ScanFix beta

Technical Due Diligence Report

REPORT #78

Prepared by ScanFix · AI-driven code quality & security assessment

Repository: Dennis19814/strategy-pilot

Branch: main

Generated: April 15, 2026 · 04:07 PM

Analysis Run: 4ed41b09-77ef-4561-a1e3-75e04cf2a7b5

OVERALL HEALTH

90/100

Higher is better

TOTAL FINDINGS	CRITICAL
15	1

HIGH	MEDIUM / LOW
8	6

Executive summary

Overall Health: 🟢 Excellent (90/100) This codebase analysis identified **15 total findings** across 15 dimensions: 1 critical, 8 high, 6 medium, and 0 low severity issues. ### ⚠️ Areas Requiring Immediate Attention The following areas scored below 60% and require immediate attention: **SECURITY**. ### ✅ Strong Areas The codebase excels in: **ARCHITECTURE, SCALABILITY, INFRASTRUCTURE, DEVOPS_CICD, OBSERVABILITY, COST_OPTIMIZATION, AI_SPECIFIC, DEPENDENCY_HEALTH, DATA_LAYER, API_DESIGN, MAINTAINABILITY** (scored 80% or higher). ### Key Recommendations 1. **Prioritize Critical Issues**: Address all 1 critical issues immediately. 2. **Focus on Weak Areas**: Improve scores in low-scoring dimensions. 3. **Maintain Excellence**: Continue best practices in high-scoring areas.

SCOPE

Repository: Dennis19814/strategy-pilot
Branch: main
Generated: April 15, 2026 · 04:07 PM
Report ID: #78

RISK OVERVIEW

Critical	1	High	8
Medium	6	Low	0

Category scores

Score /100 · higher is better

Security 29/100 — 8 issuesissue(s) mapped to security. 2 No high-confidence static issues were detected for security. Key issu...

Performance 78/100 — 1 issueperformance. 3 issue(s) mapped to performance. Ke... No high-confidence static issues were detected for

Code Quality 63/100 — 5 issuesissue(s) mapped to code quality. ... No high-confidence static issues were detected for code quality. 1

Architecture 100/100 — No high-confidence static issues were detected for architecture. 1 issue(s) mapped to architecture. ...

Logic Bugs 78/100 — 1 issueHIGH/CRITICAL severity. Top iss... 1 high-confidence static issue(s) were detected in logic bugs. 1 are

Scalability 100/100 — No high-confidence static issues were detected for scalability. 2 issue(s) mapped to scalability. Ke...

Infrastructure 100/100 — No high-confidence static issues were detected for infrastructure. 1 issue(s) mapped to infrastructu...

Devops Cicd 100/100 — No high-confidence static issues were detected for devops cicd. 2 issue(s) mapped to devops cicd. Ke...

Observability 100/100 — No high-confidence static issues were detected for observability. 1 issue(s) mapped to observability...

Cost Optimization 100/100 — No high-confidence static issues were detected for cost optimization. 1 issue(s) mapped to cost opti...

Ai Specific 100/100 No high-confidence static issues were detected for ai specific. Findings were consolidated and prior...

Dependency Health 100/100 No high-confidence static issues were detected for dependency health. 2 issue(s) mapped to dependenc...

Data Layer 100/100 No high-confidence static issues were detected for data layer. 2 issue(s) mapped to data layer. Key ...

Api Design 100/100 No high-confidence static issues were detected for api design. 1 issue(s) mapped to api design. Key ...

Maintainability 100/100 No high-confidence static issues were detected for maintainability. 1 issue(s) mapped to maintainabi...

Top recommended actions

1.

CRITICAL SECURITY

Critical Security Vulnerability in @esbuild Dependencies

Multiple @esbuild packages (aix-ppc64, android-arm, android-arm64, etc.) are using version 0.21.5 which has known security vulnerabilities. These are transitive dependencies of Vite/TypeScript tooling.

Est. 0.5h $13 `package-lock.json:1`

2.

HIGH SECURITY

Security Vulnerability: Supabase Auth Token Storage in localStorage

The Supabase client is configured to store authentication tokens in localStorage, which is vulnerable to XSS attacks. localStorage is accessible to any JavaScript running on the same domain, making tokens easily stealable if an XSS vulnerability exists elsewhere in the application.ts:12.ts:12.

