POLARIS INTELLIGENCE

Technical Due Diligence Report

fastapi

03 April 2026

Ref: a24285de

CONFIDENTIAL

1. Executive Summary

Repository Classification: Library (Confidence: high)
This repository is a published library/package. Dependency findings reflect the library's own dependencies, not a consuming application.

> **Bus factor: Tier 3: Critical Concentration (capped from high — library archetype).** Review recommended to assess key-person dependency.

The table below rates risk across 13 dimensions, from Clean (no findings) to Critical (potential deal impact). Together they form the technical risk profile of the target asset.

CATEGORY	RATING	SUMMARY
Secrets & Credentials	CLEAN	No credential exposure detected in scanned files
Dependency Vulnerabilities	CLEAN	No known vulnerabilities detected in scanned dependencies
Supply Chain Risk	CLEAN	No known supply chain incidents
Licence & IP	CLEAN	Permissive licence, no IP issues detected
Developer Concentration	MEDIUM RISK	Tier 3: Critical Concentration (capped from high — library archetype)
Code Quality	CLEAN	Grade B (62.0/100). 5 high-severity potential security anti-patterns warrant review (see Security Anti-Patterns)
Architecture	MEDIUM RISK	9 circular dependencies (2% of modules)
Malware / Destructive Code	CLEAN	No suspicious patterns detected in scanned files
Test Coverage	CLEAN	Strong testing discipline (53% test ratio, CI gates active)
Infrastructure & Deployment	CLEAN	Deployment partially codified — manual steps required (capped from low — library archetype)
Technical Debt	CLEAN	Minimal tech debt (24 markers (0.3/KLOC), 265 suppressions (2.8/KLOC))
Governance & CI/CD	MEDIUM RISK	Moderate governance gaps (Grade C, 5.3/10)

CATEGORY	RATING	SUMMARY
Engineering Maturity	LOW RISK	Adequate maturity (97.8 releases/yr, 40% community health)

2. Transaction Impact Assessment

This section translates technical findings into their commercial implications for the transaction. Ratings range from Clean (no concern) to Critical (potential deal-breaker), with specific conditions that may need to be met before or after completion.

CATEGORY	ASSESSMENT	DETAIL
Security Exposure	CLEAN	No security issues detected
Operational Risk	MEDIUM RISK	Tier 3: Critical Concentration (capped from high — library archetype); Grade B (62.0/100). 5 high-severity potential security anti-patterns warrant review (see Security Anti-Patterns)
IP & Licence Risk	CLEAN	Permissive licence, no IP issues detected
Integration Complexity	MEDIUM RISK	9 circular dependencies (2% of modules)
Maintenance Burden	MEDIUM RISK	Grade B, 3 quality issues, 24 tech debt markers

Remediation Effort Estimate

3 medium-severity items for the integration roadmap.

Estimates assume a senior developer familiar with the technology stack. Actual effort may vary based on codebase familiarity and organisational context.

3. Scope & Methodology

Repository: https://github.com/tiangolo/fastapi

Analysis date: 03 April 2026

Codebase size: 2,744 files, 357,222 lines

LANGUAGE	LINES
Markdown	241,402

LANGUAGE	LINES
Python	107,493
YAML	6,900
JavaScript	541
CSS	380

Methodology

This report was produced by Polaris Intelligence automated analysis pipeline. The following scanners were applied:

1. **GitHub Enrichment** — project metadata, release cadence, community health
2. **Secret Scanner** — regex pattern matching + context classification
3. **Dependency Scanner** — manifest parsing + OSV vulnerability cross-reference + exploitability analysis
4. **Supply Chain Intelligence** — cross-reference against known supply chain incidents
5. **Licence Auditor** — declared licence + source header contradiction detection
6. **Bus Factor Analysis** — 5-tier developer concentration taxonomy (24-month window)
7. **Code Quality Scorer** — cyclomatic complexity, duplication, security anti-patterns
8. **Architecture Mapper** — import graph, circular dependencies, module coupling
9. **Engineering Maturity** — release discipline, community governance, project signals
10. **Malware Heuristic** — destructive actions, crypto mining, exfiltration, obfuscation
11. **Governance & CI/CD** — OpenSSF Scorecard, branch protection, dependency management

Note: *File counts may vary between sections because each scanner operates on a different subset of files (e.g. quality analysis covers source code files only, while the scope total includes configuration, documentation, and data files).*

4. Secrets & Credentials CLEAN

Hardcoded credentials — API keys, database passwords, tokens — are the most common cause of data breaches. Their presence indicates both an immediate security exposure and a gap in the target's engineering practices that transfers with the acquisition.

