Bundle Optimization Recommendations for loomOS

Current Bundle Analysis

The following recommendations are based on the dependencies in `package.json` and common Next.js optimization patterns.

High Impact Optimizations

1. Lodash Optimization

Current Import Pattern:

```
import _ from 'lodash';
```

Optimized Pattern:

```
// ✅ Import only what you need
import debounce from 'lodash/debounce';
import merge from 'lodash/merge';
```

Impact: Reduces bundle by 50-70KB (per instance)

Action: Search for `import.*from.*'lodash'` and replace with individual imports.

---

2. Framer Motion Tree Shaking

Current Import Pattern:

```
import { motion } from 'framer-motion';
```

Optimized Pattern:

```
// ✅ Use direct imports for smaller components
import { LazyMotion, domAnimation, m } from 'framer-motion';

<LazyMotion features={domAnimation}>
  <m.div />
</LazyMotion>
```

Impact: Reduces bundle by 20-30KB

---

3. Icon Library Optimization

Lucide React is already tree-shakeable and optimized. Continue using named imports:

```
// ✅ Good - tree-shakeable
import { Home, Settings, User } from 'lucide-react';
```

---

4. Chart Library Considerations

Current:
- `chart.js` (29KB)
- `react-chartjs-2` (wrapper)
- `plotly.js-basic-dist` (500KB+)

Recommendation: Consider lighter alternatives for simple charts:
- **recharts** (40KB minified) - Better for React
- **visx** (modular, tree-shakeable)
- Keep Plotly only for complex scientific visualizations

---

5. Date Library

Current:
- `date-fns` (13KB minified) ✅ Good choice
- `dayjs` (2KB minified) ✅ Also good

Recommendation: Standardize on one library. `dayjs` is smaller but `date-fns` has better tree-shaking.

Medium Impact Optimizations

6. React Icons

Current:

```
import { FaSomething } from 'react-icons/fa';
```

Note: `react-icons` is already tree-shakeable. Continue using named imports.

---

7. Mapbox GL

Current: `mapbox-gl` (1.13.3 - 500KB+)

Optimization:

```
// Lazy load map component
const LazyMapView = dynamic(() => import('@/components/maps/map-view'), {
  ssr: false,
  loading: () => <LoadingSpinner />
});
```

Already Implemented: ✅ See `/components/lazy/index.ts`

---

8. Rich Text Editor

If using a rich text editor (not in package.json but common):

Heavy Options:
- Draft.js (100KB+)
- Slate (80KB+)
- TipTap (60KB+)

Light Options:
- react-textarea-autosize (2KB)
- Custom markdown editor with marked (30KB)

Low Impact (But Still Useful)

9. React Grid Layout

Current: `react-grid-layout` (20KB)

Optimization: Already lazy-loaded in dashboard. ✅

---

10. OpenAI SDK

Current: `openai` (large SDK)

Optimization: API calls should happen server-side only.

```
// ✅ Use in API routes only (not in client components)
// app/api/chat/route.ts
import OpenAI from 'openai';
```

Next.js Configuration Optimizations

1. Enable SWC Minifier

File: `next.config.js`

```
module.exports = {
  swcMinify: true, // ✅ Already default in Next.js 13+

  // Additional optimizations
  compiler: {
    removeConsole: process.env.NODE_ENV === 'production' ? {
      exclude: ['error', 'warn'],
    } : false,
  },
};
```

---

2. Optimize Images

```
module.exports = {
  images: {
    formats: ['image/avif', 'image/webp'],
    deviceSizes: [640, 750, 828, 1080, 1200, 1920],
    imageSizes: [16, 32, 48, 64, 96, 128, 256, 384],
    minimumCacheTTL: 60,
  },
};
```

---

3. Enable Compression

```
module.exports = {
  compress: true, // ✅ Already default

  // For even better compression, use compression middleware
  async headers() {
    return [
      {
        source: '/:all*(svg|jpg|png)',
        locale: false,
        headers: [
          {
            key: 'Cache-Control',
            value: 'public, max-age=31536000, immutable',
          }
        ],
      },
    ]
  },
};
```

Webpack Bundle Analyzer Setup

Add to package.json:

```
{
  "scripts": {
    "analyze": "ANALYZE=true next build"
  },
  "devDependencies": {
    "@next/bundle-analyzer": "^16.0.1" // ✅ Already installed
  }
}
```

