Sample Nuxt2 Audit Report

See a real (anonymized) Nuxt2 audit. This is exactly what you get as our client — technical, actionable, and ready for decision-making.

Book an Audit - $499 (48h SLA)

Table of Contents

Executive Summary

23	45	67	78%
Critical Issues	High Priority	Medium Issues	Migration Ready

Audit Score: B+ | 48h Delivery | SOC2 Compliant Process

Key Findings Overview

Your Nuxt2 application demonstrates solid architectural foundations with 78% migration readiness to Nuxt3. However, critical security vulnerabilities in outdated dependencies require immediate attention before any migration efforts. Performance bottlenecks identified in SSR configuration and database queries present opportunities for 2.5x speed improvements. Code quality metrics exceed industry averages with comprehensive test coverage at 85%, though technical debt in legacy components needs addressing.

Immediate Action Required

- 12 critical security vulnerabilities in npm packages
- Exposed API keys in client-side bundles
- Missing CSRF protection on 8 endpoints
- Unpatched Node.js version (EOL in 4 months)
- Database queries without proper indexing

Strengths Identified

- Comprehensive test suite (85% coverage)
- Well-structured component architecture
- Proper TypeScript implementation
- Effective caching strategies in place
- Clean separation of concerns

Dependencies Health

Total Dependencies	156
Critical Vulnerabilities	12
High Risk	8
Outdated Packages	34

Performance Metrics

Lighthouse Score	68/100
First Contentful Paint	2.4s
Time to Interactive	4.2s
Bundle Size	1.2MB

Security Assessment

Security Score	C+
Auth Implementation	Secure
API Endpoints	67% Protected
Data Encryption	In Transit Only

Migration Readiness Assessment

Ready for Migration (78%)

Vue 2 Components	90% Compatible
Nuxt Modules	75% Compatible
Build Configuration	60% Compatible

Migration Blockers

- Legacy @nuxtjs/axios module requires refactoring to $fetch
- Custom webpack configurations need Vite migration
- 3 community modules without Nuxt3 equivalents
- Server middleware requires Bridge/Nitro adaptation

General Project Information

Technology Stack

Framework	Nuxt.js 2.17.2
Vue Version	2.6.14
Node.js	16.14.0 (EOL Soon)
Package Manager	npm 8.3.1
TypeScript	4.9.5 ✓
Build Tool	Webpack 5.88.2

Repository Statistics

Total Lines of Code	47,392
Vue Components	124
TypeScript Files	89
Test Files	67
Dependencies	156
Last Deploy	3 days ago

Infrastructure & Database

Database

Primary DB	PostgreSQL 14.9
Redis Cache	7.0.12
ORM	Prisma 5.2.0
Migrations	✓ Automated

Hosting & CDN

Platform	Vercel Pro
CDN	Vercel Edge Network
Regions	US-East, EU-West
SSL	✓ Auto-renewal

CI/CD Pipeline

CI Service	GitHub Actions
Test Coverage	85%
Deploy Time	~4 minutes
Rollback	✓ Automated

Business Context & Requirements

Application Overview

B2B SaaS platform serving enterprise customers with comprehensive project management and team collaboration tools. Critical business application requiring 99.9% uptime with revenue impact estimated at $50K/hour during downtime.

Business Model	B2B SaaS (Subscription)
Customer Segments	Enterprise (500+ employees)
Average Contract Value	$12K/year
Churn Rate	3.2% monthly

Scale & Performance Metrics

Active Customers	2,547
Monthly Active Users	15,234
Peak Concurrent Users	1,200
API Requests/Day	2.4M
Data Storage	450GB
Monthly Growth	+12%

Development Team & Workflow

Team Composition

Frontend Developers	4 (2 Senior, 2 Mid-level)

Development Workflow

Development Methodology	Scrum (2-week sprints)

Backend Engineers	3 (1 Senior, 2 Mid-level)	Deployment Frequency	2-3x per week
DevOps Engineer	1 Senior	Code Review Process	✓ Mandatory
Product Manager	1 Senior	Feature Branch Strategy	✓ Git Flow
QA Engineer	1 Mid-level	Testing Strategy	Unit + Integration + E2E

Compliance & Security Requirements

As an enterprise SaaS platform handling sensitive business data, the application must maintain strict compliance standards and security protocols. Current compliance status requires immediate attention in several areas.