No hardcoded secrets or credentials detected.

5. Dependency Vulnerabilities CLEAN

Modern software relies on hundreds of third-party packages. Known vulnerabilities in these dependencies are publicly catalogued and actively exploited. Unpatched critical CVEs represent a quantifiable security liability that transfers to the acquirer.

5 dependencies analysed across 1 manifest.

> **No known vulnerabilities detected.**

Note: *OpenSSF Scorecard reports 12 vulnerabilities via OSV. Polaris found 0 after filtering to versions declared in the manifest. This discrepancy typically arises because OSV includes advisories for version ranges that do not match the pinned versions in this repository.*

> **First-party package checked: PyPI:fastapi. No known CVEs affecting current version.**

Historical CVE Record

These CVEs were filed against prior versions of the target's package. While the current version may not be affected, a history of security incidents indicates recurring risk and may inform the buyer's assessment of the development team's security posture.

fastapi: 2 CVEs on record

SEVERITY	CVE/ID	SUMMARY
HIGH RISK	CVE-2021-32677	Cross-Site Request Forgery (CSRF) in FastAPI
HIGH RISK	CVE-2024-24762	FastAPI is a web framework for building APIs with Python 3.8+ based on standard Python type hints…

Dependency Health

"No known CVEs" does not mean healthy dependencies. Stale or abandoned packages receive no security patches and represent latent risk. Health status is derived from package registry release dates.

STATUS		COUNT
Active (released within 6 months)	CLEAN	3
Stable (released within 1 year)	CLEAN	2

Dependency health score: **9.2/10**

6. Developer Concentration (Bus Factor) MEDIUM RISK

"Bus factor" measures how many developers would need to leave before critical knowledge is lost. High concentration in one developer creates key-person dependency — a material operational risk that can delay integration and increase post-acquisition costs.

Tier 3: Critical Concentration — Bus factor score: **74.6%** [risk capped to MEDIUM for library archetype]

Despite 184 contributors (13 active), a single developer has authored the majority of commits in 74.6% of core files. Effective bus factor is 1. Code review practices and pair programming should be introduced to distribute knowledge more evenly.

Top Contributors

DEVELOPER	COMMITS	OWNED FILES	CORE %	STATUS
Sebastián Ramírez	673	1674	74%	Active
Valentyn	33	5	0%	Active
Rafael de Oliveira Marques	31	6	0%	Active
svlandeg	28	0	0%	Active
Yurii Motov	27	2	0%	Active
Valentyn Druzhynin	16	0	0%	Active
Nils-Hero Lindemann	16	9	0%	Active
Zhongheng Cheng	14	2	0%	Departed
YungYueh ChanLee	14	8	0%	Departed
Quentin Takeda	13	0	0%	Departed

Departed Developer Risk

DEVELOPER	MONTHS INACTIVE	CORE FILES OWNED	RISK
Nils Lindemann	19	2	MEDIUM RISK
Emmanuel Ferdman	9	1	MEDIUM RISK
gitworkflows	19	1	MEDIUM RISK
alv2017	6	0	LOW RISK

DEVELOPER	MONTHS INACTIVE	CORE FILES OWNED	RISK
Naves	9	0	LOW RISK
Zhongheng Cheng	12	0	LOW RISK
Phương Tấn Thành	13	0	LOW RISK
timothy	10	0	LOW RISK
YungYueh ChanLee	15	0	LOW RISK
Alissa	16	0	LOW RISK

7. Licence & Intellectual Property CLEAN

Copyleft licences (GPL, AGPL) require derivative works to be released under the same open-source terms. If copyleft code is embedded in a proprietary product, the acquirer may face an obligation to open-source their own code — or costly remediation to replace the affected components.

Declared Licences

SPDX ID	RISK	SOURCE	FILE
MIT	CLEAN	licence_file	`LICENSE`
MIT	CLEAN	package_metadata	`pyproject.toml`

No licence contamination detected.

8. Code Quality & Technical Debt CLEAN

Code quality directly predicts the cost and speed of post-acquisition development. A low grade signals elevated technical debt — higher bug rates, slower feature delivery, and more expensive onboarding for new developers joining after the transaction.