Est. 3h $75 `src/integrations/supabase/client.ts:13`

3.

HIGH SECURITY

Unsafe TypeScript Compiler Options Disable Critical Safety Checks

The TypeScript configuration disables multiple important safety checks including 'strict', 'strictNullChecks', 'noImplicitAny', and 'noFallthroughCasesInSwitch'. This allows unsafe code patterns that can lead to runtime errors and security vulnerabilities.

Est. 2h $50 `tsconfig.app.json:18`

Detailed findings

Includes effort and cost estimates

CATEGORY

Security

#1 CRITICAL `package-lock.json:1` Est. 0.5h $13

Critical Security Vulnerability in @esbuild Dependencies

Multiple @esbuild packages (aix-ppc64, android-arm, android-arm64, etc.) are using version 0.21.5 which has known security vulnerabilities. These are transitive dependencies of Vite/TypeScript tooling.

RECOMMENDATION

Update all @esbuild dependencies to version 0.24.0 or higher which contains security fixes. Run 'npm update @esbuild/*' and verify with npm audit.

#2 HIGH `src/integrations/supabase/client.ts:13` Est. 3h $75

Security Vulnerability: Supabase Auth Token Storage in localStorage

The Supabase client is configured to store authentication tokens in localStorage, which is vulnerable to XSS attacks. localStorage is accessible to any JavaScript running on the same domain, making tokens easily stealable if an XSS vulnerability exists elsewhere in the application.ts:12.ts:12.

RECOMMENDATION

Switch to using httpOnly cookies for authentication tokens, or implement a more secure storage mechanism like sessionStorage with proper XSS protections. Consider using Supabase's built-in cookie-based auth or implementing a custom secure storage solution.

#3 HIGH `tsconfig.app.json:18` Est. 2h $50

Unsafe TypeScript Compiler Options Disable Critical Safety Checks

The TypeScript configuration disables multiple important safety checks including 'strict', 'strictNullChecks', 'noImplicitAny', and 'noFallthroughCasesInSwitch'. This allows unsafe code patterns that can lead to runtime errors and security vulnerabilities.

RECOMMENDATION

Enable strict mode and critical safety checks. Set 'strict': true, 'strictNullChecks': true, 'noImplicitAny': true, and 'noFallthroughCasesInSwitch': true. Fix any type errors that arise.

#4 HIGH `package-lock.json:1` Est. 0.75h $19

Outdated @babel/runtime with Security Vulnerabilities

@babel/runtime version 7.28.2 is outdated and contains multiple known security vulnerabilities including prototype pollution and regular expression denial of service (ReDoS) issues.

RECOMMENDATION

Update @babel/runtime to version 7.24.0 or higher. Run 'npm update @babel/runtime' and verify the transitive dependency chain.

#5 HIGH `supabase/functions/parse-file/index.ts:40` Est. 2h $50

Insecure File Parsing with Potential Denial of Service

The parse-file edge function (supabase/functions/parse-file/index.ts) implements custom PDF and DOCX parsing using basic regex and text decoding without proper validation. This approach is unreliable and could lead to crashes, memory issues, or denial of service if malformed files are uploaded. The function reads entire files into memory (up to 20MB) and uses inefficient algorithms like regex matching on raw binary data.

RECOMMENDATION

Replace custom parsing with a robust library or service (e.g., Tika, PDF.js for PDFs, mammoth for DOCX). Implement file size limits and validate file structure before processing. Use streaming processing for large files.

#6 HIGH `supabase/functions/scrape-url/index.ts:12` Est. 1h $25

Missing Input Validation for URL Scraping

The scrape-url edge function (supabase/functions/scrape-url/index.ts) accepts any URL from the client without validation. This could allow attackers to request internal URLs (SSRF), malicious sites, or cause excessive API usage. The function passes the URL directly to Firecrawl API without sanitization.tsx:104.

RECOMMENDATION

Implement URL validation: check protocol (allow only http/https), validate domain against allowlist or blocklist, and limit request rate per user. Use a URL parsing library to ensure proper format.

#7 MEDIUM `vite.config.ts:10` Est. 0.5h $13

Vite Development Server Exposed to All Network Interfaces Without Security

The Vite configuration sets 'host: "::"' which binds the development server to all network interfaces (IPv4 and IPv6). This exposes the dev server to the local network without authentication or security measures.

