Conviction

TECHNICAL DUE DILIGENCE REPORT

Powered by Aldemis

northgate

2026-06-04 15:00 UTC

Contents

29
/ 100

northgate

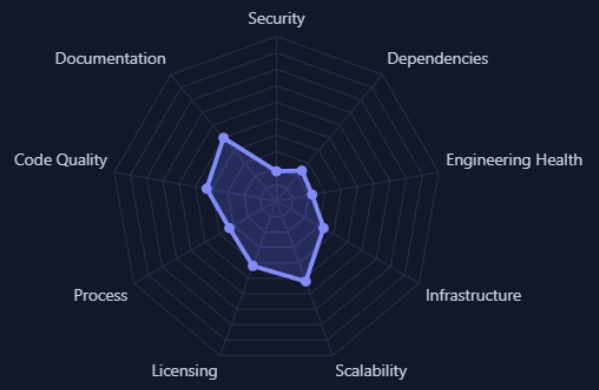
Pass / deep-remediation-required: Northgate shows critical security exposure, severe key-person risk, and a stalled codebase that together warrant a hard pass absent major remediation and price/term protection.

https://github.com/northgate/app.git commit a1b2c3d4e5 main 2026-06-04 15:00 UTC synthesis: claude:claude-opus-4-8

Claude synthesis cost ≈ $0.1763 · 17,781 in / 3,192 out

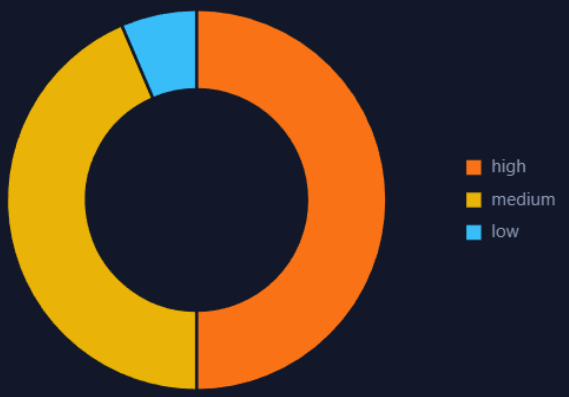
125,622 LOC Lines of code	**598** Total commits	**3** Unique contributors	**0** Active contributors (90d)	**1** Bus factor
0 % Commits from a corporate domain	**2** Contributors using personal email	**0** commits Commits in last 30 days	**0** Commits (last 90d)	**0** Release tags
0 Releases per month	**0.3** % Revert/hotfix rate	**98** days Days since last commit	**43** Test files	**0.171** Test-to-source ratio
58 TODO/FIXME markers	**2** Oversized files (>750 LOC)	**63** Peak file complexity	**no** CI/CD configured	**6** Secrets detected
3 SAST findings	**40** Vulnerable dependencies	**171** Direct dependencies	**120** days Days since dependency update	**0** Strong-copyleft dependencies
27 Weak-copyleft dependencies	**2** Unknown/unlicensed dependencies	**psycopg** Datastores	**no** Caching layer	**no** Async/queue processing
none detected Observability tooling	**none detected** Secrets management	**docker-compose** Containerization	**0** IaC misconfigurations	**27** Languages used
89 Recent pull requests (sampled)	**1** Regions in use	**2** EC2 instances	**3** S3 buckets	**2** RDS instances
2 IAM users	**132** Jira issues	**19** Issues created (90d)	**20** Median resolution (days)	**0.09** Bug : story ratio
24 Sprint spillovers (sample)	**138** Returns from testing (sample)	**29** Confluence pages	**11** Pages updated (90d)	

Dimension scorecard

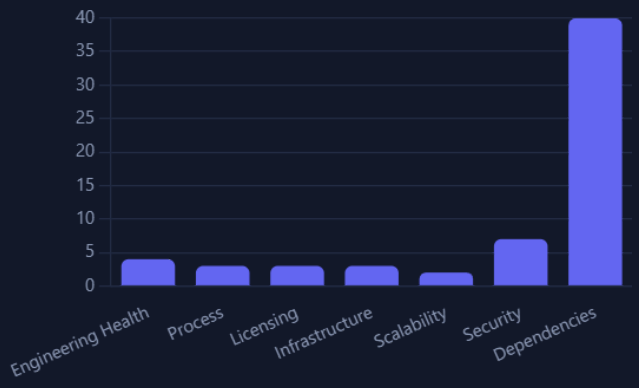


Dimension scores

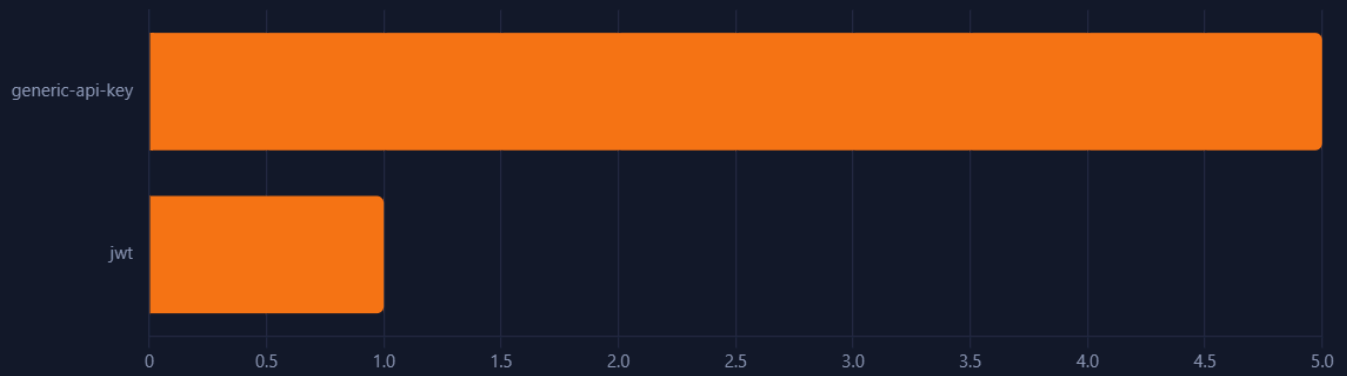
Findings by severity



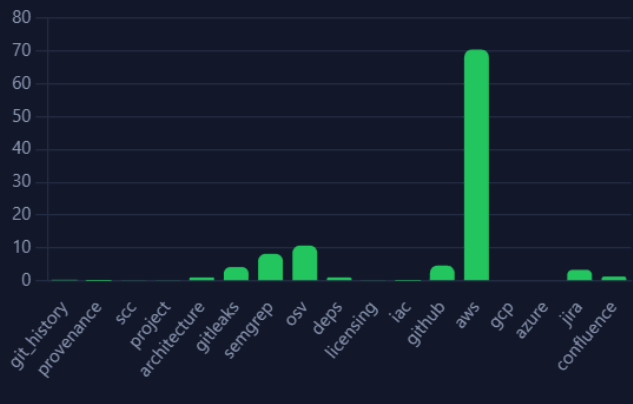
Findings by category



Leaked secrets by type



Scanner runtime (s)



