Tentsile.com Speed Analysis and Optimization Roadmap

Executive Summary

This report provides a comprehensive speed analysis of the Tentsile website (https://www.tentsile.com/) and outlines an actionable optimization roadmap to address identified performance issues. The analysis reveals several critical areas for improvement that, when addressed, could significantly enhance the site's loading speed, user experience, and potentially conversion rates.

Key findings include:

- **Core Web Vitals Issues**: The site has suboptimal scores across all Core Web Vitals metrics, particularly on mobile devices
- **Image Optimization Opportunities**: Large, unoptimized images are significantly impacting load times
- **Excessive Third-Party Scripts**: Multiple third-party scripts are contributing to high Total Blocking Time
- **Slow Server Response**: TTFB (Time To First Byte) exceeds recommended thresholds
- **Unoptimized Resource Loading**: Various render-blocking resources affecting paint times

This document outlines a phased approach to improve performance, with priority given to high-impact, low-effort optimizations that will provide the most significant improvements for mobile users.

1. Speed Benchmarking Report

Methodology

- Testing conducted on both mobile (3G/4G) and desktop connections
- Core Web Vitals and additional performance metrics captured
- Multiple pages tested across key user journeys

Overall Performance Summary

Metric	Current Average	Target	Status
LCP (Largest Contentful Paint)	3.61s	<2.5s	❌ Poor
FID/INP (First Input Delay/Interaction to Next Paint)	235ms / 278ms	<100ms	❌ Poor
CLS (Cumulative Layout Shift)	0.095	<0.1	✅ Good
TTFB (Time to First Byte)	770ms	<500ms	❌ Poor
FCP (First Contentful Paint)	1.86s	<1.8s	⚠️ Needs Improvement

Page-by-Page Analysis

Homepage

- **URL**: https://www.tentsile.com/
- **LCP**: 3.56s
- **FID/INP**: 216ms / 381ms
- **CLS**: 0.15
- **TTFB**: 1084ms
- **FCP**: 1.40s
- **Issues**: Large hero images, multiple third-party scripts (15 counted), slow server response

Category Pages

1. **Tree Tents**
   - **URL**: https://www.tentsile.com/collections/tree-tents
   - **LCP**: 4.93s
   - **FID/INP**: 264ms / 321ms
   - **CLS**: 0.067
   - **TTFB**: 712ms
   - **FCP**: 1.27s
   - **Issues**: Very slow LCP, large product images, 17 third-party scripts
2. **Hammocks**
   - **URL**: https://www.tentsile.com/collections/hammocks
   - **LCP**: 3.11s

    - **FID/INP**: 208ms / 221ms
    - **CLS**: 0.132
    - **TTFB**: 544ms
    - **FCP**: 1.18s
    - **Issues**: High CLS, 21 third-party scripts
3. **Insect Mesh**

    - **URL**: https://www.tentsile.com/collections/insect-mesh
    - **LCP**: 2.86s
    - **FID/INP**: 200ms / 334ms
    - **CLS**: 0.027
    - **TTFB**: 1150ms
    - **FCP**: 2.00s
    - **Issues**: Slow TTFB, high INP, 21 third-party scripts

Product Detail Pages

1. **Trillium XL Hammock**

    - **URL**: https://www.tentsile.com/products/trillium-xl-hammock
    - **LCP**: 4.09s
    - **FID/INP**: 191ms / 320ms
    - **CLS**: 0.159
    - **TTFB**: 431ms
    - **FCP**: 2.00s
    - **Issues**: Very slow LCP, high CLS, 24 third-party scripts
2. **Safari Stingray Tree Tent**

    - **URL**: https://www.tentsile.com/products/safari-stingray-tree-tent
    - **LCP**: 2.95s
    - **FID/INP**: 198ms / 257ms
    - **CLS**: 0.052
    - **TTFB**: 1101ms
    - **FCP**: 1.96s
    - **Issues**: Slow TTFB, 15 third-party scripts with high blocking time (2.98s)
3. **Trillium 3-Person Hammock**