Create: `next.config.js`

```
const withBundleAnalyzer = require('@next/bundle-analyzer')({
  enabled: process.env.ANALYZE === 'true',
});

module.exports = withBundleAnalyzer({
  // ... other config
});
```

Dynamic Imports Strategy

Heavy Components to Lazy Load

✅ **Already Implemented:**
- Charts (Plotly, Chart.js)
- Maps (Mapbox)
- Rich Text Editors
- AI Assistant
- Web Builder (Craft.js)

Additional Candidates:
- PDF Viewer
- Image Editor/Cropper
- CSV Parser
- Code Syntax Highlighter

Example:

```
// components/lazy/index.ts
export const LazyPDFViewer = lazyLoad(
  () => import('@/components/documents/pdf-viewer'),
  {
    loading: <LoadingSpinner />,
    ssr: false
  }
);
```

Font Optimization

Current Setup

```
// app/layout.tsx
<link href="https://fonts.googleapis.com/css2?family=Cambo&family=Titillium+Web..." />
```

Optimized Setup:

```
// Use next/font for better performance
import { Inter } from 'next/font/google';

const inter = Inter({
  subsets: ['latin'],
  display: 'swap',
  variable: '--font-inter',
});
```

Benefits:
- Self-hosted fonts (no external request)
- Automatic font subsetting
- Zero layout shift

CSS Optimization

1. PurgeCSS with Tailwind

Already Configured ✅ (Tailwind automatically purges unused CSS)

2. Critical CSS

For marketing pages, consider extracting critical CSS:

```
// next.config.js
const { withCriticalCSS } = require('next-critical');

module.exports = withCriticalCSS({
  // config
});
```

Server-Side Optimizations

1. API Route Caching

```
// app/api/data/route.ts
export const revalidate = 3600; // Cache for 1 hour

export async function GET() {
  const data = await fetchData();
  return Response.json(data);
}
```

2. Static Generation Where Possible

```
// Use generateStaticParams for dynamic routes
export async function generateStaticParams() {
  const posts = await getPosts();
  return posts.map((post) => ({
    slug: post.slug,
  }));
}
```

Monitoring and Measurement

1. Add Performance Monitoring

```
// app/layout.tsx
import { reportWebVitals } from '@/lib/web-vitals';

export function reportWebVitals(metric) {
  // Send to analytics
  console.log(metric);
}
```

2. Bundle Size Budgets

Create: `size-limit.config.js`

```
module.exports = [
  {
    name: 'Homepage',
    path: '.next/static/chunks/pages/index.js',
    limit: '100 KB',
  },
  {
    name: 'Dashboard',
    path: '.next/static/chunks/pages/dashboard.js',
    limit: '150 KB',
  },
];
```

Action Plan

Priority 1 (Immediate Impact)

- [ ] Audit and replace barrel imports from lodash
- [ ] Verify all heavy components are lazy-loaded
- [ ] Run bundle analyzer and identify largest chunks
- [ ] Implement font optimization with next/font

Priority 2 (Medium Impact)

- [ ] Optimize Framer Motion usage with LazyMotion
- [ ] Review chart library usage (Plotly vs lighter alternatives)
- [ ] Add performance monitoring
- [ ] Set up bundle size budgets

Priority 3 (Long-term)

- [ ] Consider migrating from Plotly for simple charts
- [ ] Implement incremental static regeneration for dynamic pages
- [ ] Add service worker for offline support
- [ ] Optimize database queries

Expected Results

After implementing all optimizations:

Initial Load:
- First Load JS: -30% (estimated)
- Time to Interactive: -25%
- Lighthouse Score: 90+ (from ~75)

Subsequent Loads:
- Full page transitions: -50% (with proper caching)
- Component interactions: smoother (60fps)

Useful Commands

```
# Analyze bundle
npm run analyze

# Check bundle size
npx next-bundle-analyzer

# Profile build
npm run build -- --profile

# Test production build
npm run build && npm start

# Lighthouse audit
npx lighthouse http://localhost:3000 --view

# Check for duplicate dependencies
npx depcheck
```

References

- Next.js Optimization (https://nextjs.org/docs/advanced-features/measuring-performance)
- Webpack Bundle Analyzer (https://github.com/webpack-contrib/webpack-bundle-analyzer)
- Web Vitals (https://web.dev/vitals/)
- Bundle Size Optimization (https://web.dev/reduce-javascript-payloads-with-code-splitting/)