Scanner coverage

- git_history ok
- provenance ok
- scc ok
- project ok
- architecture ok
- gitleaks ok
- semgrep ok
- osv ok
- deps ok
- licensing ok
- iac ok
- github ok
- aws ok
- gcp skipped
- azure skipped
- jira ok
- confluence ok

Red flags

- 6 secrets (API keys and a JWT) leaked into git history — must be treated as compromised, rotated, and purged
- PostgreSQL, RDP and SSH ports exposed to the entire internet (0.0.0.0/0) and 2 RDS databases publicly accessible
- No CloudTrail — AWS account activity is completely unaudited
- 40 vulnerable dependencies / 153 advisories, several with 7–19 known CVEs each
- Bus factor of 1 (65% of commits from one author) combined with 0 commits in 98 days — the project appears stalled and critically dependent on one person
- 100% of commits from personal emails with no corporate domain and no LICENSE/SPDX — clean IP title cannot be evidenced
- TLS certificate verification disabled in code (rejectUnauthorized:false) and sensitive values written to logs
- No CI/CD, no observability, and a Jira backlog growing (19 created / 0 resolved in 90d) with 65% ticket rework from QA

Strengths

- Codebase is containerized (docker-compose) and the single IaC file scanned showed 0 misconfigurations
- Strong historical PR discipline: 88 of 89 sampled pull requests were merged
- Dependency licensing is largely clean — 419 permissive, 0 strong-copyleft; vulnerable packages are mostly upgradable to patched releases
- Low revert/hotfix rate (0.3%) suggests changes that did land were relatively stable
- Some maintained documentation exists (29 Confluence pages, 11 updated in last 90 days)

Executive summary

Northgate presents a high-risk profile dominated by security and engineering-health concerns. The scanners (all primary scanners ran successfully; only GCP and Azure were skipped for lack of credentials) surfaced an actively exploitable infrastructure: databases and admin ports open to the public internet, publicly accessible RDS instances, no CloudTrail auditing, and six secrets leaked into git history. SAST additionally found TLS verification disabled and sensitive data being logged. These are not theoretical — a partner should assume credentials are compromised and the environment is exposed.

Engineering health is the second major concern. With 598 commits but a bus factor of 1 (one author wrote 65%), only 3 contributors total, and zero commits in the last 90 days against 255 in the prior period, the project looks stalled and dangerously dependent on one individual. Because 100% of commits come from personal email addresses and there is no LICENSE or SPDX declaration, clean IP title cannot be evidenced from the artifacts alone — signed IP-assignment agreements should be a condition to close.

Delivery maturity is weak: no CI/CD, no observability, a Jira backlog growing (19 created vs 0 resolved in 90 days), and 65% of sampled tickets bouncing back from QA — pointing to a fragile process. The dependency picture is poor in volume (40 vulnerable packages, 153 advisories) but largely remediable through upgrades, and licensing is mostly permissive. Note that code-quality metrics are distorted because 81% of the codebase is JSON and the largest file is ~101k LOC, so the application logic footprint is smaller than the headline LOC suggests.

Net: this is a pass-or-deeply-remediate situation. If the partner wishes to proceed, condition any term sheet on IP-assignment execution, evidence of a retained/incentivized lead engineer, full secret rotation and history purge, lockdown of AWS network and audit configuration, a dependency-patch sprint, and direct confirmation of how production is built, deployed, and monitored.

Reviewer notes

AUDITOR COMMENT

Add a auditor comment...

TEAM COMMENT

Add a team comment...

Findings (56)

SEVERITY	FINDING	DIMENSION	SCANNER
HIGH	**Leaked secret: Detected a Generic API Key, potentially exposing access to various services and sensitive operations. (5×)** gitleaks matched rule 'generic-api-key' 5 time(s) across 4 file(s) in the git history.	Security	`gitleaks` ›
HIGH	**Leaked secret: Uncovered a JSON Web Token, which may lead to unauthorized access to web applications and sensitive user data. (1×)** gitleaks matched rule 'jwt' 1 time(s) across 1 file(s) in the git history.	Security	`gitleaks` ›
HIGH	**SAST: rejectUnauthorized:false disables TLS certificate verification (MITM risk). (1×)** semgrep rule 'js-tls-reject-unauthorized-false' matched 1 time(s) across 1 file(s).	Security	`semgrep` ›
HIGH	**Vulnerable dependency: py@1.9.0 (3 advisory)** PyPI package 'py' 1.9.0 has 3 known advisory/advisories: CVE-2020-29651, CVE-2022-42969, CVE-2020-29651 (via repo/backend-cdk/requirements-dev.txt)	Dependencies	`osv` ›
HIGH	**Vulnerable dependency: tornado@6.5.1 (4 advisory)** PyPI package 'tornado' 6.5.1 has 4 known advisory/advisories: CVE-2026-31958, GHSA-78cv-mqj4-43f7, CVE-2026-35536, CVE-2026-31958 (via repo/backend/playground/requirements.txt)	Dependencies	`osv` ›
HIGH	**Vulnerable dependency: urllib3@2.5.0 (5 advisory)** PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/playground/requirements.txt)	Dependencies	`osv` ›
HIGH	**Vulnerable dependency: urllib3@2.5.0 (5 advisory)** PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/cognitoToDb/requirements.txt)	Dependencies	`osv` ›
HIGH	**Vulnerable dependency: urllib3@2.5.0 (5 advisory)** PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/departuresGet/requirements.txt)	Dependencies	`osv` ›
HIGH	**Vulnerable dependency: urllib3@2.5.0 (5 advisory)** PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/gpsPointsSave/requirements.txt)	Dependencies	`osv` ›
HIGH	**Vulnerable dependency: urllib3@2.5.0 (5 advisory)** PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/journeysGet/requirements.txt)	Dependencies	`osv` ›
HIGH	**Vulnerable dependency: urllib3@2.5.0 (5 advisory)** PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/journeysSave/requirements.txt)	Dependencies	`osv` ›
HIGH	**Vulnerable dependency: urllib3@1.26.9 (11 advisory)** PyPI package 'urllib3' 1.26.9 has 11 known advisory/advisories: CVE-2023-43804, CVE-2023-45803, CVE-2026-44431, CVE-2025-66471, CVE-2024-37891, CVE-2026-21441, CVE-2023-45803, CVE-2025-66418... (via repo/backend/src/journeysUpdate/requirements.txt)	Dependencies	`osv` ›
HIGH	**Vulnerable dependency: urllib3@2.5.0 (5 advisory)** PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/stationsGet/requirements.txt)	Dependencies	`osv` ›

