Routing System

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

Table of Contents

Router Configuration

Core Setup

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

import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

function App() {
  const basename = (import.meta.env.VITE_BASE_URL || "/").replace(/\/$/, "");

  return (
    <ThemeProvider>
      <Router basename={basename}>
        <Routes>
          {/* Routes defined here */}
        </Routes>
      </Router>
    </ThemeProvider>
  );
}
```

Key Components: - **BrowserRouter:** HTML5 History API-based routing (clean URLs) - **basename:** Support for sub-path deployments (currently `/`)

- **Routes:** Container for all route definitions - **Route:** Individual route-to-component mapping

Why BrowserRouter? - Clean URLs without hash (`jadlopez.dev/about` vs `jadlopez.dev/#/about`) - Better SEO (search engines parse paths correctly) - Natural URL structure for users - Requires server-side configuration (Vercel handles via `vercel.json`)

Route Structure

Complete Route Tree

```
<Routes>
  {/* Homepage - Main Portfolio */}
  <Route path="/" element={
    <Layout>
      <HomePage />
    </Layout>
  } />

  {/* Individual Sections (Full Page) */}
  <Route path="/about" element={
    <Layout>
      <About mode="full" />
    </Layout>
  } />

  <Route path="/work" element={
    <Layout>
      <Work mode="full" />
    </Layout>
  } />

  <Route path="/blog" element={
    <Layout>
      <Blog mode="full" />
    </Layout>
  } />

  <Route path="/contact" element={
    <Layout>
      <Contact mode="full" />
    </Layout>
```

```
  } />

  {/* Blog Post Detail */}
  <Route path="/blog/:slug" element={
    <Layout>
      <BlogPostPage />
    </Layout>
  } />

  {/* Scientific Reports (Lazy Loaded) */}
  <Route path="/projects/hearing-loss-diabetes" element={
    <Layout>
      <Suspense fallback={<LoadingFallback text="Loading Hearing Loss Report..." />}>
        <HearingLossReport />
      </Suspense>
    </Layout>
  } />

  <Route path="/projects/melanoma-workshop" element={
    <Layout>
      <Suspense fallback={<LoadingFallback text="Loading Melanoma Workshop Report..." />}>
        <MelanomaWorkshopReport />
      </Suspense>
    </Layout>
  } />
</Routes>
```

Route Categories

1. Homepage Route Path: `/` **Component:** `HomePage` **Purpose:** Main portfolio experience with all sections in scroll flow **Layout:** Includes Shader-Background, Navbar (hidden initially)

Sections Rendered:

```
<HomePage>
  <section id="hero">Hero</section>
  <section id="about">AboutIntroMerged</section>
  <section id="journey">AcademicJourney</section>
  <Work />
  <Blog />
  <Contact />
  <Footer />
</HomePage>
```

2. Section Routes (Full Page) **Paths:** `/about`, `/work`, `/blog`, `/contact` **Purpose:** Direct links to individual sections (full page, not scroll-based) **Components:** Pass `mode="full"` prop to sections

Use Cases: - Direct navigation from external links - Skip-to-content accessibility - Section-focused URLs for sharing

Note: Currently defined but **not actively used** in navigation. Primary UX is scroll-based homepage. Could be enabled for direct linking in the future.

3. Blog Post Route **Path:** `/blog/:slug` **Component:** `BlogPostPage` **Dynamic:** Yes (`:slug` parameter)

Example URLs: - `/blog/ai-biotech-genetics` - `/blog/code-to-kinase` - `/blog/masters-origin` - `/blog/scroll-systems`

4. Project Report Routes **Paths:** `/projects/hearing-loss-diabetes`, `/projects/melanoma-workshop` **Components:** Lazy loaded report components **Suspense:** Yes (with loading fallback)

Lazy Loading

Implementation

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

import { lazy, Suspense } from "react";

// Lazy imports - separate bundles
const HearingLossReport = lazy(() =>
  import("./components/projects/HearingLoss")
);

const MelanomaWorkshopReport = lazy(() =>
  import("./components/projects/MelanomaWorkshop")
);

// Usage in route
<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>
} />
```

Why Lazy Load These Routes?

Bundle Size Impact: - **Homepage:** ~500KB (critical path) - **HearingLoss:** ~200KB (separate chunk) - **MelanomaWorkshop:** ~180KB (separate chunk) - **Total Savings:** 380KB not loaded upfront

User Behavior: - Not all users visit project reports - Reports are separate navigation paths (not scroll-based) - Heavy data visualizations (D3.js charts)

Loading Experience: 1. User clicks project link 2. Route changes immediately 3. Loading message shows (~500ms-2s) 4. Report chunk downloads 5. Report renders

Network Waterfall:

```
Initial Load:
├── index.html
├── vendor.js (React, Router)
├── main.js (Homepage)
└── three.js (DNA helix)

On Report Navigation:
└── HearingLoss-[hash].js (lazy loaded)
    ├── D3.js chunk
    └── Report data (JSON)
```

Layout Wrapper

Layout Component Role

Every route is wrapped in `<Layout>` which provides:

1. **ShaderBackground** - Fixed 3D background
2. **Navbar** - Sticky navigation
3. **Route Content** - Children prop
4. **UI Reveal Logic** - Based on current path

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

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

  useEffect(() => {
    const onHome = location.pathname === "/";

    if (!onHome) {
      // Show navbar immediately on non-homepage routes
      document.body.classList.add("reveal-ui");
      document.body.classList.add("show-nav");
    } else {
      // Hide navbar initially on homepage (Hero reveals it)
      document.body.classList.remove("reveal-ui");
      document.body.classList.remove("show-nav");
    }
  }, [location.pathname]);

  return (
    <>
      <ShaderBackground />
      <Navbar />
      {children}
    </>
  );
}
```

Layout Behavior by Route

Route	Navbar Visibility	ShaderBackground	Scroll Behavior
`/` (Homepage)	Hidden → Revealed on scroll	Dark topographic	Smooth scroll between sections
`/about`	Immediately visible	Theme-aware	Standard scroll
`/blog/:slug`	Immediately visible	Theme-aware	Standard scroll
`/projects/:`	Immediately visible	Theme-aware	Standard scroll

Route Parameters

Dynamic Parameters

Blog Slug Parameter

```
<Route path="/blog/:slug" element={
  <Layout>
    <BlogPostPage />
  </Layout>
} />
```

Accessing the Parameter:

```
// /src/components/blog/BlogPostPage.jsx

import { useParams } from 'react-router-dom';

export default function BlogPostPage() {
  const { slug } = useParams();  // "ai-biotech-genetics"

  // Use slug to load correct post
  const post = posts.find(p => p.meta.slug === slug);

  return <div>{post.content}</div>;
}
```

Parameter Validation: - If slug doesn't match any post → Show 404 or redirect - Slug must be lowercase with hyphens (enforced in post metadata)

Example Flow:

```
User clicks: "AI Meets Genetics" blog card
         ↓
Navigate to: /blog/ai-biotech-genetics
         ↓
useParams() extracts: { slug: "ai-biotech-genetics" }
         ↓
Load post with matching slug
         ↓
Render BlogPostPage with post content
```

Query Parameters

Not currently used, but could implement for: - Filtering: `/blog?category=data-science` - Search: `/projects?q=hearing` - Pagination: `/blog?page=2`

Implementation Example:

```
import { useSearchParams } from 'react-router-dom';

function BlogSection() {
  const [searchParams, setSearchParams] = useSearchParams();
  const category = searchParams.get('category');  // "data-science"

  // Filter posts by category
  const filteredPosts = posts.filter(p => p.category === category);
}
```

Navigation Patterns

Programmatic Navigation

Using `useNavigate` Hook:

```
import { useNavigate } from 'react-router-dom';

function OrganelleNew({ project }) {
  const navigate = useNavigate();

  const handleClick = () => {
    navigate(`/projects/${project.slug}`);
  };

  return <div onClick={handleClick}>{project.title}</div>;
}
```

Example Usage in Codebase: - Blog cards → Navigate to `/blog/:slug` - Organelle clicks → Navigate to `/projects/*` - CTA buttons → Navigate to `/contact`

Link Components

Using `<Link>` Component:

```
import { Link } from 'react-router-dom';

<Link to="/about">About Me</Link>
<Link to="/blog/ai-biotech-genetics">Read Post</Link>
<Link to="/projects/hearing-loss-diabetes">View Report</Link>
```

Navbar Implementation:

```jsx
// /src/components/Navbar.jsx

<nav>
  <Link to="/" className="nav-logo">JDL</Link>
  <Link to="/about">About</Link>
  <Link to="/work">Work</Link>
  <Link to="/blog">Blog</Link>
  <Link to="/contact">Contact</Link>
</nav>
```

Scroll-Based Navigation (Homepage)

Anchor Links for Sections:

```jsx
// Smooth scroll to section
const scrollToSection = (sectionId) => {
  document.getElementById(sectionId)?.scrollIntoView({
    behavior: 'smooth'
  });
};

<button onClick={() => scrollToSection('about')}>
  Learn More
</button>
```

Homepage Section IDs: - `#hero` - `#about` - `#journey` - `#work` - `#blog` - `#contact`

URL Structure

URL Schema

```
jadlopez.dev/
├── /                                   # Homepage (all sections)
├── /about                              # About section (full page)
├── /work                               # Work section (full page)
├── /blog                               # Blog section (full page)
├── /blog/:slug                         # Individual blog post
│   ├── /blog/ai-biotech-genetics
│   ├── /blog/code-to-kinase
```

