flow Example: Theme Toggle



Application Entry Point

main.jsx Flow

```
// /src/main.jsx
```

```jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './styles/index.css'  // Global styles

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

Purpose: 1. **Mount Point:** Attaches React app to `<div id="root">` in index.html 2. **StrictMode:** Enables additional development checks 3. **Global CSS:** Imports base styles (reset, typography, CSS variables)

Load Order: 1. Browser loads `index.html` 2. Vite injects `<script type="module" src="/src/main.jsx">` 3. `main.jsx` imports and renders `App.jsx` 4. React hydrates the DOM

Layout System

Layout.jsx Structure

```jsx
// /src/Layout.jsx

export default function Layout({ children }) {
  const location = useLocation();

  // UI reveal logic based on route
  useEffect(() => {
    const onHome = location.pathname === "/";
    if (!onHome) {
      document.body.classList.add("reveal-ui");
      document.body.classList.add("show-nav");
    } else {
      document.body.classList.remove("reveal-ui");
      document.body.classList.remove("show-nav");
    }
  }, [location.pathname]);

  return (
    <>
      <ShaderBackground />  {/* Fixed background */}
```

```
      <Navbar />            {/* Sticky header */}
      {children}            {/* Route content */}
    </>
  );
}
```

Layout Responsibilities:

1. **ShaderBackground** (z-index: 0)
   - Fixed position behind all content
   - Three.js shader plane
   - Theme-aware color modes
   - Mobile camera adjustments
2. **Navbar** (z-index: 50)
   - Sticky header navigation
   - Logo and nav links
   - ThemeToggle button
   - Conditional visibility based on route
3. **Children** (z-index: 10)
   - Route-specific content
   - HomePage sections
   - Individual pages (About, Work, etc.)
   - Report pages (lazy loaded)

UI Reveal Logic:

- **Home page (`/`):**
  - Navbar hidden initially (revealed on scroll via Hero.jsx logic)
  - Full portfolio experience
- **Other pages:**
  - Navbar immediately visible
  - `reveal-ui` and `show-nav` classes added to body
  - Skip Hero animations

Section Architecture

HomePage Section Structure

```jsx
// /src/App.jsx - HomePage Component

function HomePage() {
  // Desktop page height adjustment
  useEffect(() => {
    const setCorrectPageHeight = () => {
      if (window.innerWidth <= 768) return; // Skip on mobile

      const contact = document.querySelector('.contact-wrapper-minimal');
      if (contact) {
        const contactBottom = contactRect.bottom + window.scrollY;
        document.body.style.height = `${contactBottom}px`;
      }
    };

    // Run multiple times to catch layout changes
    setTimeout(setCorrectPageHeight, 100);
    setTimeout(setCorrectPageHeight, 500);
    setTimeout(setCorrectPageHeight, 1000);
  }, []);

  return (
    <>
      <section id="hero" className="screen-section section--flush">
        <Hero />
      </section>

      <section id="about" className="screen-section section--about-merged">
        <AboutIntroMerged />
      </section>

      <section id="journey" className="screen-section section--auto-height">
        <AcademicJourney />
      </section>

      <Work />     {/* Has its own section wrapper */}
      <Blog />     {/* Has its own section wrapper */}
      <Contact />  {/* Has its own section wrapper */}
      <Footer />
    </>
  );
}
```

Section CSS Classes

`.screen-section` (Base class for all sections)

```css
.screen-section {
  min-height: 100vh;
  scroll-snap-align: start;
  display: flex;
  align-items: center;
  justify-content: center;
  box-sizing: border-box;
  position: relative;
  color: white;
}
```

`.section--flush` (Hero - no padding)

`.section--about-merged` (About - flex column layout)

```css
.section--about-merged {
  display: flex !important;
  flex-direction: column !important;
  align-items: center !important;
  justify-content: space-between !important;
  overflow: visible !important; /* Fixed: Jan 13, 2026 */
}
```

`.section--auto-height` (Academic Journey - natural height)

```css
.section--auto-height {
  min-height: auto !important;
  height: auto !important;
  display: block !important;
}
```

Code Splitting & Lazy Loading

Lazy Loading Strategy

```jsx
// /src/App.jsx

// Eager Loading (Always in initial bundle)
```

```
import Hero from "./components/hero/Hero";
import AboutIntroMerged from "./components/about/AboutIntroMerged";
import AcademicJourney from "./components/about/AcademicJourney";
import Work from "./components/work/WorkSection";
import Blog from "./components/blog/BlogSection";
import Contact from "./components/Contact";
import Footer from "./components/Footer";

// Lazy Loading (Separate chunks, loaded on demand)
const HearingLossReport = lazy(() => import("./components/projects/HearingLoss"));
const MelanomaWorkshopReport = lazy(() => import("./components/projects/MelanomaWorkshop"));
```

Why These Are Lazy Loaded

Scientific Reports: - **Large bundles:** Include D3.js charts, extensive data - **Not on critical path:** Users may never visit these pages - **Separate navigation:** Accessed via project links, not scroll

Bundle Impact: - Initial bundle: ~500KB (homepage sections) - HearingLoss report: ~200KB (lazy loaded) - MelanomaWorkshop report: ~180KB (lazy loaded) - **Total savings:** ~380KB not loaded upfront

Suspense Boundaries