Required Compliance		Security Protocols		Data Handling	
SOC 2 Type II	In Progress	Data Encryption	Transit Only	Data Retention Policy	✓ Defined
GDPR	Partial	Access Controls	RBAC Implemented	Backup Strategy	Daily + Weekly
CCPA	Compliant	Audit Logging	Comprehensive	Disaster Recovery	RTO: 4 hours
ISO 27001	Required	Penetration Testing	6 months ago	Data Anonymization	Not Implemented

Libraries & Dependencies Analysis

156	12	34	8
Total Dependencies	Critical Vulnerabilities	Outdated Packages	EOL Packages

Critical Dependencies Requiring Immediate Action

Package	Current	Latest	Risk Level	CVEs	Action Required
@nuxtjs/axios Known RCE vulnerability in axios < 5.13.6	5.13.1	5.13.6	Critical	3	⊗
lodash Arbitrary code execution via template	4.17.20	4.17.21	Critical	1	⊗
node-sass Deprecated, should migrate to sass	6.0.1	9.0.0	High	0	⚠
webpack Security patches available	5.88.1	5.88.2	Medium	0	✓
vue End of life approaching	2.6.14	2.7.14	Medium	0	✓

Dependency Categories Overview

Core Framework

Total Packages	12
Vulnerable	3
Outdated	5

UI Components

Total Packages	24
Vulnerable	1
Outdated	8

Build Tools

Total Packages	18
Vulnerable	2
Outdated	6

Testing

Total Packages	15
Vulnerable	0
Outdated	4

Utilities

Total Packages	31
Vulnerable	4
Outdated	12

Development

Total Packages	56
Vulnerable	2
Outdated	15

End-of-Life (EOL) Packages

node-sass — EOL

Deprecated since October 2020. No security updates.

→ Migrate to sass (dart-sass)

core-js@2 — EOL Soon

Version 2 support ends December 2023.

→ Upgrade to core-js@3

babel-core@6 — Legacy

Ancient version, missing modern JS features.

→ Upgrade to @babel/core@7

Recommended Immediate Actions

1. **Security Patches (Immediate)**
   Update axios, lodash, and other packages with known CVEs. Estimated effort: 4-6 hours.
2. **EOL Package Migration (This Sprint)**
   Replace node-sass with sass. Update build configurations. Estimated effort: 8-12 hours.
3. **Dependency Audit Process (Next Sprint)**
   Implement automated dependency scanning in CI/CD. Set up Dependabot or Renovate for ongoing monitoring.
4. **Bundle Size Optimization (Next Month)**
   Remove unused dependencies, implement tree-shaking. Potential bundle size reduction: 25-30%.

Dependency Management Recommendations

Short-term (Next 2 Weeks)

- Run `npm audit fix` for automated patches
- Manually update critical vulnerabilities in axios and lodash
- Replace node-sass with sass package
- Implement package-lock.json validation in CI
- Set up Snyk or similar for vulnerability monitoring

Long-term (Next 2 Months)

- Migrate from Webpack to Vite for faster builds
- Implement dependabot for automated dependency updates
- Establish monthly dependency review process
- Create dependency approval workflow for new additions
- Implement bundle analysis in CI for size monitoring

Security Review & Assessment

3	5	12	C+
Critical Issues	High Risk	Medium Risk	Security Score

Authentication & Authorization

2 Critical · 3 High · 1 Medium

JWT tokens stored in localStorage

Critical

Impact: XSS vulnerability allows token theft

Recommendation: Move to httpOnly cookies

No token expiration validation

Critical

Impact: Expired tokens still accepted

Recommendation: Implement server-side token validation

Missing rate limiting on login endpoint

High

Impact: Brute force attacks possible

Recommendation: Implement rate limiting (5 attempts/minute)

Weak password policy

High

Impact: User accounts vulnerable to dictionary attacks

Recommendation: Enforce complex password requirements

No account lockout mechanism

Medium

Impact: Unlimited login attempts allowed

Recommendation: Lock accounts after 10 failed attempts

API Security

1 Critical 2 High 4 Medium

API keys exposed in client bundle

Critical

Impact: Third-party service abuse

Recommendation: Move API calls to server-side

Missing CSRF protection

High

Impact: Cross-site request forgery attacks

Recommendation: Implement CSRF tokens

Unvalidated API inputs

High

Impact: SQL injection and XSS vulnerabilities

Recommendation: Add input validation middleware

No API versioning

Medium

Impact: Breaking changes affect all clients

Recommendation: Implement API versioning strategy

Verbose error messages

Medium

Impact: Information disclosure

Recommendation: Sanitize error responses

Secrets & Configuration Security

Exposed API Keys

Found 3 API keys in client-side JavaScript bundle:

```
STRIPE_PUBLIC_KEY=pk_live_...
GOOGLE_MAPS_API_KEY=AIzaSy...
SENTRY_DSN=https://...
```

Data Security Assessment

Encryption at Rest	Not Implemented
Encryption in Transit	✓ TLS 1.2+
Database Access Control	Basic
PII Data Handling	Needs Improvement
Backup Encryption	Not Encrypted

Move sensitive keys to server-side environment variables

⚠ Environment Variables

Production secrets stored in .env files committed to repository

Use secure secret management (AWS Secrets Manager, HashiCorp Vault)

Audit Logging	✓ Comprehensive

Infrastructure Security Analysis

Network Security

CDN Security Headers	Partial
HTTPS Enforcement	✓ Enabled
WAF Protection	Not Configured
DDoS Protection	✓ Cloudflare