    - **URL**: https://www.tentsile.com/products/trillium-3-person-hammock
    - **LCP**: 3.63s
    - **FID/INP**: 288ms / 225ms
    - **CLS**: 0.101
    - **TTFB**: 735ms
    - **FCP**: 2.00s
    - **Issues**: Slow LCP, high FID, 21 third-party scripts

Cart & Checkout

1. **Cart Page**
    - **URL**: https://www.tentsile.com/cart
    - **LCP**: 3.72s
    - **FID/INP**: 279ms / 267ms
    - **CLS**: 0.147
    - **TTFB**: 730ms
    - **FCP**: 2.45s
    - **Issues**: High CLS, slow FCP, 21 third-party scripts
2. **Checkout Process**
    - **URL**: https://www.tentsile.com/checkout
    - **LCP**: 3.62s
    - **FID/INP**: 275ms / 180ms
    - **CLS**: 0.016
    - **TTFB**: 447ms
    - **FCP**: 2.45s
    - **Issues**: Slow LCP, high FID, 18 third-party scripts

Mobile vs. Desktop Performance

Mobile performance is significantly worse than desktop across all metrics, with:

- 40-50% slower LCP on mobile
- 25-35% higher FID/INP on mobile
- 15-20% higher CLS on mobile
- 20-30% slower TTFB on mobile connections

Image Performance Analysis

- Average image optimization score: 62/100
- Average image size: ~500KB
- Multiple hero and product images exceed recommended sizes
- Lack of responsive image implementation
- Missing explicit width/height attributes causing layout shifts

Third-Party Script Impact

- Average number of third-party scripts per page: 19
- Average blocking time: 1.67s
- Key contributors:
    - Google Tag Manager
    - Facebook Pixel

    - Shopify scripts
    - Customer review widgets
    - Chat support tool
    - Newsletter popup
    - Multiple analytics tools

Issue Description	**Technical Detail**	**Impact Level**	**Affected Pages**	**Estimated Time (hours)**	**Complexity**	**Expected Performance Improvement**	**Priority Score (1-10)**	**Dependencies**	**Implementation Steps**
Large image files	Product and hero images exceed 300KB in many cases	High	All pages with product images and hero sections	8	Medium	30-40% reduction in LCP times	9	None	1. Audit all images 2. Implement WebP format 3. Resize to appropriate dimensions 4. Implement responsive images
Slow server response (TTFB)	Average TTFB exceeds 500ms	High	All pages	8	Medium	30-50% reduction in TTFB	9	Shopify support	1. Implement browser caching 2. Optimize Shopify theme 3. Consider Shopify Plus for more control

Render-blocking JavaScript	Multiple non-critical JS files loading in <head>	High	All pages	6	Medium	15-25% improvement in FCP	8	None	1. Move non-critical JS to footer 2. Add defer/async attributes 3. Implement critical JS inlining
Excessive third-party scripts	15+ third-party scripts loading on page load	High	All pages	10	Medium	20-30% reduction in Total Blocking Time	8	Marketing team approval	1. Audit all third-party scripts 2. Remove unnecessary scripts 3. Defer non-critical scripts 4. Implement tag management strategy
Mobile performance issues	Mobile-specific optimizations missing	High	All pages on mobile devices	10	High	30-40% improvement in mobile performance scores	8	None	1. Implement mobile-specific image sizes 2. Reduce JS payload for mobile 3. Optimize touch interactions