Quality grade: **B (62.0/100)** — Acceptable

Note: 5 high security anti-patterns detected — see Security Anti-Patterns below. Grade reflects structural quality only, not security posture.

Grade Scale

GRADE	MEANING
A	Excellent maintainability
B	Good engineering quality
C	Moderate technical debt
D	High technical debt
F	Severe structural risk

METRIC	VALUE
Total files	531
Code lines	26,829
Functions	1085
Classes	383
Avg function length	21.9 lines
Avg complexity	2.6

Technical Debt Indicators

The following indicators are informational. At this grade level, they represent normal characteristics of a healthy codebase rather than actionable concerns. Duplication above automated thresholds is common in test files and generated code.

- 9 files exceed 500 lines
- 20 functions with high cyclomatic complexity
- Code duplication at 41.5% (threshold: 5.0%)

Complexity Hotspots

SEVERITY	FILE	FUNCTION	COMPLEXITY
HIGH RISK	`fastapi/routing.py`	get_request_handler	49
HIGH RISK	`fastapi/dependencies/utils.py`	analyze_param	48
HIGH RISK	`fastapi/routing.py`	app	45

SEVERITY	FILE	FUNCTION	COMPLEXITY
HIGH RISK	`fastapi/openapi/utils.py`	get_openapi_path	44
HIGH RISK	`fastapi/encoders.py`	jsonable_encoder	35
HIGH RISK	`fastapi/routing.py`	__init__	31
HIGH RISK	`scripts/docs.py`	remove_unused_docs_src	29
HIGH RISK	`fastapi/routing.py`	include_router	29
HIGH RISK	`fastapi/openapi/utils.py`	get_openapi	29
HIGH RISK	`scripts/notify_translations.py`	main	28

Large Files (>500 lines)

FILE	TOTAL LINES	CODE LINES
`fastapi/routing.py`	4957	4275
`fastapi/applications.py`	4750	4045
`fastapi/param_functions.py`	2461	2323
`fastapi/dependencies/utils.py`	1058	925
`fastapi/params.py`	755	718
`scripts/docs.py`	742	604
`scripts/doc_parsing_utils.py`	734	573
`fastapi/security/oauth2.py`	694	603
`fastapi/openapi/utils.py`	607	571

Security Anti-Patterns

These patterns represent common security vulnerabilities (OWASP Top 10, CWE). Each warrants developer review to confirm whether it represents an actual risk in context.

SEVERITY	PATTERN	LOCATION	DESCRIPTION
HIGH RISK	exec()	`docs_src/sql_databases/tutorial002_py310.py:68`	Dynamic code execution via exec() — potential code injection vector

SEVERITY	PATTERN	LOCATION	DESCRIPTION
HIGH RISK	exec()	`docs_src/sql_da...s/tutorial001_an_py310.py:54` [F1]	Dynamic code execution via exec() — potential code injection vector
HIGH RISK	exec()	`docs_src/sql_databases/tutorial001_py310.py:50`	Dynamic code execution via exec() — potential code injection vector
HIGH RISK	exec()	`docs_src/sql_da...s/tutorial002_an_py310.py:71` [F2]	Dynamic code execution via exec() — potential code injection vector
HIGH RISK	.innerHTML =	`docs/en/docs/js/termynal.js:86`	Direct innerHTML assignment — potential XSS vector

9. Architectural Assessment MEDIUM RISK

526 production modules, 555 internal dependencies, 124 external imports.