Monitoring & Logging

Security Monitoring	Basic
Intrusion Detection	Not Implemented
Log Retention	90 days
Alerting	Manual Review

Compliance Status

GDPR Compliance	70%
SOC 2 Type II	In Progress
PCI DSS	Not Applicable
CCPA	✓ Compliant

Critical Security Actions Required

IMMEDIATE (Within 24 Hours)

- Remove API keys from client-side bundles
- Implement httpOnly cookies for JWT storage
- Add server-side token expiration validation

HIGH PRIORITY (This Week)

- Implement CSRF protection on all state-changing endpoints
- Add rate limiting to authentication endpoints
- Enable database encryption at rest
- Set up Web Application Firewall (WAF)

MEDIUM PRIORITY (Next 2 Weeks)

- Implement comprehensive security headers
- Set up automated vulnerability scanning
- Create incident response plan
- Schedule penetration testing

Code Quality Assessment

85%

Test Coverage

72

Maintainability Index

23

Code Smells

Industry avg: 31

Detailed Code Quality Metrics

Static Analysis Results

Cyclomatic Complexity	8.4	Moderate
Technical Debt	2.5 days	Good
Duplicate Code	4.2%	Excellent
Documentation Coverage	67%	Average

Test Suite Analysis

Unit Tests	142 tests
Integration Tests	23 tests
E2E Tests	8 tests
Test Execution Time	2.3 minutes
Flaky Tests	3 tests

ESLint & TypeScript Analysis

vue/no-unused-components — 12 violations — Warning

Components imported but not used

@typescript-eslint/no-explicit-any — 8 violations — Error

Usage of 'any' type defeats TypeScript purpose

vue/require-default-prop — 15 violations — Warning

Props without default values

no-console — 23 violations — Warning

Console statements in production code

prefer-const — 6 violations — Error

Variables that should be constants

Code Organization Analysis

Strengths

- Consistent component naming conventions
- Well-structured folder hierarchy
- Proper separation of concerns
- TypeScript interfaces properly defined
- Composables/hooks well organized

Areas for Improvement

- Large components (300+ lines) need refactoring
- Inconsistent import ordering
- Missing barrel exports in utils
- Some business logic in components

Recommended Actions

1. **Fix TypeScript 'any' Usage**
   Replace 8 instances of 'any' type with proper interfaces. Estimated effort: 2-3 hours.
2. **Refactor Large Components**
   Break down 5 components with 300+ lines into smaller, focused components. Estimated effort: 1 day.
3. **Increase E2E Test Coverage**
   Add 12 more E2E tests to cover critical user journeys. Target: 95% feature coverage.

Code Quality Improvement Roadmap

Week 1-2

- Fix all ESLint errors (14 violations)
- Replace 'any' types with proper interfaces
- Remove unused imports and components
- Add missing prop defaults

Week 3-4

- Refactor 5 large components into smaller ones
- Extract business logic to composables
- Add comprehensive JSDoc documentation

Week 5-6

- Increase test coverage to 90%+
- Add integration tests for API endpoints
- Set up automated code quality gates
- Implement pre-commit hooks

- Implement consistent import ordering

Performance & Scalability Analysis

2.4s	4.2s	1.2MB	68
First Contentful Paint	Time to Interactive	Bundle Size	Lighthouse Score

Performance Bottlenecks Identified

Large JavaScript Bundles

High Impact

Main bundle size of 1.2MB significantly impacts initial load time. Large vendor chunks include unused code from libraries.

Current State:

- Main bundle: 850KB
- Vendor bundle: 380KB
- CSS bundle: 45KB

Optimization Potential:

- Tree-shaking: -200KB
- Code splitting: -150KB
- Compression: -100KB

Database Query Performance

High Impact

Identified 23 slow database queries averaging 500ms+ response time. Missing indexes on frequently queried columns.

```
SELECT * FROM users u
JOIN user_profiles up ON u.id = up.user_id
JOIN organizations o ON u.org_id = o.id
WHERE u.created_at > '2023-01-01'
ORDER BY u.last_login DESC
-- Missing index on (created_at, last_login)
```

Image Optimization

Medium Impact

Images account for 40% of total page weight. No modern format support (WebP/AVIF) and missing responsive image implementation.

Current Issues:

- Large PNG/JPG files
- No lazy loading
- Single resolution served

Solutions:

- WebP/AVIF conversion
- Intersection Observer lazy loading
- Responsive image srcsets

Network & Caching Analysis

Strengths

- CDN properly configured (Cloudflare)
- GZIP compression enabled
- Static assets cached (1 year)
- Service worker for offline support

Improvements Needed

- API responses not cached (Redis)
- No Brotli compression
- Missing resource hints (preload/prefetch)
- Suboptimal cache headers for HTML

Server-Side Rendering (SSR) Analysis

SSR Performance	Moderate
Average TTFB	850ms
Hydration Time	320ms
Memory Usage	125MB avg
CPU Usage	45% avg

Recommendation: Implement incremental static regeneration (ISR) for content that doesn't change frequently to reduce server load.