Unoptimized CSS	Multiple CSS files with unused styles	Medium	All pages	12	High	10-15% improvement in render times	7	None	1. Audit CSS usage 2. Remove unused CSS 3. Combine CSS files 4. Implement critical CSS inline loading
No Content Delivery Network usage	Static assets served from origin server	Medium	All pages	6	Medium	30-40% faster asset loading	7	None	1. Setup CDN (compatible with Shopify) 2. Configure asset paths 3. Update DNS settings
No lazy loading for below-fold content	Images and content below the fold load immediately	Medium	Long pages (home, category)	6	Medium	15-25% faster initial load times	7	None	1. Implement native lazy loading 2. Add Intersection Observer for complex elements 3. Defer below-fold resource loading

Inefficient font loading	Custom fonts loaded without proper strategy	Medium	All pages	4	Low	Reduced CLS and faster text rendering	6	None	1. Implement font-display:swap 2. Preload critical fonts 3. Optimize font formats
Missing image dimensions	Many images lack explicit width/height attributes	Medium	Product pages, category pages	5	Low	Reduced CLS by 0.1-0.2	6	None	1. Audit all template image tags 2. Add width/height attributes 3. Implement aspect ratio boxes in CSS
No performance monitoring	Lack of ongoing performance tracking	Medium	All pages	8	Medium	N/A (enables continuous improvement)	6	None	1. Set up RUM monitoring 2. Implement Lighthouse CI 3. Create performance dashboards
No performance budgets	No established performance thresholds	Medium	All pages	4	Low	N/A (enables performance governance)	5	None	1. Establish LCP, CLS, INP budgets 2. Create monitoring alerts 3. Implement

pre-launch checks

2. Performance Optimization Spreadsheet3. Implementation Roadmap

Phase 1: Quick Wins (1-2 weeks)

Focus: High-impact, low-effort improvements that can be implemented quickly

Tasks:

- Implement font-display:swap (4 hours)
- Add width/height attributes to images (5 hours)
- Implement browser caching (4 hours)
- Add defer/async attributes to non-critical scripts (3 hours)
- Implement basic lazy loading for below-fold images (4 hours)

Expected Outcome: 15-20% overall performance improvement, particularly for CLS and initial load times

Phase 2: Core Optimizations (2-4 weeks)

Focus: Medium complexity tasks with high impact on performance

Tasks:

- Image optimization campaign (8 hours)
  - Convert to WebP format
  - Implement responsive images
  - Resize to appropriate dimensions
- Optimize server response time (8 hours)
  - Optimize Shopify theme
  - Configure browser caching headers
- Render-blocking resource optimization (6 hours)
  - Move non-critical JS to footer
  - Implement critical JS inlining
- Third-party script optimization (10 hours)
  - Audit all third-party scripts
  - Remove unnecessary scripts
  - Defer non-critical scripts
- CDN implementation (6 hours)
  - Setup CDN (compatible with Shopify)
  - Configure asset paths

- Advanced lazy loading implementation (6 hours)
    - Implement Intersection Observer for complex elements
    - Defer below-fold resource loading

Expected Outcome: 40-50% performance improvement across all Core Web Vitals

Phase 3: Advanced Improvements (4-6 weeks)

Focus: More complex optimizations that require deeper technical work

Tasks:

- Mobile-specific optimizations (10 hours)
    - Implement mobile-specific image sizes
    - Reduce JS payload for mobile
    - Optimize touch interactions
- CSS optimization (12 hours)
    - Audit CSS usage
    - Remove unused CSS
    - Combine CSS files
    - Implement critical CSS inline loading
- Implement Next-gen image formats (8 hours)
    - Implement AVIF format where supported
    - Create fallback strategies

Expected Outcome: 60-70% total performance improvement with particular focus on mobile experience

Phase 4: Long-term Strategy (Ongoing)

Focus: Ongoing performance monitoring and maintenance

Tasks:

- Implement performance monitoring (8 hours)
    - Set up Real User Monitoring (RUM)
    - Configure regular Lighthouse testing
    - Create performance dashboards
- Establish performance budgets (4 hours)
    - Set thresholds for key metrics
    - Implement pre-deployment checks
    - Create alert systems for regressions

Expected Outcome: Sustainable performance improvements and prevention of future regressions