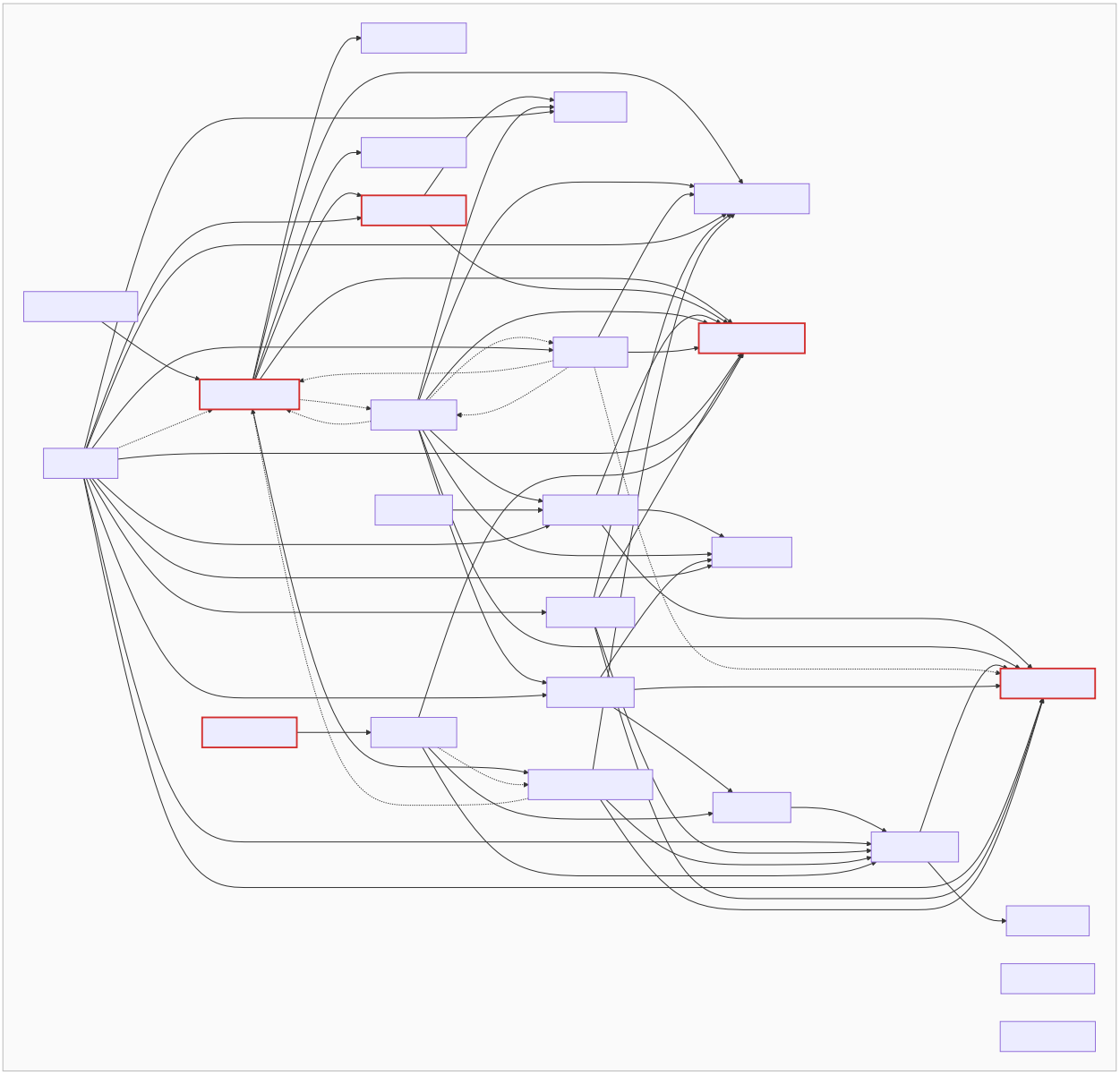
> **9 circular dependencies detected. Circular imports increase coupling and make refactoring more expensive. Breaking these cycles should be prioritised to reduce integration risk.**

Change impact zones: 5 modules have high fan-in (many dependents). Top modules: `fastapi/__init__.py`, `fastapi/responses.py`, `fastapi/exceptions.py`. Changes to these modules carry elevated regression risk.

30 isolated modules detected (no internal imports or dependents). These are typically standalone utilities, examples, platform-specific implementations, or generated code.

Module Dependency Map

Modules are grouped by architectural layer and coloured accordingly. Arrows show import dependencies. Numbers in parentheses indicate how many other modules depend on each file. Thick red borders highlight high fan-in modules (change risk).



Layer Distribution

Modules are grouped by their role in the application: business logic (core functionality), presentation (user-facing), data (storage), and test. A well-structured codebase separates these concerns clearly.