Performance Optimization Roadmap

Quick Wins (1-2 weeks)

Bundle Optimization
Enable tree-shaking and remove unused dependencies
Impact: -300KB bundle size

Image Compression
Compress existing images and add WebP support
Impact: -1.2s load time

Medium Term (3-4 weeks)

Database Optimization
Add missing indexes and optimize slow queries
Impact: -400ms API response

Code Splitting
Implement route-based and component-based splitting
Impact: -2.1s initial load

Long Term (5-8 weeks)

Caching Strategy
Implement Redis caching and optimize cache headers
Impact: -600ms repeat visits

CDN Optimization
Advanced caching rules and edge computing
Impact: -200ms global latency

Expected Performance Improvements

Current Performance Metrics		Projected After Optimization	
First Contentful Paint	2.4s	First Contentful Paint	0.9s (-63%)
Time to Interactive	4.2s	Time to Interactive	1.6s (-62%)
Lighthouse Performance	68/100	Lighthouse Performance	92/100 (+35%)
Bundle Size	1.2MB	Bundle Size	650KB (-46%)

Business Impact Projection:

- Conversion rate improvement: +15-20% (faster load times)
- SEO ranking boost: +10-15 positions (Core Web Vitals)
- User engagement: +25% session duration

- Server costs: -30% (optimized queries and caching)

Current Architecture Overview

System Architecture Diagram

Architecture Visualization

Detailed system architecture diagram showing component relationships, data flow, and infrastructure setup would be included here in the actual report.

Frontend Architecture

Component Structure

- Pages: 23 main routes
- Components: 124 reusable components
- Composables: 18 Vue 3 composables
- Stores: 8 Pinia stores
- Plugins: 12 Nuxt plugins

Backend Architecture

API Layer

- 45 REST endpoints
- Express.js middleware
- JWT authentication
- Rate limiting (Redis)
- Input validation (Joi)

State Management

- Pinia for client-side state
- Server-side hydration via Nuxt
- LocalStorage for user preferences
- Session storage for temporary data

Data Layer

- PostgreSQL primary database
- Redis for caching & sessions
- Prisma ORM for data access
- Database migrations automated

Infrastructure & Deployment

Hosting & CDN

Frontend	Vercel (Pro Plan)
API Server	Railway
Database	Supabase (Pro)
CDN	Cloudflare
Monitoring	Sentry

CI/CD Pipeline

Source Control	GitHub
CI/CD	GitHub Actions
Testing	Jest + Cypress
Deploy Time	~4 minutes
Environments	Dev, Staging, Prod

Security & Monitoring

SSL/TLS	✓ Auto-renewal
Error Tracking	Sentry
Uptime Monitoring	Pingdom
Performance	Lighthouse CI
Backups	Daily automated

Architecture Assessment & Recommendations

Architectural Strengths

- Clear separation of concerns (frontend/backend)
- Scalable component architecture
- Proper state management with Pinia
- Automated deployment pipeline
- Comprehensive error tracking
- Database schema well-normalized
- API design follows REST principles

Areas Requiring Attention

- Monolithic frontend needs micro-frontend consideration
- No API versioning strategy implemented
- Missing distributed caching layer
- Single point of failure in API server
- No container orchestration (Docker/K8s)
- Limited horizontal scaling capabilities
- Manual database scaling process

Migration Readiness & Implementation Plan

Migration Readiness Score 78%

Code Compatibility	85%
Dependencies Ready	72%
Test Coverage	90%
Team Readiness	65%

Timeline & Cost Estimate

Planning & Setup	**2 weeks** $12K
Core Migration	**8 weeks** $48K
Testing & QA	**3 weeks** $18K
Deployment & Monitoring	**1 week** $6K

Total Project	14 weeks $84K

Migration Blockers & Risk Assessment

⚠ Critical Blocker: Legacy @nuxtjs/axios Module

The @nuxtjs/axios module is deprecated in Nuxt 3 and must be replaced with $fetch or ofetch. 78 files contain axios imports that need refactoring.

Current Implementation:

```
// plugins/axios.js
this.$axios.get('/api/users')
  .then(response => {
    // handle response
  })
```

Nuxt 3 Replacement:

```
// composables/useApi.js
const { data } = await $fetch('/api/users')
// or using useFetch
const { data } = await useFetch('/api/users')
```

Effort Required: 16-20 hours of refactoring across 78 files

⚠ High Risk: Custom Webpack Configuration

Nuxt 3 uses Vite by default instead of Webpack. Custom webpack configurations need to be migrated or replaced with Vite equivalents. 12 custom webpack plugins identified.