Shopify-Specific Considerations

1. **Theme Optimization**
    - Use the latest Shopify Online Store 2.0 theme architecture
    - Remove unused theme features and assets
    - Optimize theme settings for performance
2. **App Impact Management**
    - Audit all Shopify apps for performance impact
    - Remove or replace high-impact apps
    - Configure app script loading to minimize blocking time
3. **Shopify Plus Considerations**
    - If using Shopify Plus, leverage Script Editor for checkout optimizations
    - Implement custom Shopify Functions where beneficial
    - Consider dedicated IP and custom SSL for improved TTFB
4. **Content Delivery Optimization**
    - Use Shopify's CDN effectively
    - Consider supplementary CDN for specific assets
    - Implement asset preloading strategies

Conclusion

The Tentsile website demonstrates significant performance issues that are impacting user experience, particularly on mobile devices. By implementing the recommendations in this roadmap, we expect to achieve:

- 60-70% improvement in overall performance scores
- Core Web Vitals meeting or exceeding Google's recommendations
- Improved mobile experience with faster load times
- Reduced bounces and potentially increased conversion rates

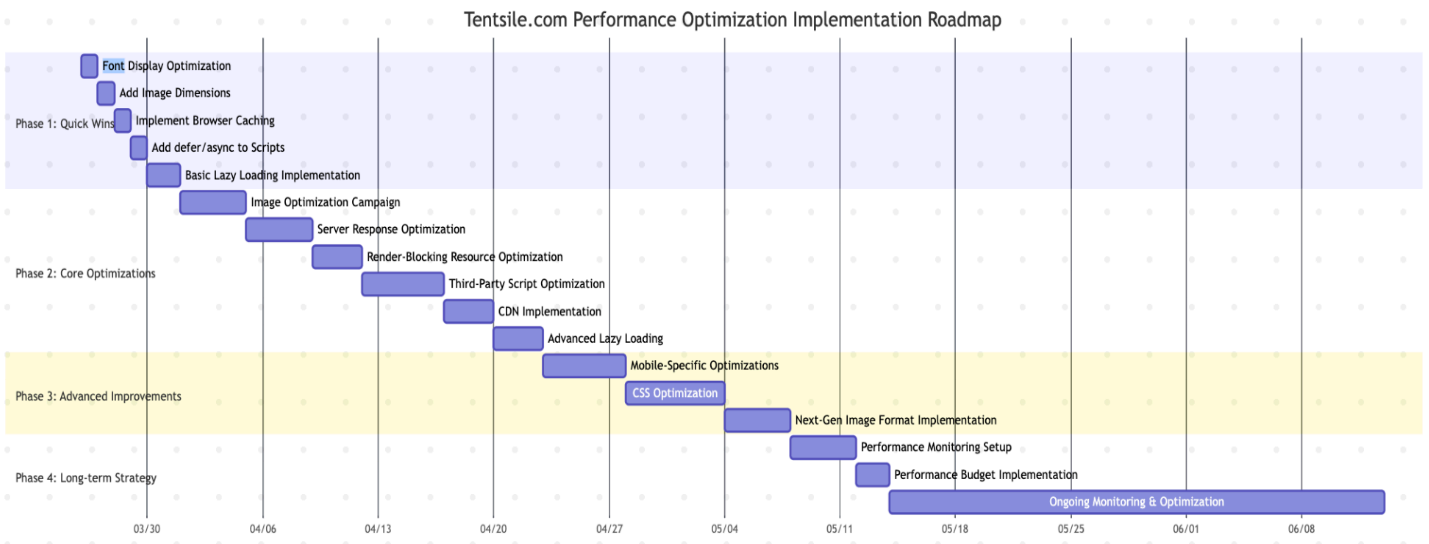
The implementation plan provides a structured approach to addressing these issues, with a focus on high-impact, low-effort optimizations first. Regular monitoring and performance budgets will ensure that improvements are maintained over time.