LAYER	MODULES
other	513
infrastructure	9
presentation	4

Entry Points (340)

LOCATION	PATTERN
`docs_src/openap...acks/tutorial001_py310.py:33` [F3]	`app.post(`
`docs_src/reques...s/tutorial001_an_py310.py:8` [F4]	`app.post(`
`docs_src/reques...iles/tutorial001_py310.py:6` [F5]	`app.post(`
`docs_src/body/tutorial002_py310.py:15`	`app.post(`
`docs_src/body/tutorial004_py310.py:15`	`app.put(`
`docs_src/body/tutorial001_py310.py:15`	`app.post(`
`docs_src/body/tutorial003_py310.py:15`	`app.put(`
`docs_src/cookie...dels/tutorial002_py310.py:15` [F6]	`app.get(`
`docs_src/cookie...s/tutorial001_an_py310.py:15` [F7]	`app.get(`
`docs_src/cookie...dels/tutorial001_py310.py:13` [F8]	`app.get(`

Circular Dependencies (9)

```
fastapi/__init__.py → fastapi/routing.py → fastapi/utils.py → fastapi/__init__.py
```

```
fastapi/routing.py → fastapi/utils.py → fastapi/routing.py
```

```
fastapi/_compat/__init__.py → fastapi/_compat/v2.py → fastapi/_compat/__init__.py
```

```
fastapi/__init__.py → fastapi/routing.py → fastapi/utils.py → fastapi/_compat/__init__.py → fastapi/_compat/v2.py → fastapi/__init__.py
```

```
fastapi/__init__.py → fastapi/routing.py → fastapi/__init__.py
```

4 additional cycles not shown.

High Fan-In Modules (Change Risk)

“Fan-in” measures how many other parts of the codebase depend on a given file. High fan-in files are widely relied upon — changes to them carry elevated risk because they can affect many dependent components.

MODULE	DEPENDENTS	RISK
`fastapi/__init__.py`	344	HIGH RISK
`fastapi/responses.py`	45	HIGH RISK
`fastapi/exceptions.py`	17	MEDIUM RISK
`fastapi/security/__init__.py`	15	MEDIUM RISK
`fastapi/_compat/__init__.py`	10	MEDIUM RISK

Isolated Modules (30)

These files neither use nor are used by any other file in the codebase. They may be unused code, standalone utilities, or components loaded indirectly. Each should be reviewed to confirm it is genuinely needed.

- `docs/en/docs/js/custom.js`
- `docs/en/docs/js/init_kapa_widget.js`
- `docs/en/docs/js/termynal.js`
- `docs_src/dependencies/tutorial007_py310.py`
- `docs_src/dependencies/tutorial010_py310.py`
- `docs_src/generate_clients/tutorial004.js`
- `docs_src/generate_clients/tutorial004_py310.py`
- `docs_src/pydantic_v1_in_v2/tutorial001_an_py310.py`
- `docs_src/python_types/tutorial001_py310.py`
- `docs_src/python_types/tutorial002_py310.py`
- `docs_src/python_types/tutorial003_py310.py`
- `docs_src/python_types/tutorial004_py310.py`
- `docs_src/python_types/tutorial005_py310.py`
- `docs_src/python_types/tutorial006_py310.py`
- `docs_src/python_types/tutorial007_py310.py`

9b. Architecture Topology

This section applies graph-theoretic analysis to the module dependency network identified in the Architectural Assessment. Community detection reveals natural subsystem boundaries; centrality analysis identifies critical bridge modules whose failure or refactoring would disproportionately affect the codebase.

METRIC	VALUE
Modules analysed	526
Internal dependencies	555
Subsystems detected	19
Modularity score	0.424 (moderate modularity)
Unclustered modules	136

Subsystem Overview

Communities are groups of modules that are more tightly connected to each other than to the rest of the codebase. Each represents a natural subsystem boundary.

SUBSYSTEM	MODULES
docs_src/ (other, 243 modules)	243
docs_src/bigger_applications/ (other, 5 modules)	5
docs_src/security/ (other, 16 modules)	16
docs_src/server_sent_events/ (other, 6 modules)	6
docs_src/ (other, 2 modules)	2
fastapi/ (other, 42 modules)	42
docs_src/ (other, 2 modules)	2
docs_src/settings/ (other, 2 modules)	2
docs_src/settings/ (other, 2 modules)	2
docs_src/ (other, 2 modules)	2
docs_src/ (other, 8 modules)	8
scripts/ (other, 2 modules)	2
docs_src/settings/ (other, 2 modules)	2
fastapi/ (other, 2 modules)	2
docs_src/ (other, 7 modules)	7

4 additional subsystems not shown.

Bridge Modules (Bottleneck Risk)

Bridge modules sit on critical paths between subsystems. High betweenness centrality means many inter-module communication paths pass through this module — changes here carry elevated risk of cascading failures.