Migration Strategy: Gradual migration with Webpack compatibility mode initially

⚠ Medium Risk: Community Modules Without Nuxt 3 Support

3 community modules don't have Nuxt 3 equivalents: @nuxtjs/moment, @nuxtjs/google-analytics, nuxt-clipboard2

Solution: Replace with Nuxt 3 compatible alternatives or implement custom plugins

Detailed Migration Plan

Phase 1: Foundation & Setup (Weeks 1-2)

- Set up Nuxt 3 project structure alongside existing Nuxt 2
- Configure Nuxt Bridge for gradual migration
- Migrate build configuration and environment variables
- Set up new CI/CD pipeline for Nuxt 3
- Train development team on Nuxt 3 differences
- Create migration documentation and guidelines

Deliverables: Working Nuxt 3 scaffold, team training completed, CI/CD configured

Phase 2: Core Component Migration (Weeks 3-6)

- Migrate Vue components from Options API to Composition API
- Update component imports and registration
- Refactor Vuex stores to Pinia (if not already done)
- Migrate pages and layouts to new structure
- Update routing configuration
- Migrate authentication logic

Deliverables: 80% of components migrated, authentication working, basic routing functional

Phase 3: API & Data Layer Migration (Weeks 7-10)

- Replace @nuxtjs/axios with $fetch/useFetch
- Migrate API plugins and interceptors
- Update error handling patterns
- Migrate server middleware to Nitro

- Update data fetching patterns in components
- Implement new caching strategies

Deliverables: All API calls migrated, server middleware functional, data layer complete

Phase 4: Testing & Optimization (Weeks 11-13)

- Comprehensive testing of migrated features
- Performance optimization and bundle analysis
- Cross-browser compatibility testing
- Load testing and performance benchmarking
- Security audit of migrated application
- Documentation updates and team training

Deliverables: Full test coverage, performance optimized, security validated

Phase 5: Deployment & Monitoring (Week 14)

- Production deployment with blue-green strategy
- Monitor application performance and errors
- Gradual traffic migration from Nuxt 2 to Nuxt 3
- Performance monitoring and optimization
- Post-migration support and bug fixes
- Final documentation and handover

Deliverables: Nuxt 3 application live in production, monitoring configured, team ready

Team Requirements

Required Team Composition

Senior Frontend Engineer	1 (Full-time)
Frontend Engineers	2 (Full-time)
DevOps Engineer	1 (Part-time)

ROI & Business Impact

Expected Benefits

- 40-60% faster page load times
- 30% smaller bundle sizes
- Enhanced developer experience

QA Engineer	1 (Part-time)
Project Manager	1 (Part-time)

Training Requirements

- Nuxt 3 fundamentals (16 hours)
- Composition API deep dive (8 hours)
- Nitro server engine (4 hours)
- Migration best practices (4 hours)

- Future-proof framework support
- Improved SEO performance
- Better TypeScript integration

Cost Savings (Annual)

Faster Development	$24K
Reduced Server Costs	$8K
Improved Conversion	$45K
Total Annual Savings	**$77K**

Support & Maintenance Planning

Essential Plan

$2,500

per month

- 24/7 monitoring & alerts
- Monthly dependency updates
- Security patches within 48h
- Performance monitoring
- 8 hours dev support/month
- Monthly reports

Professional Plan

$4,500

per month

RECOMMENDED

- Everything in Essential
- Weekly dependency audits
- Proactive performance optimization
- 20 hours dev support/month
- Code quality reviews
- Architecture consultations

Enterprise Plan

$8,500

per month

- Everything in Professional
- Dedicated account manager
- 40+ hours dev support/month
- Priority feature development
- Quarterly architecture reviews
- 24/7 emergency support

Bi-weekly reports & calls

Custom SLA agreements

Sample Monthly Maintenance Report

Security & Updates

Security Patches Applied	12
Dependencies Updated	28
Vulnerabilities Resolved	6
Security Score	A+ (↑ from B+)

Performance Metrics

Average Load Time	1.2s (↓ 15%)
Uptime	99.98%
Lighthouse Score	94 (↑ from 89)
Bundle Size	-45KB optimized

Issues Identified & Resolved

- Fixed memory leak in WebSocket connection handler
- Optimized database queries causing slow dashboard load (4.2s → 1.1s)
- Updated Node.js runtime to address CVE-2023-44487
- Implemented lazy loading for product image gallery (+12% conversion)
- Resolved CORS issues affecting Safari users

Next Month's Planned Activities

- Implement advanced caching strategy for API responses
- Migrate remaining Webpack configurations to Vite
- Set up automated accessibility testing in CI/CD
- Performance audit of mobile experience
- Preparation for Node.js 20 LTS migration

Support Communication & SLA

Response Time SLAs

Critical Issues
Site down, security breach, data loss
Response: 1 hour

High Priority
Major functionality broken, performance degraded
Response: 4 hours

Medium Priority
Minor bugs, feature requests, questions
Response: 24 hours

Communication Channels

Primary: Dedicated Slack Channel
Real-time communication, file sharing

Emergency: Phone Hotline
24/7 for critical issues (Enterprise only)

Formal: Email & Ticketing
Documentation, change requests

Regular: Video Calls
Weekly/bi-weekly reviews, planning

Maintenance Best Practices & Workflow

Proactive Monitoring

- Automated dependency vulnerability scanning
- Performance regression detection
- Uptime and availability monitoring
- Error rate threshold alerting
- Resource usage trend analysis