SEVERITY	FINDING	DIMENSION	SCANNER	
HIGH	**Vulnerable dependency: urllib3@2.5.0 (5 advisory)** PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/stationsGetAll/requirements.txt)	Dependencies	osv	›
HIGH	**Vulnerable dependency: urllib3@2.5.0 (5 advisory)** PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/userDelete/requirements.txt)	Dependencies	osv	›
HIGH	**Vulnerable dependency: urllib3@2.5.0 (5 advisory)** PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/userStationsGet/requirements.txt)	Dependencies	osv	›
HIGH	**Vulnerable dependency: urllib3@2.5.0 (5 advisory)** PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/userStationsSet/requirements.txt)	Dependencies	osv	›
HIGH	**Vulnerable dependency: requests@2.9.2 (7 advisory)** PyPI package 'requests' 2.9.2 has 7 known advisory/advisories: CVE-2018-18074, CVE-2023-32681, CVE-2024-47081, CVE-2024-35195, CVE-2026-25645, CVE-2023-32681, CVE-2018-18074 (via repo/backend/tests/requirements.txt)	Dependencies	osv	›
HIGH	**Vulnerable dependency: urllib3@1.24.3 (14 advisory)** PyPI package 'urllib3' 1.24.3 has 14 known advisory/advisories: CVE-2020-26137, CVE-2021-33503, CVE-2023-43804, CVE-2023-45803, CVE-2026-44431, CVE-2025-66471, CVE-2024-37891, CVE-2026-21441... (via repo/backend/tests/requirements.txt)	Dependencies	osv	›
HIGH	**Vulnerable dependency: flatted@3.3.3 (2 advisory)** npm package 'flatted' 3.3.3 has 2 known advisory/advisories: CVE-2026-32141, CVE-2026-33228 (via repo/map-gps-tool/package-lock.json)	Dependencies	osv	›
HIGH	**Vulnerable dependency: minimatch@3.1.2 (3 advisory)** npm package 'minimatch' 3.1.2 has 3 known advisory/advisories: CVE-2026-27904, CVE-2026-26996, CVE-2026-27903 (via repo/map-gps-tool/package-lock.json)	Dependencies	osv	›
HIGH	**Vulnerable dependency: minimatch@9.0.5 (3 advisory)** npm package 'minimatch' 9.0.5 has 3 known advisory/advisories: CVE-2026-27904, CVE-2026-26996, CVE-2026-27903 (via repo/map-gps-tool/package-lock.json)	Dependencies	osv	›
HIGH	**Vulnerable dependency: next@16.1.6 (19 advisory)** npm package 'next' 16.1.6 has 19 known advisory/advisories: CVE-2026-44575, CVE-2026-45109, CVE-2026-44573, CVE-2026-44572, CVE-2026-27980, CVE-2026-44574, GHSA-8h8q-6873-q5fj, CVE-2026-44578... (via repo/map-gps-tool/package-lock.json)	Dependencies	osv	›
HIGH	**Vulnerable dependency: picomatch@2.3.1 (2 advisory)** npm package 'picomatch' 2.3.1 has 2 known advisory/advisories: CVE-2026-33672, CVE-2026-33671 (via repo/map-gps-tool/package-lock.json)	Dependencies	osv	›
HIGH	**Vulnerable dependency: picomatch@4.0.3 (2 advisory)** npm package 'picomatch' 4.0.3 has 2 known advisory/advisories: CVE-2026-33672, CVE-2026-33671 (via repo/map-gps-tool/package-lock.json)	Dependencies	osv	›
HIGH	**Vulnerable dependency: uuid@13.0.0 (1 advisory)** npm package 'uuid' 13.0.0 has 1 known advisory/advisories: CVE-2026-41907 (via repo/map-gps-tool/package-lock.json)	Dependencies	osv	›

SEVERITY	FINDING	DIMENSION	SCANNER	
HIGH	**Vulnerable dependency: pyjwt@2.8 (7 advisory)** PyPI package 'pyjwt' 2.8 has 7 known advisory/advisories: CVE-2025-45768, CVE-2026-32597, CVE-2026-48522, CVE-2026-48524, CVE-2026-48525, CVE-2026-48526, CVE-2026-32597 (via repo/network-align/requirements.txt)	Dependencies	`osv`	›
HIGH	**Security groups open to the internet (5 rule(s))** Sensitive ports reachable from 0.0.0.0/0: PostgreSQL, RDP, SSH.	Security	`aws`	›
HIGH	**Publicly accessible RDS database(s) (2)** RDS instances are reachable from the public internet.	Security	`aws`	›
HIGH	**No CloudTrail trail configured** No CloudTrail trail was found — API activity is not being audited.	Infrastructure	`aws`	›
MEDIUM	**No commits authored from a corporate email domain** All authorship comes from personal or masked email addresses — there is no company-domain identity in the git history. IP ownership cannot be evidenced from authorship alone.	Engineering Health	`provenance`	›
MEDIUM	**No CI/CD configuration found** No CI configuration (GitHub Actions, GitLab CI, Jenkins, etc.) was detected.	Process	`project`	›
MEDIUM	**No LICENSE file in the repository** No LICENSE/COPYING file — licensing intent and IP terms are unclear.	Licensing	`project`	›
MEDIUM	**No observability tooling detected** No error-tracking, metrics, or tracing library (Sentry, Datadog, OpenTelemetry, Prometheus, etc.) was found in the dependencies. Without observability the team is effectively flying blind in production — incidents are detected late and diagnosed slowly.	Infrastructure	`architecture`	›
MEDIUM	**SAST: A logging/print call references a secret- or PII-like value (password/token/ssn/credit-card). Sensitive data in logs is a common privacy/com (2×)** semgrep rule 'log-of-sensitive-value' matched 2 time(s) across 2 file(s).	Security	`semgrep`	›
MEDIUM	**Vulnerable dependency: pytest@6.2.5 (1 advisory)** PyPI package 'pytest' 6.2.5 has 1 known advisory/advisories: CVE-2025-71176 (via repo/backend-cdk/requirements-dev.txt)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: pygments@2.19.2 (1 advisory)** PyPI package 'pygments' 2.19.2 has 1 known advisory/advisories: CVE-2026-4539 (via repo/backend/playground/requirements.txt)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: idna@3.10 (1 advisory)** PyPI package 'idna' 3.10 has 1 known advisory/advisories: CVE-2026-45409 (via repo/backend/playground/requirements.txt)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: python-dotenv@1.1.1 (1 advisory)** PyPI package 'python-dotenv' 1.1.1 has 1 known advisory/advisories: CVE-2026-28684 (via repo/backend/playground/requirements.txt)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: requests@2.32.4 (1 advisory)** PyPI package 'requests' 2.32.4 has 1 known advisory/advisories: CVE-2026-25645 (via repo/backend/playground/requirements.txt)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: idna@3.10 (1 advisory)** PyPI package 'idna' 3.10 has 1 known advisory/advisories: CVE-2026-45409 (via repo/backend/src/departuresGet/requirements.txt)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: requests@2.32.4 (1 advisory)** PyPI package 'requests' 2.32.4 has 1 known advisory/advisories: CVE-2026-25645 (via repo/backend/src/departuresGet/requirements.txt)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: pygments@2.9.0 (3 advisory)**	Dependencies	`osv`	›