MODULE	CENTRALITY	BRIDGES	FAN-IN / OUT	RISK
`fastapi/__init__.py`	0.0369	16	344 / 9	LOW RISK

God Modules (Excessive Coupling)

"God modules" have direct dependencies spanning many subsystems. They violate separation of concerns and make the codebase harder to modify safely — changes risk unintended side effects across multiple subsystems.

MODULE	SUBSYSTEMS	CROSS-EDGES	TOTAL DEGREE	RISK
`fastapi/__init__.py`	16	108	353	HIGH RISK
`fastapi/responses.py`	4	7	47	MEDIUM RISK

Structural Gaps

Subsystem pairs with very few connecting dependencies that may indicate missing integration.

SUBSYSTEM A	SUBSYSTEM B	CONNECTING EDGES
docs_src/ (other, 243 modules)	scripts/ (other, 2 modules)	0
docs_src/ (other, 243 modules)	fastapi/ (other, 2 modules)	0
docs_src/bigger_applications/ (other, 5 modules)	docs_src/security/ (other, 16 modules)	0
docs_src/bigger_applications/ (other, 5 modules)	docs_src/server_sent_events/ (other, 6 modules)	0
docs_src/bigger_applications/ (other, 5 modules)	docs_src/ (other, 2 modules)	0
docs_src/bigger_applications/ (other, 5 modules)	fastapi/ (other, 42 modules)	0
docs_src/bigger_applications/ (other, 5 modules)	docs_src/ (other, 2 modules)	0
docs_src/bigger_applications/ (other, 5 modules)	docs_src/settings/ (other, 2 modules)	0
docs_src/bigger_applications/ (other, 5 modules)	docs_src/settings/ (other, 2 modules)	0

SUBSYSTEM A	SUBSYSTEM B	CONNECTING EDGES
docs_src/bigger_applications/ (other, 5 modules)	docs_src/ (other, 2 modules)	0

Topology Findings

HIGH RISK **God Module:** Module 'fastapi/__init__.py' connects 16 subsystems with 108 cross-boundary edges. Changes here risk cascading across the codebase.

MEDIUM RISK **God Module:** Module 'fastapi/responses.py' connects 4 subsystems with 7 cross-boundary edges. Changes here risk cascading across the codebase.

10. Test Coverage CLEAN

Automated tests act as safety nets for the codebase. When developers make changes, tests verify nothing else has broken. Strong test coverage means the acquirer's team can modify and extend the code with confidence; weak or absent testing means changes carry a higher risk of introducing undetected problems.

Strong testing discipline — automated tests cover a significant portion of the codebase and run automatically before changes are released.

METRIC	VALUE
Test files	599
Source files	1125
Test-to-source ratio	53.2%
Test functions / cases	2151

This is a healthy ratio, indicating testing is a routine part of development.

The project uses established testing tools (pytest, unittest), indicating the team has invested in testing infrastructure.

> **Tests run automatically before code changes are accepted. This means new code is verified before it reaches the main codebase, reducing the chance of regressions.**

CI configuration found in: .github/workflows/test-redistribute.yml, .github/workflows/issue-manager.yml, .github/workflows/contributors.yml, .github/workflows/people.yml, .github/workflows/labeler.yml, .github/workflows/translate.yml, .github/workflows/build-docs.yml, .github/workflows/notify-translations.yml, .github/workflows/publish.yml, .github/workflows/test.yml, .github/workflows/label-approved.yml, .github/workflows/pre-commit.yml, .github/workflows/detect-conflicts.yml, .github/workflows/deploy-docs.yml, .github/workflows/smokeshow.yml, .github/workflows/sponsors.yml, .github/workflows/add-to-project.yml, .github/workflows/topic-repos.yml, .github/workflows/latest-changes.yml

11. Infrastructure & Deployment CLEAN

This section assesses whether the deployment process — how the software is built, packaged, and released — is documented in code or relies on undocumented manual steps. Codified deployment means a new team can operate the software independently; undocumented deployment creates dependency on the original developers and increases transition risk.

Build infrastructure is partially codified — some CI/CD automation exists for testing or publishing.

Infrastructure Found

CATEGORY	TOOLS / CONFIGURATION
CI/CD Automation	github-actions (19)
Deployment Scripts	deploy-script

As a library distributed via package registry, containerisation, orchestration, and deployment infrastructure are not expected. CI/CD for build and publish is the primary infrastructure concern.