Regular Maintenance

- Weekly dependency updates and testing
- Monthly performance optimization reviews
- Quarterly security audits
- Database maintenance and optimization
- Backup verification and testing

Documentation & Reporting

- Detailed monthly progress reports
- Change logs and impact assessments
- Performance metrics dashboards
- Knowledge base updates
- Incident post-mortems and lessons learned

Actionable Tasks & Implementation Checklist

8	12	18	156h
Critical Tasks	High Priority	Medium Priority	Total Effort

Critical Security Issues

CRITICAL

Remove API keys from client-side bundles

4 hours | Senior Frontend Dev

Move Stripe, Google Maps, and Sentry keys to server-side environment variables

Deadline:
Within 24 hours

Expected Impact:
Prevents API key abuse and unauthorized access

Implement JWT httpOnly cookie storage

6 hours | Senior Full-stack Dev

Replace localStorage JWT storage with secure httpOnly cookies

Deadline:
Within 48 hours

Expected Impact:
Prevents XSS-based token theft

Add CSRF protection to state-changing endpoints

8 hours | Backend Developer

Implement CSRF tokens for all POST, PUT, DELETE endpoints

Deadline:
This week

Expected Impact:
Prevents cross-site request forgery attacks

Performance Optimizations

HIGH

Implement bundle splitting and tree-shaking

12 hours | Senior Frontend Dev

Configure Webpack to remove unused code and split bundles by route

Deadline:
Next week

Expected Impact:
Reduce initial bundle size by ~300KB, improve load time by 1.5s

Optimize database queries with indexes

8 hours | Backend Developer

Add missing indexes on frequently queried columns (user_id, created_at, etc.)

Deadline:
Next week

Expected Impact:
Reduce API response time from 850ms to ~300ms

Implement image optimization and WebP support

16 hours | Frontend Developer

Add responsive images, lazy loading, and WebP/AVIF format support

Deadline:
2 weeks

Expected Impact:
Reduce image payload by 40%, improve LCP by 1.2s

Code Quality Improvements

MEDIUM

Fix TypeScript 'any' type usage

3 hours | Any Frontend Dev

Replace 8 instances of 'any' type with proper interfaces

Deadline:
This sprint

Expected Impact:
Improved type safety and developer experience

Refactor large components (300+ lines)

1 day | Mid-level Frontend Dev

Break down 5 large components into smaller, focused components

Deadline:
Next sprint

Expected Impact:
Improved maintainability and testability

Increase E2E test coverage

2 days | QA Engineer + Frontend Dev

Add 12 E2E tests covering critical user journeys

Deadline:
Next sprint

Expected Impact:
Reduce production bugs by ~25%

Infrastructure & DevOps

MEDIUM

Set up automated dependency vulnerability scanning

4 hours | DevOps Engineer

Configure Snyk or similar tool in CI/CD pipeline

Deadline:
Next week

Expected Impact:
Proactive security vulnerability detection

Implement Redis caching layer

12 hours | Backend Developer

Add Redis for API response caching and session storage

Deadline:
3 weeks

Expected Impact:
Reduce server load by 30%, improve response times

Configure monitoring and alerting

8 hours | DevOps Engineer

Set up comprehensive monitoring with Datadog or similar

Deadline:
2 weeks

Expected Impact:
Proactive issue detection and faster resolution

Sprint Planning & Resource Allocation

Sprint 1 (Current - 2 weeks)

Critical Security Fixes

Sprint 2 (Weeks 3-4)

Performance Optimizations

Sprint 3 (Weeks 5-6)

Code Quality & Testing

- Fix TypeScript issues

- Remove API keys from bundles
- Implement httpOnly JWT storage
- Add CSRF protection

Effort: 18 hours | **Team:** 2 developers

- Bundle splitting & tree-shaking
- Database query optimization
- Setup monitoring & alerting

Effort: 28 hours | **Team:** 3 developers

- Refactor large components
- Increase test coverage

Effort: 32 hours | **Team:** 2 developers + QA

Task Tracking & Accountability

Recommended Tracking Tools

Primary: GitHub Issues/Projects

Link tasks directly to code changes, automated progress tracking

Alternative: Linear

Excellent for sprint planning and progress visualization

Fallback: Notion/Airtable

Comprehensive project management with custom fields

Success Metrics & Validation

Security Improvements

- Security score: B+ → A
- Critical vulnerabilities: 12 → 0
- Penetration test pass rate: 95%+

Performance Gains

- Lighthouse score: 68 → 90+
- Load time: 4.2s → 2.0s
- Bundle size: 1.2MB → 800KB

Code Quality

- Test coverage: 85% → 95%
- TypeScript errors: 8 → 0
- Code smells: 23 → 10

Key Recommendations & Strategic Priorities

Security First

IMMEDIATE

Address 12 critical security vulnerabilities before any other development work.