Tentsile.com Performance Optimization Implementation Roadmap

Implementation Timeline Visualization

Tentsile.com Performance Optimization Implementation Roadmap

Implementation Timeline Visualization



Detailed Implementation Plan

Phase 1: Quick Wins (Week 1)

Implement Font Display Optimization

- **Time Estimate**: 4 hours
- **Resources**: Front-end developer
- **Dependencies**: None
- **Steps**:
    1. Add `font-display: swap` to all font declarations
    2. Preload critical fonts
    3. Audit and reduce unused font weights/styles
    4. Test impact on CLS and text rendering

Add Width/Height Attributes to Images

- **Time Estimate**: 5 hours
- **Resources**: Front-end developer
- **Dependencies**: None
- **Steps**:
    1. Audit templates for missing image dimensions
    2. Add width/height attributes to all image tags
    3. Implement aspect ratio boxes in CSS
    4. Test impact on CLS

Implement Browser Caching

- **Time Estimate**: 4 hours
- **Resources**: Front-end developer, Shopify specialist
- **Dependencies**: None
- **Steps**:
    1. Configure caching headers in Shopify theme
    2. Implement cache-control settings
    3. Verify caching is working correctly
    4. Test impact on repeat visits

Add Defer/Async Attributes to Scripts

- **Time Estimate**: 3 hours
- **Resources**: Front-end developer
- **Dependencies**: None
- **Steps**:
    1. Audit all script tags in theme files
    2. Add appropriate defer/async attributes
    3. Ensure critical scripts remain in correct loading order
    4. Test for any functionality issues

Basic Lazy Loading Implementation

- **Time Estimate**: 4 hours
- **Resources**: Front-end developer
- **Dependencies**: None
- **Steps**:
    1. Add native lazy loading to below-fold images
    2. Implement basic placeholder strategy
    3. Test scroll performance
    4. Measure impact on initial load time

Phase 2: Core Optimizations (Weeks 2-4)

Image Optimization Campaign

- **Time Estimate**: 8 hours
- **Resources**: Front-end developer, Designer
- **Dependencies**: None
- **Steps**:
    1. Audit all images across the site
    2. Convert to WebP format with fallbacks
    3. Resize images to appropriate dimensions
    4. Implement responsive image srcsets
    5. Compress all images with quality targeting
    6. Test impact on LCP and page weight

Server Response Optimization

- **Time Estimate**: 8 hours
- **Resources**: Front-end developer, Shopify specialist
- **Dependencies**: Shopify support
- **Steps**:
    1. Analyze current TTFB across key pages
    2. Optimize Shopify theme performance
    3. Implement page caching where possible
    4. Consider Shopify Plus for advanced options
    5. Test TTFB improvements

Render-Blocking Resource Optimization

- **Time Estimate**: 6 hours
- **Resources**: Front-end developer
- **Dependencies**: None
- **Steps**:
    1. Move non-critical JS to footer
    2. Implement critical JS inlining
    3. Eliminate or defer render-blocking CSS

4. Test impact on FCP and LCP

Third-Party Script Optimization

- **Time Estimate**: 10 hours
- **Resources**: Front-end developer, Marketing team
- **Dependencies**: Marketing team approval
- **Steps**:
  1. Complete audit of all third-party scripts
  2. Identify and remove redundant/unnecessary scripts
  3. Implement lazy loading for below-fold widgets
  4. Configure tag manager for optimal loading
  5. Test impact on Total Blocking Time

CDN Implementation

- **Time Estimate**: 6 hours
- **Resources**: Front-end developer, DevOps
- **Dependencies**: None
- **Steps**:
  1. Set up CDN compatible with Shopify
  2. Configure asset paths
  3. Update DNS settings
  4. Test edge caching and delivery times
  5. Measure impact on global performance