SEVERITY	FINDING	DIMENSION	SCANNER	
	PyPI package 'pygments' 2.9.0 has 3 known advisory/advisories: CVE-2022-40896, CVE-2026-4539, CVE-2022-40896 (via repo/backend/tests/requirements.txt)			
MEDIUM	**Vulnerable dependency: ajv@6.12.6 (1 advisory)** npm package 'ajv' 6.12.6 has 1 known advisory/advisories: CVE-2025-69873 (via repo/map-gps-tool/package-lock.json)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: brace-expansion@1.1.12 (1 advisory)** npm package 'brace-expansion' 1.1.12 has 1 known advisory/advisories: CVE-2026-33750 (via repo/map-gps-tool/package-lock.json)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: brace-expansion@2.0.2 (1 advisory)** npm package 'brace-expansion' 2.0.2 has 1 known advisory/advisories: CVE-2026-33750 (via repo/map-gps-tool/package-lock.json)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: postcss@8.4.31 (1 advisory)** npm package 'postcss' 8.4.31 has 1 known advisory/advisories: CVE-2026-41305 (via repo/map-gps-tool/package-lock.json)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: postcss@8.5.6 (1 advisory)** npm package 'postcss' 8.5.6 has 1 known advisory/advisories: CVE-2026-41305 (via repo/map-gps-tool/package-lock.json)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: djangorestframework@3.15 (1 advisory)** PyPI package 'djangorestframework' 3.15 has 1 known advisory/advisories: CVE-2024-21520 (via repo/network-align/requirements.txt)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: python-dotenv@1.0 (1 advisory)** PyPI package 'python-dotenv' 1.0 has 1 known advisory/advisories: CVE-2026-28684 (via repo/network-align/requirements.txt)	Dependencies	`osv`	›
MEDIUM	**Vulnerable dependency: requests@2.31 (3 advisory)** PyPI package 'requests' 2.31 has 3 known advisory/advisories: CVE-2024-47081, CVE-2024-35195, CVE-2026-25645 (via repo/network-align/requirements.txt)	Dependencies	`osv`	›
MEDIUM	**Weak-copyleft (LGPL/MPL/EPL) dependencies in use (27)** Weak-copyleft licenses impose obligations (e.g. publishing modifications to the licensed component, or dynamic-linking constraints) that are usually manageable but should be confirmed for compliance.	Licensing	`licensing`	›
MEDIUM	**No recognized open-source license declared** GitHub reports no SPDX license for the repository.	Licensing	`github`	›
MEDIUM	**No IAM password policy** No password policy is configured for IAM users.	Infrastructure	`aws`	›
MEDIUM	**High rework — tickets returned from testing (65%)** 86 of 132 sampled tickets moved backward out of testing/QA (138 times).	Process	`jira`	›
LOW	**No secrets-management tooling detected** No managed secret store (Vault, AWS/GCP/Azure Secrets Manager, SOPS, Doppler) was found in dependencies, suggesting secrets are handled via environment variables / files. Combined with any leaked-secret findings, this points to immature secret hygiene.	Security	`architecture`	›

Ignored findings (6)

SEVERITY	FINDING	DIMENSION	SCANNER
HIGH	**Bus factor of 1 — severe key-person risk** A single contributor authored 65% of all commits. Loss of this person would critically endanger the codebase. *Ignored — known and accepted, reasonable for size of business*	Engineering Health	`git_history` ›
MEDIUM	**Declining commit velocity** 0 commits in the last 90 days vs 255 in the prior 90 — development is slowing materially. *Ignored — this is expected given the business situation*	Engineering Health	`git_history` ›
MEDIUM	**Backlog growing faster than it is resolved** 19 issues created vs 0 resolved in the last 90 days. *Ignored — accepted for project stage*	Process	`jira` ›
LOW	**Low recent activity** No commits in 98 days. *Ignored — expected, ignoring*	Engineering Health	`git_history` ›
LOW	**No caching layer detected** A web service backed by a datastore was detected, but no cache (Redis/Memcached). Read-heavy workloads typically need caching to scale cost-effectively — worth confirming the scaling plan. *Ignored — by design, accepted*	Scalability	`architecture` ›
LOW	**No async/background-processing layer detected** No message queue or background-job system (Kafka/RabbitMQ/SQS/Celery/Sidekiq/BullMQ) was found. Heavy work done inline in request handlers limits throughput and resilience as load grows. *Ignored — this is by design, ignoring*	Scalability	`architecture` ›

Appendix — Findings detail & remediation

HIGH **1. Leaked secret: Detected a Generic API Key, potentially exposing access to various services and sensitive operations. (5×)**

Security · scanner gitleaks

gitleaks matched rule 'generic-api-key' 5 time(s) across 4 file(s) in the git history.

A secret of this type was committed to the repository's git history. Even if deleted from the current code it remains retrievable from history, so it must be treated as compromised. Depending on the credential, exposure can enable unauthorized access to cloud accounts, payment systems, or customer data.

REMEDIATION STEPS

1. Revoke or rotate the affected credential at the provider immediately — assume it is compromised.
2. Review the provider's access/audit logs for unauthorized use during the exposure window.
3. Purge the secret from git history (git-filter-repo or BFG), force-push, and have all clones re-cloned.
4. Confirm whether the credential was ever live in production and assess blast radius if so.
5. Add secret scanning to CI and a pre-commit hook (gitleaks) to prevent recurrence.
6. Move secrets into a managed secret store (Vault, AWS/GCP Secrets Manager) referenced at runtime.

EVIDENCE

rule	generic-api-key
count	5
files	backend/src/departuresGet/app.py, backend/src/userDelete/app.py, frontend/.env, network-align/.env
example_commit	6f2c91ab48d3e07b5c1af9e24d86b30c7ae5192d

REFERENCES

git-filter-repo — https://github.com/newren/git-filter-repo
GitHub: remove sensitive data — https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/removing-sensitive-data-from-a-repository
OWASP Secrets Management — https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html

---

HIGH **2. Leaked secret: Uncovered a JSON Web Token, which may lead to unauthorized access to web applications and sensitive user data. (1×)**

Security · scanner gitleaks

gitleaks matched rule 'jwt' 1 time(s) across 1 file(s) in the git history.

A secret of this type was committed to the repository's git history. Even if deleted from the current code it remains retrievable from history, so it must be treated as compromised. Depending on the credential, exposure can enable unauthorized access to cloud accounts, payment systems, or customer data.

REMEDIATION STEPS

1. Revoke or rotate the affected credential at the provider immediately — assume it is compromised.
2. Review the provider's access/audit logs for unauthorized use during the exposure window.
3. Purge the secret from git history (git-filter-repo or BFG), force-push, and have all clones re-cloned.
4. Confirm whether the credential was ever live in production and assess blast radius if so.
5. Add secret scanning to CI and a pre-commit hook (gitleaks) to prevent recurrence.
6. Move secrets into a managed secret store (Vault, AWS/GCP Secrets Manager) referenced at runtime.

EVIDENCE

rule	jwt
count	1
files	frontend/android/app/src/main/AndroidManifest.xml
example_commit	b83d17ce62f90a4d15eb7c38f0629ad4e517cb80

REFERENCES

git-filter-repo — https://github.com/newren/git-filter-repo
GitHub: remove sensitive data — https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/removing-sensitive-data-from-a-repository
OWASP Secrets Management — https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html

---

HIGH 3. SAST: rejectUnauthorized:false disables TLS certificate verification (MITM risk). (1×)

Security · scanner semgrep

semgrep rule 'js-tls-reject-unauthorized-false' matched 1 time(s) across 1 file(s).

A static-analysis (SAST) finding — a code-level pattern flagged as a potential security weakness. SAST findings need triage for exploitability but are a standard part of a security review.

