# Performance Strategy

## Performance Targets

* **First Contentful Paint (FCP)**: < 1.5s
* **Largest Contentful Paint (LCP)**: < 2.5s
* **Time to Interactive (TTI)**: < 3.5s
* **Total Blocking Time (TBT)**: < 300ms
* **Cumulative Layout Shift (CLS)**: < 0.1

## Optimization Techniques

**1. Server Components by Default**:

// ✅ Good: Server Component (no client JS)  
export default async function PlanPage({ params }) {  
 const plan = await fetchPlan(params.id) // Server-side  
 return <PlanView plan={plan} />  
}  
  
// ❌ Bad: Client Component when not needed  
'use client'  
export default function PlanPage({ params }) {  
 const [plan, setPlan] = useState(null)  
 useEffect(() => {  
 fetchPlan(params.id).then(setPlan)  
 }, [params.id])  
 return <PlanView plan={plan} />  
}

**2. Streaming with Suspense**:

import { Suspense } from 'react'  
  
export default function DashboardPage() {  
 return (  
 <div>  
 <Suspense fallback={<ChartsSkeleton />}>  
 <DashboardCharts /> {/\* Server Component, async data fetch \*/}  
 </Suspense>  
  
 <Suspense fallback={<TableSkeleton />}>  
 <InitiativeTable />  
 </Suspense>  
 </div>  
 )  
}

**3. Database Query Optimization**:

// ✅ Good: Select only needed columns  
const { data } = await supabase  
 .from('initiatives')  
 .select('id, name, status, total\_year\_1\_cost')  
 .eq('strategic\_goal\_id', goalId)  
  
// ❌ Bad: Select all columns  
const { data } = await supabase  
 .from('initiatives')  
 .select('\*')

**4. Caching Strategy**:

// Page-level caching (Next.js)  
export const revalidate = 60 // Revalidate every 60 seconds  
  
// Route segment config  
export const dynamic = 'force-static' // Static generation  
export const fetchCache = 'force-cache'

**5. Image Optimization**:

import Image from 'next/image'  
  
<Image  
 src="/department-logo.png"  
 alt="Department Logo"  
 width={200}  
 height={100}  
 priority // For above-the-fold images  
/>

**6. Bundle Size Optimization**:

// Dynamic imports for large components  
import dynamic from 'next/dynamic'  
  
const PDFExporter = dynamic(() => import('@/components/PDFExporter'), {  
 loading: () => <LoadingSpinner />,  
 ssr: false, // Only load on client  
})