```
│   ├── /blog/masters-origin
│   └── /blog/scroll-systems
├── /contact                              # Contact section (full page)
└── /projects/:project-slug               # Scientific reports
    ├── /projects/hearing-loss-diabetes
    └── /projects/melanoma-workshop
```

URL Best Practices Followed

Lowercase: All URLs use lowercase letters **Hyphens:** Word separators use hyphens (not underscores) **Semantic:** URLs describe content (`/blog/ai-biotech-genetics`) **Hierarchical:** Related content grouped (`/projects/*`) **Short:** Concise but descriptive slugs **No File Extensions:** Clean paths (no `.html`, `.php`)

SEO Considerations

Clean URLs: - Search engines prefer hierarchical, semantic URLs - `/blog/ai-biotech-genetics` is better than `/post?id=123`

Route Titles (via React Helmet - potential future enhancement):

```
import { Helmet } from 'react-helmet';

function BlogPostPage({ post }) {
  return (
    <>
      <Helmet>
        <title>{post.title} | Jesús D. López</title>
        <meta name="description" content={post.excerpt} />
      </Helmet>
      {/* Post content */}
    </>
  );
}
```

Server Configuration

Vercel Configuration

```
// /vercel.json
```

```
{
  "rewrites": [
    { "source": "/(.*)", "destination": "/index.html" }
  ]
}
```

Purpose: - **SPA Routing:** All paths serve `index.html` - **Client-Side Routing:** React Router handles routing in browser - **Deep Links:** Direct URL access works (e.g., sharing `/blog/ai-biotech-genetics`)

How It Works:

```
User visits: jadlopez.dev/projects/hearing-loss-diabetes
         ↓
Vercel server receives request
         ↓
Rewrite rule matches: /(.*) → /index.html
         ↓
Serve: index.html (with embedded app)
         ↓
React loads → Router parses path → Matches route → Lazy loads report
```

Without this configuration: - Direct URL access → 404 error - Only homepage would work - Refresh on any route → broken

---

Route Guards & Protection

Password Protection (Optional)

```
// /src/App.jsx

const isPasswordProtected = import.meta.env.VITE_ACCESS_PASSWORD;

if (isPasswordProtected) {
  return <PasswordGate>{appContent}</PasswordGate>;
}

return appContent;
```

Environment Variable: - `VITE_ACCESS_PASSWORD` - Set in Vercel environment variables - If set → Wrap entire app in `<PasswordGate>` - If not set → Public access

PasswordGate Component: - Prompts for password before showing app - Stores authentication in sessionStorage - Validates on every page load

Use Cases: - Pre-launch testing - Private portfolio version for specific clients - Draft content protection

---

Future Routing Enhancements

Potential Additions

1. **404 Page**

   ```
   <Route path="*" element={<NotFound />} />
   ```

2. **Nested Routes for Projects**

   ```
   <Route path="/projects" element={<ProjectsLayout />}>
     <Route index element={<ProjectsList />} />
     <Route path="hearing-loss-diabetes" element={<HearingLossReport />} />
     <Route path="melanoma-workshop" element={<MelanomaWorkshopReport />} />
   </Route>
   ```

3. **Blog Categories**

   ```
   <Route path="/blog/category/:category" element={<BlogByCategory />} />
   ```

4. **Admin Routes** (if adding CMS)

   ```
   <Route path="/admin" element={<AdminLayout />}>
     <Route path="posts" element={<ManagePosts />} />
     <Route path="projects" element={<ManageProjects />} />
   </Route>
   ```

5. **Redirects**

   ```
   <Route path="/old-url" element={<Navigate to="/new-url" replace />} />
   ```

Performance Considerations

Route-Based Code Splitting

Current Split: - Main bundle: Homepage sections - Lazy chunks: Reports (HearingLoss, MelanomaWorkshop)

Impact:

```
Homepage Load:
├── vendor.js (150KB)
├── main.js (500KB)
├── three.js (250KB)
└── d3.js (100KB) - if using charts
Total: ~1000KB

Report Navigation:
└── HearingLoss.js (200KB) - on-demand
```

Prefetching (Future Enhancement)

```
// Prefetch lazy routes on hover
<Link
  to="/projects/hearing-loss-diabetes"
  onMouseEnter={() => {
    import("./components/projects/HearingLoss");
  }}
>
  View Report
</Link>
```

Benefits: - Faster perceived navigation - Load chunk during hover (~500ms head start) - No extra cost if user doesn't click

---

Debugging Routes

React Router DevTools

Installation:

```
npm install @react-router/devtools
```

Usage:

```
import { RouterDevTools } from '@react-router/devtools';

<Router>
  <Routes>{/* routes */}</Routes>
  <RouterDevTools />
</Router>
```

Features: - Visual route tree - Active route highlighting - Route param inspection

Console Logging

```
import { useLocation } from 'react-router-dom';

function RouteDebugger() {
  const location = useLocation();

  useEffect(() => {
    console.log('Route changed:', location.pathname);
    console.log('Query params:', location.search);
    console.log('Hash:', location.hash);
  }, [location]);

  return null;
}
```

Related Documentation

- ARCHITECTURE.md - Overall application architecture
- STATE-MANAGEMENT.md - State management patterns
- BLOG-SECTION.md *(coming soon)* - Blog routing details
- PROJECTS-ARCHITECTURE.md - Report routing patterns

This routing system balances simplicity, performance, and user experience for a single-page portfolio with deep-linkable content.