REMEDIATION STEPS

1. Triage the finding for exploitability in context (true vs false positive).
2. Remediate confirmed issues; suppress reviewed false positives with a rule comment.
3. Run semgrep in CI to prevent regressions on new code.

EVIDENCE

rule	js-tls-reject-unauthorized-false
count	1
files	repo\map-gps-tool\app\api\db_connection.ts

REFERENCES

semgrep — https://semgrep.dev
OWASP Top 10 — https://owasp.org/www-project-top-ten/

HIGH 4. Vulnerable dependency: py@1.9.0 (3 advisory)

Dependencies · scanner osv

PyPI package 'py' 1.9.0 has 3 known advisory/advisories: CVE-2020-29651, CVE-2022-42969, CVE-2020-29651 (via repo/backend-cdk/requirements-dev.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	py
version	1.9.0
ecosystem	PyPI
advisories	CVE-2020-29651, CVE-2022-42969, CVE-2020-29651
source	repo/backend-cdk/requirements-dev.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 5. Vulnerable dependency: tornado@6.5.1 (4 advisory)

Dependencies · scanner osv

PyPI package 'tornado' 6.5.1 has 4 known advisory/advisories: CVE-2026-31958, GHSA-78cv-mqj4-43f7, CVE-2026-35536, CVE-2026-31958 (via repo/backend/playground/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	tornado
version	6.5.1
ecosystem	PyPI
advisories	CVE-2026-31958, GHSA-78cv-mqj4-43f7, CVE-2026-35536, CVE-2026-31958
source	repo/backend/playground/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 6. Vulnerable dependency: urllib3@2.5.0 (5 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/playground/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	2.5.0
ecosystem	PyPI
advisories	CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431
source	repo/backend/playground/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 7. Vulnerable dependency: urllib3@2.5.0 (5 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/cognitoToDb/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	2.5.0
ecosystem	PyPI
advisories	CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431
source	repo/backend/src/cognitoToDb/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 8. Vulnerable dependency: urllib3@2.5.0 (5 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/departuresGet/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	2.5.0
ecosystem	PyPI
advisories	CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431
source	repo/backend/src/departuresGet/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 9. Vulnerable dependency: urllib3@2.5.0 (5 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/gpsPointsSave/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	2.5.0
ecosystem	PyPI
advisories	CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431
source	repo/backend/src/gpsPointsSave/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 10. Vulnerable dependency: urllib3@2.5.0 (5 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/journeysGet/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	2.5.0
ecosystem	PyPI
advisories	CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431
source	repo/backend/src/journeysGet/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 11. Vulnerable dependency: urllib3@2.5.0 (5 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/journeysSave/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	2.5.0
ecosystem	PyPI
advisories	CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431
source	repo/backend/src/journeysSave/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 12. Vulnerable dependency: urllib3@1.26.9 (11 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 1.26.9 has 11 known advisory/advisories: CVE-2023-43804, CVE-2023-45803, CVE-2026-44431, CVE-2025-66471, CVE-2024-37891, CVE-2026-21441, CVE-2023-45803, CVE-2025-66418… (via repo/backend/src/journeysUpdate/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	1.26.9
ecosystem	PyPI
advisories	CVE-2023-43804, CVE-2023-45803, CVE-2026-44431, CVE-2025-66471, CVE-2024-37891, CVE-2026-21441, CVE-2023-45803, CVE-2025-66418, CVE-2025-50181, CVE-2026-44431, CVE-2023-43804
source	repo/backend/src/journeysUpdate/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 13. Vulnerable dependency: urllib3@2.5.0 (5 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/stationsGet/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	2.5.0
ecosystem	PyPI
advisories	CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431
source	repo/backend/src/stationsGet/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 14. Vulnerable dependency: urllib3@2.5.0 (5 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/stationsGetAll/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	2.5.0
ecosystem	PyPI
advisories	CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431
source	repo/backend/src/stationsGetAll/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 15. Vulnerable dependency: urllib3@2.5.0 (5 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/userDelete/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	2.5.0
ecosystem	PyPI
advisories	CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431
source	repo/backend/src/userDelete/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 16. Vulnerable dependency: urllib3@2.5.0 (5 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/userStationsGet/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	2.5.0
ecosystem	PyPI
advisories	CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431
source	repo/backend/src/userStationsGet/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 17. Vulnerable dependency: urllib3@2.5.0 (5 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 2.5.0 has 5 known advisory/advisories: CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431 (via repo/backend/src/userStationsSet/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	2.5.0
ecosystem	PyPI
advisories	CVE-2026-44431, CVE-2025-66471, CVE-2026-21441, CVE-2025-66418, CVE-2026-44431
source	repo/backend/src/userStationsSet/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 18. Vulnerable dependency: requests@2.9.2 (7 advisory)

Dependencies · scanner osv

PyPI package 'requests' 2.9.2 has 7 known advisory/advisories: CVE-2018-18074, CVE-2023-32681, CVE-2024-47081, CVE-2024-35195, CVE-2026-25645, CVE-2023-32681, CVE-2018-18074 (via repo/backend/tests/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	requests
version	2.9.2
ecosystem	PyPI
advisories	CVE-2018-18074, CVE-2023-32681, CVE-2024-47081, CVE-2024-35195, CVE-2026-25645, CVE-2023-32681, CVE-2018-18074
source	repo/backend/tests/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 19. Vulnerable dependency: urllib3@1.24.3 (14 advisory)

Dependencies · scanner osv

PyPI package 'urllib3' 1.24.3 has 14 known advisory/advisories: CVE-2020-26137, CVE-2021-33503, CVE-2023-43804, CVE-2023-45803, CVE-2026-44431, CVE-2025-66471, CVE-2024-37891, CVE-2026-21441… (via repo/backend/tests/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	urllib3
version	1.24.3
ecosystem	PyPI
advisories	CVE-2020-26137, CVE-2021-33503, CVE-2023-43804, CVE-2023-45803, CVE-2026-44431, CVE-2025-66471, CVE-2024-37891, CVE-2026-21441, CVE-2023-45803, CVE-2025-66418, CVE-2025-50181, CVE-2026-44431, CVE-2023-43804, CVE-2020-26137
source	repo/backend/tests/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 20. Vulnerable dependency: flatted@3.3.3 (2 advisory)

Dependencies · scanner osv

npm package 'flatted' 3.3.3 has 2 known advisory/advisories: CVE-2026-32141, CVE-2026-33228 (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	flatted
version	3.3.3
ecosystem	npm
advisories	CVE-2026-32141, CVE-2026-33228
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 21. Vulnerable dependency: minimatch@3.1.2 (3 advisory)

Dependencies · scanner osv

npm package 'minimatch' 3.1.2 has 3 known advisory/advisories: CVE-2026-27904, CVE-2026-26996, CVE-2026-27903 (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	minimatch
version	3.1.2
ecosystem	npm
advisories	CVE-2026-27904, CVE-2026-26996, CVE-2026-27903
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 22. Vulnerable dependency: minimatch@9.0.5 (3 advisory)

Dependencies · scanner osv

npm package 'minimatch' 9.0.5 has 3 known advisory/advisories: CVE-2026-27904, CVE-2026-26996, CVE-2026-27903 (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	minimatch
version	9.0.5
ecosystem	npm
advisories	CVE-2026-27904, CVE-2026-26996, CVE-2026-27903
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 23. Vulnerable dependency: next@16.1.6 (19 advisory)