RECOMMENDATION

Restrict dev server to localhost only in development. Use 'host: "localhost"' or conditionally enable network access only when needed with proper authentication.

#8 MEDIUM `supabase/functions/scrape-url/index.ts:4` Est. 0.5h $13

CORS Misconfiguration Allowing Any Origin

All edge functions (scrape-url, parse-file, analyze) set CORS headers to allow any origin ('*'). While this might be intentional for development, in production it exposes the API to cross-origin attacks and unauthorized usage from any website.

RECOMMENDATION

Restrict CORS to specific origins (your frontend domains) in production. Use environment variables to manage allowed origins for different environments.

CATEGORY

Performance

#1 HIGH `Multiple files:87` Est. 1h $25

Multiple Potential Memory Leaks in React Components

Detected potential memory leaks in multiple React components (sidebar.tsx:87, use-mobile.tsx:13, Processing.tsx:37) due to missing cleanup of event listeners, subscriptions, or effects.

RECOMMENDATION

Add proper cleanup functions to all useEffect hooks, remove event listeners on component unmount, and cancel pending promises or subscriptions.

CATEGORY

Code Quality

#1 HIGH `tsconfig.app.json:20` Est. 2h $50

TypeScript Strict Mode Disabled - Runtime Type Safety Compromised

Multiple TypeScript configuration files have critical strictness options disabled: 'strict: false', 'noImplicitAny: false', 'strictNullChecks: false', 'noUnusedLocals: false', 'noUnusedParameters: false'. This allows type errors to pass through compilation, increasing the risk of runtime errors and making refactoring difficult.app.json:18.

RECOMMENDATION

Enable strict TypeScript mode by setting 'strict: true' and enabling all strictness flags. Start with 'strict: true' and address type errors incrementally.

#2 MEDIUM `vite.config.ts:12` Est. 0.125h $3

Insecure Server Configuration - HMR Overlay Disabled

Vite configuration disables the HMR (Hot Module Replacement) error overlay with 'hmr: { overlay: false }'. This prevents runtime error visibility during development, hiding critical errors from developers.

RECOMMENDATION

Remove the 'overlay: false' setting or set it to true to enable error overlays during development.

#3 MEDIUM `src/App.tsx:12` Est. 1h $25

Missing Error Boundaries in App Component

The main App component lacks error boundaries, meaning any uncaught errors in child components will crash the entire application without graceful recovery. This is particularly risky for production applications.

RECOMMENDATION

Implement React Error Boundaries around major sections of the application. Use try-catch blocks for async operations and provide fallback UI components.

#4 MEDIUM `eslint.config.js:23` Est. 0.5h $13

ESLint Disables Critical TypeScript Rules

ESLint configuration explicitly disables '@typescript-eslint/no-unused-vars' rule, allowing unused variables to remain in code. This reduces code quality and can hide real bugs.

RECOMMENDATION

Enable '@typescript-eslint/no-unused-vars' rule with appropriate configuration (e.g., 'warn' or 'error').

#5 MEDIUM `src/pages/Analyze.tsx:58` Est. 1.5h $38

Insufficient Error Handling in Analyze Page API Calls

The Analyze page makes multiple Supabase function calls without proper validation of responses or handling of edge cases. The error handling catches errors but doesn't distinguish between different types of failures or provide specific guidance to users.

RECOMMENDATION

Implement comprehensive error handling with specific error types, user-friendly messages, and retry logic for transient failures. Validate API responses before using them.

CATEGORY

Logic Bugs

#1 HIGH `src/components/ui/carousel.tsx:46` Est. 1h $25

Potential operator misuse in condition

Incorrect operator usage (assignment in condition or loose equality).tsx:72, src/components/ui/carousel.tsx:75, src/components/ui/carousel.tsx:111, and 49 more.

RECOMMENDATION

Use strict comparison operators and remove accidental assignments inside conditional expressions.

ScanFix beta

Technical Due Diligence Report

Generated by ScanFix — an AI-powered platform for code quality, security, and reliability reviews. Recommendations are based on automated analysis and industry benchmarks.

Report #78 · April 15, 2026 · 04:07 PM