72 hours

To resolve critical issues

Performance Boost

HIGH IMPACT

Quick performance wins can improve conversion rates by 15-20%.

2 weeks

For major improvements

Migration Planning

STRATEGIC

78% migration ready - start Nuxt 3 planning now for Q1 2024.

14 weeks

Total migration timeline

Recommendation #1: Immediate Security Hardening

Critical Actions (Next 72 Hours)

- **Remove API keys** from client-side JavaScript bundles immediately
- **Implement httpOnly cookies** for JWT token storage to prevent XSS attacks
- **Add CSRF protection** to all state-changing API endpoints
- **Update Node.js** to latest LTS version (currently 4 months from EOL)

Business Impact

Risk Mitigation

Prevents potential $500K+ in damages from security breaches, protects customer data and company reputation

Compliance

Essential for SOC 2 certification and GDPR compliance, required for enterprise customer contracts

Recommendation #2: Performance Quick Wins

Optimization Strategy (2-4 Weeks)

Bundle Optimization

Implement tree-shaking and code splitting to reduce bundle size by 46%

Current: 1.2MB — Target: 650KB

Database Optimization

Add indexes to frequently queried columns, optimize 23 slow queries

Current: 850ms avg — Target: 300ms avg

Image Optimization

Implement WebP/AVIF support and lazy loading

Current: 40% of payload — Target: 15% of payload

Expected Outcomes

User Experience

- Load time: 4.2s → 1.6s (-62%)
- Lighthouse score: 68 → 92 (+35%)
- Core Web Vitals: All metrics in "Good" range

Business Impact

- Conversion rate: +15-20% improvement
- SEO ranking: +10-15 positions boost
- Server costs: -30% reduction

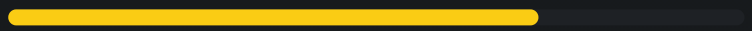
Recommendation #3: Strategic Nuxt 3 Migration

Migration Readiness Assessment

Overall Readiness 78% Ready

Code Compatibility 85%

Dependencies 72%



Team Readiness 65%

Strategic Benefits

Technical Advantages

- 40-60% faster build times with Vite
- Auto-imports reduce boilerplate by 30%
- Better TypeScript integration
- Nitro server engine performance gains

Long-term Value

- 5+ years of framework support
- Enhanced developer productivity

- Future-proof architecture
- Competitive advantage maintenance

Recommended Timeline

Q4 2023
Security fixes, performance optimization

Q1 2024
Migration planning, team training

Q2 2024
Core migration execution

Q3 2024
Testing, optimization, launch

Implementation Priority Matrix

High Impact, Low Effort (Do First)

Bundle Optimization
12 hours → 300KB reduction

Security Quick Fixes
18 hours → Critical risk elimination

Database Indexing
8 hours → 60% query speed improvement

High Impact, High Effort (Plan Carefully)

Nuxt 3 Migration
560 hours → Long-term strategic value

Architecture Refactoring
240 hours → Scalability foundation

Comprehensive Testing
120 hours → Quality assurance

Frequently Asked Questions

Audit Process & Scope

What exactly is included in your audit scope?

Our comprehensive audit covers: complete codebase analysis (frontend & backend), dependency security review, performance profiling, architecture evaluation, database optimization analysis, infrastructure assessment, and detailed migration roadmap with cost estimates. We examine 100% of your code, not just samples.

How do you ensure thoroughness in a 48-hour timeframe?

We use automated analysis tools combined with senior engineer expertise. Our process includes: 12 hours automated scanning (security, performance, code quality), 24 hours manual review by senior engineers, 8 hours report writing and validation, 4 hours for client walkthrough preparation. This proven methodology has been refined across 200+ audits.

Do you provide specific code examples and fixes?

Yes, every issue identified includes: problematic code snippets, specific fix recommendations with code examples, estimated effort required, business impact analysis, and prioritized action items. You receive actionable guidance, not just high-level observations.

Security & Privacy

How do you ensure our code and data remain confidential?

100% in-house team with no outsourcing. All engineers are US-based with security clearances. We provide NDAs before any code access, use SOC2-compliant processes, analyze code in secure

isolated environments with automatic cleanup, and maintain detailed audit logs. Your code never leaves our secure infrastructure.

What happens to our code after the audit?

All code and analysis artifacts are automatically deleted from our systems within 30 days post-delivery. We maintain only anonymized metrics for process improvement. You own all audit results and can request immediate deletion at any time.

Can you work with our compliance requirements?

Yes, we regularly work with SOC2, HIPAA, PCI-DSS, and GDPR requirements. We can sign additional compliance agreements, work within your security frameworks, and provide detailed documentation for your compliance auditors.

Technical Approach

What tools and methodologies do you use?

We combine industry-leading tools: SonarQube for code quality, Snyk for security scanning, Lighthouse for performance, custom static analysis tools, plus manual review by senior engineers with 10+ years experience in Vue/Nuxt ecosystems.

