

Codebase Analysis Report – AWF

Executive Summary

The codebase contains **1,239,139 total lines** across **4,301 files**, with **1,009,299 lines of actual code**.

The project is primarily composed of **C#, HTML, and XML** technologies, indicating a web application with significant data processing components.

Key Statistics

Metric	Value	Percentage
Total Files	4,301	-
Total Lines	1,239,139	100%
Source Code	1,009,299	81.5%
Comments	120,548	9.7%
Blank Lines	107,292	8.7%

Detailed Language Breakdown

Language	Files	Code	Comments	Blanks	Total Lines
XML	388	310,871	795	446	312,112
C#	1,026	274,741	35,589	49,200	359,530
HTML	1,530	227,804	51,640	24,632	304,076
JavaScript	236	84,742	29,080	21,890	135,712
CSS	110	33,994	170	2,176	36,340
MS SQL	554	29,376	284	914	30,574
Visual Basic	212	28,794	2,886	5,256	36,936
TeX	56	10,784	0	1,944	12,728
XSL	26	4,516	104	482	5,102
Properties	124	1,812	0	270	2,082
XAML	6	1,644	0	2	1,646
Batch	28	80	0	34	114
Markdown	2	68	0	25	93
YAML	1	34	0	7	41
Shell Script	1	32	0	14	46
JSON	1	7	0	0	7
TOTAL	4,301	1,009,299	120,548	107,292	1,239,139

Code Quality Indicators

Documentation Level

Language	Comment Ratio	Assessment
Overall	9.7%	Good (10–15% is std)
C#	12.9%	Well-documented
JavaScript	21.4%	Excellent
HTML	18.5%	Good
CSS	0.5%	Needs improvement
XML	0.3%	Needs improvement

Code Density Metrics

Metric	Value	Assessment
Code-to-comment ratio	8.4:1	Healthy balance
Blank lines percentage	8.7%	Good readability
Average file size	288 lines	Moderate complexity

Technology Stack Analysis

Primary Technologies

1. C# (359,530 lines)

- Core application logic with 1,026 files
- Excellent comment ratio (12.9%)
- Well-documented with 35,589 comments

2. HTML (304,076 lines)

- Extensive frontend with 1,530 files
- Good documentation with 51,640 comments
- Represents significant UI investment

3. XML (312,112 lines)





- Configuration and data processing
- 388 files with minimal comments
- Suggests standardized configuration formats

Supporting Technologies

Technology	Role	Assessment
JavaScript	Client-side functionality	Well-documented
CSS	Styling and presentation	Under-documented
Visual Basic	Legacy/integration code	Moderate docs
MS SQL	Database operations	Minimal docs

Security Analysis







High Risk Areas

Risk Area	Files	Risk Level	Concerns
XML Processing	388	 High	XXE attacks, insecure deserialization
SQL Injection	554	 High	Dynamic query vulnerabilities
Client-Side Sec.	1,766	 Medium	XSS, client-side validation
Legacy Code (VB)	212	 Medium	Outdated security practices

Security Assessment by Language

Language	Security Risk	Primary Concerns	Recommendation
XML	High	XXE, deserialization	Input validation, DTD disabling
MS SQL	High	SQL injection	Use parameterized queries, ORM
HTML/JS	Medium	XSS, CSRF	CSP, sanitization
Visual Basic	Medium	Deprecated practices	Security audit, migration
Properties	Medium	Hardcoded secrets	Secret management system

Risk Assessment Summary

Risk Level	Area	Impact	Priority	Recommendation
 High	XML Processing	Critical	Immediate	XML security hardening
 High	SQL Injection	Critical	Immediate	Audit queries, parameterize
 Medium	Client-Side Sec.	High	Short-term	XSS prevention, CSP
 Medium	Legacy VB Code	Medium	Medium-term	Audit & migrate
 Medium	Config Files	Medium	Short-term	Implement secret management
 Low	C# Backend	Low	Ongoing	Maintain standards

Recommendations

Immediate Actions

1. XML Security Hardening (disable DTD, schema validation, audit configs)
2. SQL Injection Prevention (audit queries, parameterized SQL, ORM)
3. Frontend Security (CSP headers, sanitization, XSS prevention)

Short-term Actions

1. Secret Management (Key Vault, scan configs, secure storage)
2. Dependency Security (audit NuGet/npm, automated scans)
3. Documentation Improvements (CSS, XML configs, security docs)

Long-term Strategy

1. Legacy Modernization (VB → C#, refactoring, decommissioning)
 2. Security Automation (SAST/DAST in CI/CD, auto testing)
 3. Continuous Improvement (training, audits, compliance)
-

Technical Debt Assessment

Positive Indicators

- Good overall documentation standards
- Balanced technology stack
- Maintainable code structure
- Strong C# documentation practices

Areas for Improvement

- XML and SQL security vulnerabilities
- Legacy component maintenance
- Configuration security gaps
- Lack of automated security testing

Overall Assessment:

Medium technical debt, with high-priority security concerns needing immediate attention.

Conclusion

The codebase represents a **well-established enterprise application** with strong engineering practices but notable **security vulnerabilities in XML and SQL**.

- **Overall Security Rating:** Medium (requires immediate hardening)
 - **Next Steps:** XML hardening, SQL audit, security automation rollout
 - **Review Cycle:** Quarterly with dependency and vulnerability scans
-