```
<Route path="/projects/hearing-loss-diabetes" element={
  <Layout>
    <Suspense fallback={
      <div style={{
        display: 'flex',
        alignItems: 'center',
        justifyContent: 'center',
        minHeight: '50vh',
        color: 'var(--text-color)',
        fontSize: '1.125rem'
      }}>
        Loading Hearing Loss Report...
      </div>
    }>
      <HearingLossReport />
    </Suspense>
  </Layout>
} />
```

Fallback UI: - Centered loading message - Uses CSS variable for theme-aware text color - Minimum height to prevent layout shift - Visible for ~500ms-2s

depending on connection

Build System

Vite Configuration

```
// /vite.config.js

import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig({
  plugins: [react()],
  base: '/',
  build: {
    outDir: 'dist',
    sourcemap: false, // Disabled for production
    rollupOptions: {
      output: {
        manualChunks: {
          // Vendor chunk splitting
          vendor: ['react', 'react-dom', 'react-router-dom'],
          three: ['three', '@react-three/fiber', '@react-three/drei'],
          d3: ['d3'],
          animation: ['framer-motion', 'gsap'],
        }
      }
    }
  }
})
```

Build Output

```
dist/
├── index.html                    # Entry HTML
├── assets/
│   ├── index-[hash].js           # Main bundle (~500KB)
│   ├── vendor-[hash].js          # React, Router (~150KB)
│   ├── three-[hash].js           # Three.js (~250KB)
│   ├── d3-[hash].js              # D3.js (~100KB)
│   ├── animation-[hash].js       # Framer Motion, GSAP (~80KB)
│   ├── HearingLoss-[hash].js     # Lazy chunk (~200KB)
```

```
│   ├── MelanomaWorkshop-[hash].js # Lazy chunk (~180KB)
│   ├── index-[hash].css          # Bundled CSS (~50KB)
│   └── *.glb                     # 3D models (compressed)
└── data/                          # JSON data files
```

Build Process

1. **Development:** `npm run dev`
   - Vite dev server on port 5173
   - HMR (Hot Module Replacement)
   - Source maps enabled
   - Fast refresh
2. **Production:** `npm run build`
   - Tree shaking (removes unused code)
   - Minification (Terser for JS, cssnano for CSS)
   - Code splitting (vendor, three, d3, animation, lazy routes)
   - Asset optimization (images, fonts)
   - Hash filenames for cache busting
3. **Preview:** `npm run preview`
   - Test production build locally
   - Simulates Vercel environment

Performance Considerations

Initial Load Strategy

Critical Path (Homepage): 1. Load HTML (index.html) 2. Load vendor chunk (React, Router) 3. Load main bundle (HomePage components) 4. Load Three.js chunk (DNA helix, shader) 5. Render Hero section (above the fold)

Deferred Loading: - Academic Journey protein models (Suspense) - Blog post images (lazy loading) - Scientific reports (route-based code splitting) - Non-critical fonts (font-display: swap)

Optimization Techniques

1. **Code Splitting**
   - Route-based (reports)
   - Vendor chunking (React, Three.js, D3.js separate)

2. **Asset Optimization**
    - GLB models compressed (~20KB each)
    - Images served from Vercel CDN
    - CSS purged of unused styles (planned)
3. **Runtime Optimization**
    - React.memo for expensive components (ProteinViewer)
    - useMemo for heavy calculations (physics, chart data)
    - useCallback for event handlers
    - Throttled scroll handlers
4. **Rendering Optimization**
    - Three.js frameloop: "demand" (render only when needed)
    - CSS transforms for animations (GPU accelerated)
    - will-change hints on animated elements

Mobile Architecture Adaptations

Responsive Detection

```
// Pattern used throughout the codebase
const [isMobile, setIsMobile] = useState(false);

useEffect(() => {
  const checkMobile = () => {
    setIsMobile(window.innerWidth <= 768);
  };
  checkMobile();
  window.addEventListener('resize', checkMobile);
  return () => window.removeEventListener('resize', checkMobile);
}, []);
```

Mobile-Specific Adjustments

BackgroundScene.jsx:

```
<Canvas camera={{
  position: [0, 0, isMobile ? 10 : 15],  // Zoom in on mobile
  fov: isMobile ? 60 : 50                 // Wider FOV
}} />
```

AcademicJourney.jsx: - Desktop: Sticky timeline, horizontal cards - Mobile: Vertical stack, year badges on cards

WorkSection.jsx:

```
const ORGANELLE_SIZE = isMobile ? 70 : 100;  // Smaller on mobile
const PHYSICS_FRICTION = isMobile ? 0.95 : 0.98; // More damping
```

Architecture Decision Records

Why React Context for Theme?

Decision: Use React Context API instead of Redux or Zustand

Rationale: - Single global state (dark/light mode) - No complex state updates - Lightweight (~50 lines of code) - Built-in to React - Easy to understand for contributors

Why Lazy Loading Reports Only?

Decision: Eagerly load homepage sections, lazy load reports

Rationale: - Homepage is the primary experience - Sections depend on each other (scroll-based) - Reports are separate navigation paths - Significant bundle size reduction (380KB saved)

Why React Three Fiber?

Decision: Use @react-three/fiber instead of vanilla Three.js

Rationale: - Declarative API matches React patterns - Automatic cleanup (no manual dispose()) - React hooks for Three.js (useFrame, useThree) - Component-based architecture - Better integration with React lifecycle

---

Related Documentation

- ROUTING.md - Detailed routing configuration
- STATE-MANAGEMENT.md - State management patterns
- DESIGN_SYSTEM_ANALYSIS.md - Design tokens and styling
- THREE-JS-COMPONENTS.md *(coming soon)* - 3D component details

- PERFORMANCE.md *(coming soon)* - Performance optimization guide

---

This architecture supports a highly interactive, visually rich portfolio while maintaining good performance and developer experience.