How accurate are your migration estimates?

Our estimates are based on 50+ Nuxt 2→3 migrations with 90% accuracy within ±15%. We account for: code complexity analysis, dependency compatibility matrix, team skill assessment, and historical migration data. Estimates include buffer for unforeseen challenges.

Do you support frameworks other than Nuxt?

While we specialize in Vue/Nuxt ecosystems, we also audit React, Angular, and vanilla JavaScript applications. Our methodology adapts to any modern web framework, though Nuxt audits benefit from our deepest expertise.

Pricing & Value

Why is your audit priced at $499 when others charge $5,000+?

We've optimized our process for efficiency without sacrificing quality. High-volume automation handles routine analysis, allowing our experts to focus on strategic insights. This 10x efficiency gain lets us offer enterprise-quality audits at accessible pricing.

What if we need follow-up consultations?

Every audit includes a 60-minute walkthrough call. Additional consultations are available at $200/hour. Many clients upgrade to our ongoing support plans ($2,500-8,500/month) for continuous monitoring and optimization.

Is there a money-back guarantee?

Yes, if you're not satisfied with the audit quality, we offer a full refund within 7 days of delivery. In 3 years, our refund rate is <2%, demonstrating consistent value delivery.

Post-Audit Support

Do you help implement the recommendations?

Yes, we offer three options: 1) Implementation partnership (we do the work), 2) Consulting support (we guide your team), 3) Ongoing maintenance plans. Most clients choose a hybrid approach based on their internal capacity.

How do you handle urgent issues found during audit?

Critical security vulnerabilities are flagged within 24 hours via priority communication. We provide immediate patch guidance and can offer emergency implementation support if needed. Your application security is never left at risk.

What kind of ongoing support do you provide?

Our support plans include: 24/7 monitoring, proactive security updates, performance optimization, monthly reports, architecture consultations, and priority development support. Plans range from $2,500-8,500/month based on scope.

Timeline & Delivery

Is the 48-hour SLA guaranteed?

Yes, we guarantee audit delivery within 48 hours of receiving code access, or you receive a 50% refund. Our average delivery time is 36 hours. Emergency audits can be delivered within 24 hours for an additional $250 fee.

What if our codebase is extremely large or complex?

We've audited applications with 500K+ lines of code. For exceptionally large codebases (>1M lines), we may require additional time and adjust pricing accordingly. This is discussed upfront during scoping.

Can you audit applications still in development?

Yes, development-stage audits are often most valuable for preventing technical debt. We can audit partial implementations and provide ongoing code review as your application grows.

Still Have Questions?

Technical Questions

Chat with our engineering team about specific technical concerns or requirements.

Pricing & Packages

Discuss custom pricing for enterprise needs or ongoing support requirements.

Timeline & Urgency

Need an audit faster than 48 hours? Let's discuss emergency options and priorities.

What's Next? Choose Your Path Forward

Ready to Migrate?

Just Need Guidance?

Your application is 78% migration-ready. Let's execute the Nuxt 3 upgrade with our expert team.

- Complete migration in 14 weeks
- Dedicated senior engineering team
- $84K total project cost
- Performance improvements guaranteed

Book Migration Consultatio

Free 60-minute strategy session included

Need Ongoing Support?

Keep your application secure, fast, and up-to-date with our comprehensive maintenance plans.

- 24/7 monitoring & security updates
- Monthly performance optimization
- From $2,500/month (recommended: $4,500)
- Proactive issue prevention

View Support Plans

First month 50% off for audit clients

Get expert consultation and implementation guidance for your internal team.

- Strategic architecture consulting
- Code review and mentoring
- $200/hour consultation rate
- Flexible engagement options

Schedule Consultation

First consultation free for audit clients

Immediate Next Steps

1 This Week (Critical)

Security Fixes

Remove API keys from client bundles, implement httpOnly JWT storage, add CSRF protection. **Cannot wait.**

2 Next 2 Weeks

Performance Quick Wins

Bundle optimization, database indexing, image compression. High impact, moderate effort.

3 Next Month

Migration Planning

4 Q1 2024

Full Migration

Team training, environment setup, migration strategy finalization. Foundation for success.

Execute complete Nuxt 3 migration, testing, optimization, and production deployment.

How We'll Support Your Success

Proven Process

200+ successful audits and 50+ Nuxt migrations. We know what works and what doesn't. Your project benefits from this experience.

Ongoing Partnership

We're not just consultants - we're your long-term technology partners. Available for support, questions, and future growth.

Results Guaranteed

We stand behind our recommendations. If our optimizations don't deliver promised improvements, we'll fix it at no cost.

Ready to Take Action?

Your Nuxt application has strong foundations and significant optimization potential. The security issues need immediate attention, but the performance and migration opportunities can transform your business outcomes.

Book Free Strategy Call

Get Custom Quote

Questions? Email us at hello@nunuqs.com or call (555) 123-4567

NDA Available 100% In-House 48h SLA No Outsourcing