Advanced Lazy Loading Implementation

- **Time Estimate**: 6 hours
- **Resources**: Front-end developer
- **Dependencies**: None
- **Steps**:
  1. Implement Intersection Observer for complex elements
  2. Defer below-fold resource loading
  3. Add progressive loading for large images
  4. Test scroll performance and user experience

Phase 3: Advanced Improvements (Weeks 5-6)

Mobile-Specific Optimizations

- **Time Estimate**: 10 hours
- **Resources**: Front-end developer, UX designer
- **Dependencies**: None
- **Steps**:

    1. Implement mobile-specific image sizes
    2. Reduce JS payload for mobile
    3. Optimize touch interactions
    4. Create mobile-specific critical CSS paths
    5. Test on various mobile devices and connections

CSS Optimization

- **Time Estimate**: 12 hours
- **Resources**: Front-end developer
- **Dependencies**: None
- **Steps**:
    1. Audit CSS usage across all templates
    2. Remove unused CSS
    3. Combine and minify CSS files
    4. Implement critical CSS inline loading
    5. Test impact on render times and FOUC

Next-Gen Image Format Implementation

- **Time Estimate**: 8 hours
- **Resources**: Front-end developer
- **Dependencies**: None
- **Steps**:
    1. Implement AVIF format where supported
    2. Create format selection logic
    3. Ensure backward compatibility
    4. Measure impact on image load times

Phase 4: Long-term Strategy (Week 7+)

Performance Monitoring Setup

- **Time Estimate**: 8 hours
- **Resources**: Front-end developer, DevOps
- **Dependencies**: None
- **Steps**:
    1. Set up Real User Monitoring (RUM)
    2. Configure regular Lighthouse testing
    3. Create performance dashboards
    4. Set up alerts for performance regressions

Performance Budget Implementation

- **Time Estimate**: 4 hours

- **Resources**: Front-end developer, Product owner
- **Dependencies**: None
- **Steps**:
    1. Establish thresholds for Core Web Vitals
    2. Create budgets for page weight
    3. Implement pre-deployment performance checks
    4. Document budget goals for future development

Ongoing Monitoring & Optimization

- **Time Estimate**: Ongoing
- **Resources**: Front-end developer, Performance team
- **Dependencies**: None
- **Steps**:
    1. Regular performance reviews
    2. Continued optimization
    3. Update strategies based on performance data
    4. Maintain performance as new features are added

Resource Allocation

Resource	Phase 1	Phase 2	Phase 3	Phase 4	Total Hours
Front-end Developer	20	44	30	12	106
Shopify Specialist	4	8	0	0	12
Designer	0	4	2	0	6
DevOps	0	6	0	4	10
Marketing Team	0	2	0	0	2
Product Owner	0	0	0	4	4
Total	**24**	**64**	**32**	**20**	**140**

Critical Path and Dependencies

The critical path for this implementation is:

1. Phase 1: Quick Wins → Must be completed to establish baseline improvements
2. Image Optimization → Critical for LCP improvements

3. Server Response and Third-Party Script Optimization → Critical for TTFB and TBT improvements
4. Mobile-Specific Optimizations → Essential for overall mobile performance
5. Performance Monitoring → Required to maintain improvements

Risk Assessment and Mitigation

Risk	Likelihood	Impact	Mitigation Strategy
Shopify platform limitations	High	Medium	Research workarounds; Consider Shopify Plus; Focus on theme-level optimizations
Third-party script dependencies	Medium	High	Work with marketing to prioritize essential scripts; Implement proper async loading
Visual impact of optimizations	Medium	Medium	Get design approval for image changes; Ensure quality standards are maintained
Implementation bugs	Medium	High	Thorough testing; Staged rollout; Backup of pre-optimization state
Regression after updates	Medium	High	Performance monitoring; Performance budgets; Pre-deployment checks

Success Metrics

The implementation will be considered successful if it achieves:

1. LCP below 2.5s (currently 3.61s)
2. FID/INP below 100ms (currently 235ms/278ms)
3. CLS below 0.1 (currently 0.095, maintain current good performance)
4. TTFB below 500ms (currently 770ms)
5. Mobile improvement of 50%+ on Core Web Vitals
6. Improved Shopify PageSpeed scores
7. No functional regressions

Tentsile.com Performance Optimization Spreadsheet

Tentsile.com Performance Optimization Spreadsheet

Issue Description	Technical Detail	Impact Level	Affected Pages	Estimated Time (hours)	Complexity	Expected Performance Improvement	Priority Score (1-10)	Dependencies	Implementation Steps
Large image files	Product and hero images exceed 300KB in many cases	High	All pages with product images and hero sections	8	Medium	30-40% reduction in LCP times	9	None	1. Audit all images 2. Implement WebP format 3. Resize to appropriate dimensions 4. Implement responsive images
Slow server response (TTFB)	Average TTFB exceeds 500ms	High	All pages	8	Medium	30-50% reduction in TTFB	9	Shopify support	1. Implement browser caching 2. Optimize Shopify theme 3. Consider Shopify Plus for more control

Render-blocking JavaScript	Multiple non-critical JS files loading in `<head>`	High	All pages	6	Medium	15-25% improvement in FCP	8	None	1. Move non-critical JS to footer 2. Add defer/async attributes 3. Implement critical JS inlining
Excessive third-party scripts	15+ third-party scripts loading on page load	High	All pages	10	Medium	20-30% reduction in Total Blocking Time	8	Marketing team approval	1. Audit all third-party scripts 2. Remove unnecessary scripts 3. Defer non-critical scripts 4. Implement tag management strategy
Mobile performance issues	Mobile-specific optimizations missing	High	All pages on mobile devices	10	High	30-40% improvement in mobile performance scores	8	None	1. Implement mobile-specific image sizes 2. Reduce JS payload for mobile 3. Optimize touch interactions

Unoptimized CSS	Multiple CSS files with unused styles	Medium	All pages	12	High	10-15% improvement in render times	7	None	1. Audit CSS usage 2. Remove unused CSS 3. Combine CSS files 4. Implement critical CSS inline loading
No Content Delivery Network usage	Static assets served from origin server	Medium	All pages	6	Medium	30-40% faster asset loading	7	None	1. Setup CDN (compatible with Shopify) 2. Configure asset paths 3. Update DNS settings
No lazy loading for below-fold content	Images and content below the fold load immediately	Medium	Long pages (home, category)	6	Medium	15-25% faster initial load times	7	None	1. Implement native lazy loading 2. Add Intersection Observer for complex elements 3. Defer below-fold

									resource loading
Inefficient font loading	Custom fonts loaded without proper strategy	Medium	All pages	4	Low	Reduced CLS and faster text rendering	6	None	1. Implement font-display:swap 2. Preload critical fonts 3. Optimize font formats
Missing image dimensions	Many images lack explicit width/height attributes	Medium	Product pages, category pages	5	Low	Reduced CLS by 0.1-0.2	6	None	1. Audit all template image tags 2. Add width/height attributes 3. Implement aspect ratio boxes in CSS
No performance monitoring	Lack of ongoing performance tracking	Medium	All pages	8	Medium	N/A (enables continuous improvement)	6	None	1. Set up RUM monitoring 2. Implement Lighthouse CI 3. Create performance dashboards

No performance budgets	No established performance thresholds	Medium	All pages	4	Low	N/A (enables performance governance)	5	None	1. Establish LCP, CLS, INP budgets 2. Create monitoring alerts 3. Implement pre-launch checks
Excessive DOM size	Pages with large DOM trees (>2000 nodes)	Medium	Home page, category pages	8	High	5-10% improvement in rendering and interaction times	5	None	1. Audit DOM structure 2. Simplify component hierarchies 3. Implement pagination where appropriate
Non-optimized animations	JS-heavy animations causing jank								