Dependencies · scanner osv

npm package 'next' 16.1.6 has 19 known advisory/advisories: CVE-2026-44575, CVE-2026-45109, CVE-2026-44573, CVE-2026-44572, CVE-2026-27980, CVE-2026-44574, GHSA-8h8q-6873-q5fj, CVE-2026-44578… (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	next
version	16.1.6
ecosystem	npm
advisories	CVE-2026-44575, CVE-2026-45109, CVE-2026-44573, CVE-2026-44572, CVE-2026-27980, CVE-2026-44574, GHSA-8h8q-6873-q5fj, CVE-2026-44578, CVE-2026-44581, CVE-2026-29057, CVE-2026-44580, CVE-2026-27979, CVE-2026-44577, CVE-2026-27977, CVE-2026-44579, CVE-2026-27978, GHSA-q4gf-8mx6-v5v3, CVE-2026-44582, CVE-2026-44576
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 24. Vulnerable dependency: picomatch@2.3.1 (2 advisory)

Dependencies · scanner osv

npm package 'picomatch' 2.3.1 has 2 known advisory/advisories: CVE-2026-33672, CVE-2026-33671 (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	picomatch
version	2.3.1
ecosystem	npm
advisories	CVE-2026-33672, CVE-2026-33671
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 25. Vulnerable dependency: picomatch@4.0.3 (2 advisory)

Dependencies · scanner osv

npm package 'picomatch' 4.0.3 has 2 known advisory/advisories: CVE-2026-33672, CVE-2026-33671 (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	picomatch
version	4.0.3
ecosystem	npm
advisories	CVE-2026-33672, CVE-2026-33671
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 26. Vulnerable dependency: uuid@13.0.0 (1 advisory)

Dependencies · scanner osv

npm package 'uuid' 13.0.0 has 1 known advisory/advisories: CVE-2026-41907 (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	uuid
version	13.0.0
ecosystem	npm
advisories	CVE-2026-41907
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 27. Vulnerable dependency: pyjwt@2.8 (7 advisory)

Dependencies · scanner osv

PyPI package 'pyjwt' 2.8 has 7 known advisory/advisories: CVE-2025-45768, CVE-2026-32597, CVE-2026-48522, CVE-2026-48524, CVE-2026-48525, CVE-2026-48526, CVE-2026-32597 (via repo/network-align/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	pyjwt
version	2.8
ecosystem	PyPI
advisories	CVE-2025-45768, CVE-2026-32597, CVE-2026-48522, CVE-2026-48524, CVE-2026-48525, CVE-2026-48526, CVE-2026-32597
source	repo/network-align/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

HIGH 28. Security groups open to the internet (5 rule(s))

Security · scanner aws

Sensitive ports reachable from 0.0.0.0/0: PostgreSQL, RDP, SSH.

An AWS infrastructure weakness identified via read-only API review. Cloud misconfigurations are a leading cause of breaches and a core infrastructure diligence area — confirm each finding against the intended architecture.

REMEDIATION STEPS

1. Validate the finding against the account owner's intended architecture and data sensitivity.
2. Remediate following AWS Well-Architected / CIS Benchmark guidance for the affected service.
3. Restrict network exposure (security groups, public access) and enforce encryption at rest + in transit.
4. Enable continuous posture monitoring (AWS Security Hub / Config) to prevent regressions.

EVIDENCE

exposures	[object Object], [object Object], [object Object], [object Object], [object Object]

REFERENCES

AWS Well-Architected — Security — https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/welcome.html
CIS AWS Foundations Benchmark — https://www.cisecurity.org/benchmark/amazon_web_services

---

HIGH 29. Publicly accessible RDS database(s) (2)

Security · scanner aws

RDS instances are reachable from the public internet.

An AWS infrastructure weakness identified via read-only API review. Cloud misconfigurations are a leading cause of breaches and a core infrastructure diligence area — confirm each finding against the intended architecture.

REMEDIATION STEPS

1. Validate the finding against the account owner's intended architecture and data sensitivity.
2. Remediate following AWS Well-Architected / CIS Benchmark guidance for the affected service.
3. Restrict network exposure (security groups, public access) and enforce encryption at rest + in transit.
4. Enable continuous posture monitoring (AWS Security Hub / Config) to prevent regressions.

EVIDENCE

instances	db-primary-01, db-reporting-01

REFERENCES

AWS Well-Architected — Security — https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/welcome.html
CIS AWS Foundations Benchmark — https://www.cisecurity.org/benchmark/amazon_web_services

---

HIGH 30. No CloudTrail trail configured

Infrastructure · scanner aws

No CloudTrail trail was found — API activity is not being audited.

An AWS infrastructure weakness identified via read-only API review. Cloud misconfigurations are a leading cause of breaches and a core infrastructure diligence area — confirm each finding against the intended architecture.

REMEDIATION STEPS

1. Validate the finding against the account owner's intended architecture and data sensitivity.
2. Remediate following AWS Well-Architected / CIS Benchmark guidance for the affected service.
3. Restrict network exposure (security groups, public access) and enforce encryption at rest + in transit.
4. Enable continuous posture monitoring (AWS Security Hub / Config) to prevent regressions.

REFERENCES

AWS Well-Architected — Security — https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/welcome.html
CIS AWS Foundations Benchmark — https://www.cisecurity.org/benchmark/amazon_web_services

---

MEDIUM 31. No commits authored from a corporate email domain

Engineering Health · scanner provenance

All authorship comes from personal or masked email addresses — there is no company-domain identity in the git history. IP ownership cannot be evidenced from authorship alone.

An IP-ownership signal from commit authorship. Code authored from personal or masked email accounts (rather than a corporate domain) cannot evidence that the company holds clean title — contributors may be contractors or founders' personal accounts without signed IP assignment.

REMEDIATION STEPS

1. Map the personal-email authors to people and confirm employment/contractor status.
2. Obtain signed contributor / IP-assignment agreements for every contributor.
3. Reconcile masked (no-reply) authors to identities via the platform.
4. Make clean, documented IP title a condition to close.

EVIDENCE

personal_pct	100
masked_pct	0

REFERENCES

Why IP assignment matters (YC) — https://www.ycombinator.com/library

---

MEDIUM 32. No CI/CD configuration found

Process · scanner project

No CI configuration (GitHub Actions, GitLab CI, Jenkins, etc.) was detected.

A repository-maturity signal (tests, CI/CD, containerization, hygiene files). Weak engineering scaffolding raises regression risk and slows safe change — a core code-quality and process diligence area.

REMEDIATION STEPS

1. Confirm the gap with the engineering lead (tooling may live outside the repo).
2. Weigh the maturity gap against the company stage and the integration plan.
3. Prioritise tests + CI as the foundation for safe iteration post-investment.

REFERENCES

Test coverage (Martin Fowler) — https://martinfowler.com/bliki/TestCoverage.html

---

MEDIUM 33. No LICENSE file in the repository

Licensing · scanner project

No LICENSE/COPYING file — licensing intent and IP terms are unclear.