12. Technical Debt CLEAN

Technical debt represents shortcuts, deferred work, and known problems that the development team has acknowledged but not yet fixed. Every codebase carries some debt; what matters is the volume and severity. High technical debt increases the cost of post-acquisition development and the risk that changes introduce new problems.

Minimal technical debt — the codebase is well-maintained with few acknowledged shortcuts or deferred work items.

Developers have left **24** notes in the code flagging work that needs to be done (TODO items, known bugs, temporary workarounds). That is **0.3 markers per 1,000 lines of code**. This density is typical for a well-maintained project.

Debt Markers by Type

TYPE	COUNT
TODO	23
WORKAROUND	1

23 markers are planned work items (TODO) while 1 flags known problems (FIXME, HACK, BUG). A high proportion of FIXME/HACK markers is more concerning than TODOs, as they indicate acknowledged broken or fragile code.

265 warning suppressions detected (2.8 per 1,000 lines) — within normal range for a codebase of this size. These represent deliberate exceptions to coding rules, not a material concern.

4 uses of deprecated APIs detected — normal for a mature codebase and not a material risk at current levels.

6 minor error handling patterns detected (empty catch blocks or broad exception handlers) — within normal range and not a material concern at this volume.

13. Governance & CI/CD Security MEDIUM RISK

Governance measures the security and maturity of development processes — CI pipeline hardening, release signing, code review enforcement, and dependency management. Weak governance increases post-acquisition remediation costs and ongoing operational risk.

Overall governance: C (5.3/10) [OpenSSF Scorecard + local analysis]

OpenSSF Scorecard aggregate: 5.3/10

CI Pipeline Security — Grade D

CHECK	SCORE	SOURCE	DETAIL
Token-Permissions	0/10	scorecard	detected GitHub workflow tokens with excessive permissions
Pinned-Dependencies	0/10	scorecard	dependency not pinned by hash detected -- score normalized to 0
Dangerous-Workflow	10/10	scorecard	no dangerous workflow patterns detected
CI-Tests	10/10	scorecard	13 out of 13 merged PRs checked by a CI test -- score normalized to 10
SAST	0/10	scorecard	SAST tool is not run on all commits -- score normalized to 0

CI pipelines have significant security weaknesses requiring remediation. Workflow tokens have excessive permissions — apply principle of least privilege. No static analysis (SAST) detected — consider CodeQL, Semgrep, or similar.

Release Engineering — Grade A

CHECK	SCORE	SOURCE	DETAIL
Signed-Releases	3/10	scorecard	Releases exist (97.8/yr via GitHub tags) but are not cryptographically signed. Score reflects absence of signing, not absence of releases.
Packaging	N/A	scorecard	packaging workflow not detected
Maintained	10/10	scorecard	30 commits and 14 issue activity found in the last 90 days -- score normalized to 10
release_frequency	10/10	enrichment	97.8 releases/year, 100% semver compliant

Release engineering practices are mature.

Governance Posture — Grade A

CHECK	SCORE	SOURCE	DETAIL
Security-Policy	10/10	scorecard	security policy file detected
Contributors	10/10	scorecard	project has 31 contributing companies or organizations
CII-Best-Practices	0/10	scorecard	no effort to earn an OpenSSF best practices badge detected
License	10/10	scorecard	license file detected

Project demonstrates strong governance practices.

Branch Protection & Code Review — Grade D

CHECK	SCORE	SOURCE	DETAIL
Branch-Protection	3/10	scorecard	branch protection is not maximal on development and all release branches
Code-Review	2/10	scorecard	Found 7/26 approved changesets -- score normalized to 2

Weak or absent branch protection — code can be pushed without review. Significant proportion of changes merged without peer review.

Dependency Management — Grade C

CHECK	SCORE	SOURCE	DETAIL
Dependency-Update-Tool	10/10	scorecard	update tool detected
Vulnerabilities	0/10	scorecard	Scorecard reports 12 via OSV; Polaris found 0 after filtering to versions in the manifest.
Fuzzing	0/10	scorecard	project is not fuzzed

Dependency management has gaps — automated updates or vulnerability tracking missing.

14. Engineering Maturity LOW RISK

Engineering maturity measures the project's operational health beyond source code quality — release discipline, community governance, and project signals that indicate long-term viability.