A repository-maturity signal (tests, CI/CD, containerization, hygiene files). Weak engineering scaffolding raises regression risk and slows safe change — a core code-quality and process diligence area.

REMEDIATION STEPS

1. Confirm the gap with the engineering lead (tooling may live outside the repo).
2. Weigh the maturity gap against the company stage and the integration plan.
3. Prioritise tests + CI as the foundation for safe iteration post-investment.

REFERENCES

Test coverage (Martin Fowler) — https://martinfowler.com/bliki/TestCoverage.html

---

MEDIUM 34. No observability tooling detected

Infrastructure · scanner architecture

No error-tracking, metrics, or tracing library (Sentry, Datadog, OpenTelemetry, Prometheus, etc.) was found in the dependencies. Without observability the team is effectively flying blind in production — incidents are detected late and diagnosed slowly.

An architecture/operational-maturity signal inferred from the technology stack (datastores, caching, async processing, observability, secrets management). These shape how the system scales and how operable it is in production.

REMEDIATION STEPS

1. Confirm the detected stack with the engineering lead (tooling may be configured outside manifests).
2. Validate the scaling plan against projected load — caching and async processing for read/write-heavy paths.
3. Confirm production observability (error tracking, metrics, tracing) and incident response.
4. Factor any maturity gaps into the post-investment technical roadmap.

EVIDENCE

checked	dependency manifests

REFERENCES

AWS Well-Architected — https://aws.amazon.com/architecture/well-architected/

MEDIUM **35. SAST: A logging/print call references a secret- or PII-like value (password/token/ssn/credit-card). Sensitive data in logs is a common privacy/com (2×)**

Security · scanner semgrep

semgrep rule 'log-of-sensitive-value' matched 2 time(s) across 2 file(s).

A static-analysis (SAST) finding — a code-level pattern flagged as a potential security weakness. SAST findings need triage for exploitability but are a standard part of a security review.

REMEDIATION STEPS

1. Triage the finding for exploitability in context (true vs false positive).
2. Remediate confirmed issues; suppress reviewed false positives with a rule comment.
3. Run semgrep in CI to prevent regressions on new code.

EVIDENCE

rule	log-of-sensitive-value
count	2
files	repo\backend\src\departuresGet\lambda_function.py, repo\backend\src\userDelete\lambda_function.py

REFERENCES

semgrep — https://semgrep.dev
OWASP Top 10 — https://owasp.org/www-project-top-ten/

MEDIUM **36. Vulnerable dependency: pytest@6.2.5 (1 advisory)**

Dependencies · scanner osv

PyPI package 'pytest' 6.2.5 has 1 known advisory/advisories: CVE-2025-71176 (via repo/backend-cdk/requirements-dev.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	pytest
version	6.2.5
ecosystem	PyPI
advisories	CVE-2025-71176
source	repo/backend-cdk/requirements-dev.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 37. Vulnerable dependency: pygments@2.19.2 (1 advisory)

Dependencies · scanner osv

PyPI package 'pygments' 2.19.2 has 1 known advisory/advisories: CVE-2026-4539 (via repo/backend/playground/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	pygments
version	2.19.2
ecosystem	PyPI
advisories	CVE-2026-4539
source	repo/backend/playground/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 38. Vulnerable dependency: idna@3.10 (1 advisory)

Dependencies · scanner osv

PyPI package 'idna' 3.10 has 1 known advisory/advisories: CVE-2026-45409 (via repo/backend/playground/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	idna
version	3.10
ecosystem	PyPI
advisories	CVE-2026-45409
source	repo/backend/playground/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 39. Vulnerable dependency: python-dotenv@1.1.1 (1 advisory)

Dependencies · scanner osv

PyPI package 'python-dotenv' 1.1.1 has 1 known advisory/advisories: CVE-2026-28684 (via repo/backend/playground/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	python-dotenv
version	1.1.1
ecosystem	PyPI
advisories	CVE-2026-28684
source	repo/backend/playground/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 40. Vulnerable dependency: requests@2.32.4 (1 advisory)

Dependencies · scanner osv

PyPI package 'requests' 2.32.4 has 1 known advisory/advisories: CVE-2026-25645 (via repo/backend/playground/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	requests
version	2.32.4
ecosystem	PyPI
advisories	CVE-2026-25645
source	repo/backend/playground/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 41. Vulnerable dependency: idna@3.10 (1 advisory)

Dependencies · scanner osv

PyPI package 'idna' 3.10 has 1 known advisory/advisories: CVE-2026-45409 (via repo/backend/src/departuresGet/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	idna
version	3.10
ecosystem	PyPI
advisories	CVE-2026-45409
source	repo/backend/src/departuresGet/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 42. Vulnerable dependency: requests@2.32.4 (1 advisory)

Dependencies · scanner osv

PyPI package 'requests' 2.32.4 has 1 known advisory/advisories: CVE-2026-25645 (via repo/backend/src/departuresGet/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	requests
version	2.32.4
ecosystem	PyPI
advisories	CVE-2026-25645
source	repo/backend/src/departuresGet/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 43. Vulnerable dependency: pygments@2.9.0 (3 advisory)

Dependencies · scanner osv

PyPI package 'pygments' 2.9.0 has 3 known advisory/advisories: CVE-2022-40896, CVE-2026-4539, CVE-2022-40896 (via repo/backend/tests/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	pygments
version	2.9.0
ecosystem	PyPI
advisories	CVE-2022-40896, CVE-2026-4539, CVE-2022-40896
source	repo/backend/tests/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 44. Vulnerable dependency: ajv@6.12.6 (1 advisory)

Dependencies · scanner osv

npm package 'ajv' 6.12.6 has 1 known advisory/advisories: CVE-2025-69873 (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	ajv
version	6.12.6
ecosystem	npm
advisories	CVE-2025-69873
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM **45. Vulnerable dependency: brace-expansion@1.1.12 (1 advisory)**

Dependencies · scanner osv

npm package 'brace-expansion' 1.1.12 has 1 known advisory/advisories: CVE-2026-33750 (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	brace-expansion
version	1.1.12
ecosystem	npm
advisories	CVE-2026-33750
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM **46. Vulnerable dependency: brace-expansion@2.0.2 (1 advisory)**

Dependencies · scanner osv

npm package 'brace-expansion' 2.0.2 has 1 known advisory/advisories: CVE-2026-33750 (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	brace-expansion
version	2.0.2
ecosystem	npm
advisories	CVE-2026-33750
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM **47. Vulnerable dependency: postcss@8.4.31 (1 advisory)**

Dependencies · scanner osv

npm package 'postcss' 8.4.31 has 1 known advisory/advisories: CVE-2026-41305 (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	postcss
version	8.4.31
ecosystem	npm
advisories	CVE-2026-41305
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM **48. Vulnerable dependency: postcss@8.5.6 (1 advisory)**

Dependencies · scanner osv

npm package 'postcss' 8.5.6 has 1 known advisory/advisories: CVE-2026-41305 (via repo/map-gps-tool/package-lock.json)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	postcss
version	8.5.6
ecosystem	npm
advisories	CVE-2026-41305
source	repo/map-gps-tool/package-lock.json

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 49. Vulnerable dependency: djangorestframework@3.15 (1 advisory)