Overall maturity: **B** — minor maturity gaps only (risk rating: LOW)

Release Cadence

METRIC	VALUE
Releases per year	97.8
Days since last release	1
Semver compliance	100.0%
Grade	A

Community Health

DOCUMENT	STATUS
SECURITY.md	Present
CONTRIBUTING.md	Present
CODE_OF_CONDUCT	Missing
Issue template	Missing

DOCUMENT	STATUS
PR template	Missing
Health score	40%
Grade	D

Project Signals

METRIC	VALUE
Stars	96,800
Forks	8,991
Open issues	165
Contributors	100
Repository created on GitHub	2018-12-08
Grade	A

Note: *GitHub API reports 100 contributors while git history analysis (Bus Factor section) identified 30. This discrepancy arises because GitHub counts all commit authors across the full history, while git analysis may use a limited clone depth or different author-deduplication rules.*

15. Malware & Destructive Action Scan CLEAN

This scan searches for code patterns commonly associated with protestware, supply-chain attacks, and sabotage — filesystem wipers, obfuscated payloads, unauthorised network calls, and install-hook abuse. Findings are heuristic and warrant manual review rather than automatic condemnation.

Files scanned: 1,131

No suspicious patterns detected.

16. Risk Summary & Recommendations

Recommendations are prioritised by their potential impact on the transaction. Immediate and Urgent items should be addressed as conditions precedent; Medium items can be scheduled into the post-acquisition integration roadmap.

PRIORITY	CATEGORY	RECOMMENDED ACTION
MEDIUM RISK	overall	No high-priority recommendations. The codebase presents a low-risk profile.

Appendix A: Raw Data

Dependency Inventory (5 packages)

ECOSYSTEM	PACKAGE	VERSION	VULNS	HEALTH	LAST RELEASE
PyPI	starlette	inherited	0	active	2026-03-22
PyPI	pydantic	inherited	0	active	2025-11-26
PyPI	typing-extensions	inherited	0	stable	2025-08-25
PyPI	typing-inspection	inherited	0	stable	2025-10-01
PyPI	annotated-doc	inherited	0	active	2025-11-10

Module Dependencies (top 30)

MODULE	FAN-IN	LAYER
`fastapi/__init__.py`	344	other
`fastapi/responses.py`	45	other
`fastapi/exceptions.py`	17	other
`fastapi/security/__init__.py`	15	other
`fastapi/_compat/__init__.py`	10	other
`fastapi/encoders.py`	9	other
`fastapi/routing.py`	9	other
`fastapi/openapi/models.py`	8	other
`fastapi/types.py`	8	other
`fastapi/sse.py`	8	other
`fastapi/datastructures.py`	7	other
`fastapi/testclient.py`	6	other
`fastapi/utils.py`	5	other
`fastapi/security/base.py`	5	other
`fastapi/dependencies/models.py`	3	other
`fastapi/staticfiles.py`	3	other
`fastapi/websockets.py`	3	other
`fastapi/openapi/docs.py`	3	other

MODULE	FAN-IN	LAYER
`fastapi/logger.py`	3	other
`docs_src/bigger..._an_py310/dependencies.py` [F9]	2	other
`fastapi/openapi/utils.py`	2	other
`fastapi/params.py`	2	other
`fastapi/background.py`	2	other
`fastapi/param_functions.py`	2	other
`fastapi/security/oauth2.py`	2	other
`fastapi/_compat/v2.py`	2	other
`fastapi/security/utils.py`	2	other
`fastapi/openapi/constants.py`	2	other
`fastapi/dependencies/utils.py`	2	other
`fastapi/exception_handlers.py`	2	other

Appendix B: File Reference

Full file paths for truncated references in the report.

REF	FULL PATH
F1	`docs_src/sql_databases/tutorial001_an_py310.py`
F2	`docs_src/sql_databases/tutorial002_an_py310.py`
F3	`docs_src/openapi_callbacks/tutorial001_py310.py`
F4	`docs_src/request_forms_and_files/tutorial001_an_py310.py`
F5	`docs_src/request_forms_and_files/tutorial001_py310.py`
F6	`docs_src/cookie_param_models/tutorial002_py310.py`
F7	`docs_src/cookie_param_models/tutorial001_an_py310.py`
F8	`docs_src/cookie_param_models/tutorial001_py310.py`
F9	`docs_src/bigger_applications/app_an_py310/dependencies.py`