Dependencies · scanner osv

PyPI package 'djangorestframework' 3.15 has 1 known advisory/advisories: CVE-2024-21520 (via repo/network-align/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	djangorestframework
version	3.15
ecosystem	PyPI
advisories	CVE-2024-21520
source	repo/network-align/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 50. Vulnerable dependency: python-dotenv@1.0 (1 advisory)

Dependencies · scanner osv

PyPI package 'python-dotenv' 1.0 has 1 known advisory/advisories: CVE-2026-28684 (via repo/network-align/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	python-dotenv
version	1.0
ecosystem	PyPI
advisories	CVE-2026-28684
source	repo/network-align/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 51. Vulnerable dependency: requests@2.31 (3 advisory)

Dependencies · scanner osv

PyPI package 'requests' 2.31 has 3 known advisory/advisories: CVE-2024-47081, CVE-2024-35195, CVE-2026-25645 (via repo/network-align/requirements.txt)

A dependency with one or more known security advisories (CVEs/GHSAs) is in use. Vulnerable dependencies are a common breach vector and a standard diligence red flag — especially when the advisory is network-exploitable.

REMEDIATION STEPS

1. Identify the patched version from the linked advisory.
2. Upgrade the dependency and run the full test suite to catch breaking changes.
3. If no fix exists, assess exploitability in your usage context and consider a workaround, fork, or replacement.
4. Re-run the dependency scan to confirm the advisory clears.
5. Adopt automated dependency updates (Dependabot/Renovate) and fail CI on new high-severity advisories.

EVIDENCE

package	requests
version	2.31
ecosystem	PyPI
advisories	CVE-2024-47081, CVE-2024-35195, CVE-2026-25645
source	repo/network-align/requirements.txt

REFERENCES

OSV database — https://osv.dev
osv-scanner — https://github.com/google/osv-scanner

MEDIUM 52. Weak-copyleft (LGPL/MPL/EPL) dependencies in use (27)

Licensing · scanner licensing

Weak-copyleft licenses impose obligations (e.g. publishing modifications to the licensed component, or dynamic-linking constraints) that are usually manageable but should be confirmed for compliance.

A dependency-license obligation that can affect the company's right to keep its product proprietary. Strong-copyleft (GPL/AGPL) code linked into a closed-source product can compel source disclosure — a material IP-contamination risk and a standard legal-diligence finding.

REMEDIATION STEPS

1. Have counsel review each flagged license against how the dependency is used (linked, modified, distributed).
2. Replace strong-copyleft dependencies with permissively-licensed equivalents, or isolate them behind a network/service boundary.
3. Resolve unknown/unlicensed dependencies to their actual terms and confirm usage rights.
4. Add a license-policy gate (e.g. allow-list) to CI to prevent regressions.
5. Make a clean third-party-license bill of materials a condition to close.

EVIDENCE

packages	@img/sharp-libvips-darwin-arm64@1.2.4, @img/sharp-libvips-darwin-x64@1.2.4, @img/sharp-libvips-linux-arm64@1.2.4, @img/sharp-libvips-linux-arm@1.2.4, @img/sharp-libvips-linux-ppc64@1.2.4, @img/sharp-libvips-linux-riscv64@1.2.4, @img/sharp-libvips-linux-s390x@1.2.4, @img/sharp-libvips-linux-x64@1.2.4, @img/sharp-libvips-linuxmusl-arm64@1.2.4, @img/sharp-libvips-linuxmusl-x64@1.2.4, @img/sharp-wasm32@0.34.5, @img/sharp-win32-arm64@0.34.5, @img/sharp-win32-ia32@0.34.5, @img/sharp-win32-x64@0.34.5, axe-core@4.11.1, lightningcss-android-arm64@1.30.2, lightningcss-darwin-arm64@1.30.2, lightningcss-darwin-x64@1.30.2, lightningcss-freebsd-x64@1.30.2, lightningcss-linux-arm-gnueabihf@1.30.2, lightningcss-linux-arm64-gnu@1.30.2, lightningcss-linux-arm64-musl@1.30.2, lightningcss-linux-x64-gnu@1.30.2, lightningcss-linux-x64-musl@1.30.2, lightningcss-win32-arm64-msvc@1.30.2
count	27

REFERENCES

SPDX license list — https://spdx.org/licenses/
Copyleft (GNU) — https://www.gnu.org/licenses/copyleft.en.html

---

MEDIUM 53. No recognized open-source license declared

Licensing · scanner github

GitHub reports no SPDX license for the repository.

No recognized open-source license is declared. With multiple historical contributors, the right to use, relicense, or sell the code may be unclear — a material legal diligence item.

REMEDIATION STEPS

1. Clarify the intended license / proprietary status with the company.
2. Confirm all contributors are employees or covered by CLAs / assignment agreements.
3. Check for incorporated third-party or copyleft code that could impose obligations.
4. Make clean IP ownership a condition to close.

EVIDENCE

license	

REFERENCES

SPDX license list — https://spdx.org/licenses/
choosealicense.com — https://choosealicense.com/

MEDIUM 54. No IAM password policy

Infrastructure · scanner aws

No password policy is configured for IAM users.

An AWS infrastructure weakness identified via read-only API review. Cloud misconfigurations are a leading cause of breaches and a core infrastructure diligence area — confirm each finding against the intended architecture.

REMEDIATION STEPS

1. Validate the finding against the account owner's intended architecture and data sensitivity.
2. Remediate following AWS Well-Architected / CIS Benchmark guidance for the affected service.
3. Restrict network exposure (security groups, public access) and enforce encryption at rest + in transit.
4. Enable continuous posture monitoring (AWS Security Hub / Config) to prevent regressions.

REFERENCES

AWS Well-Architected — Security — https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/welcome.html
CIS AWS Foundations Benchmark — https://www.cisecurity.org/benchmark/amazon_web_services

---

MEDIUM 55. High rework — tickets returned from testing (65%)

Process · scanner jira

86 of 132 sampled tickets moved backward out of testing/QA (138 times).

An engineering-process signal derived from Jira (sprint flow, rework, cycle time, estimation accuracy). Process maturity affects predictability of delivery and is a standard diligence area.

REMEDIATION STEPS

1. Validate the signal with the engineering lead against their known ways of working.
2. Probe sprint planning, definition-of-done, and QA practices in management interviews.
3. Compare delivery predictability (commit vs complete) against the roadmap's assumptions.

EVIDENCE

returns	138
tickets_returned	86
sample	132

LOW **56. No secrets-management tooling detected**

Security · scanner architecture

No managed secret store (Vault, AWS/GCP/Azure Secrets Manager, SOPS, Doppler) was found in dependencies, suggesting secrets are handled via environment variables / files. Combined with any leaked-secret findings, this points to immature secret hygiene.

An architecture/operational-maturity signal inferred from the technology stack (datastores, caching, async processing, observability, secrets management). These shape how the system scales and how operable it is in production.

REMEDIATION STEPS

1. Confirm the detected stack with the engineering lead (tooling may be configured outside manifests).
2. Validate the scaling plan against projected load — caching and async processing for read/write-heavy paths.
3. Confirm production observability (error tracking, metrics, tracing) and incident response.
4. Factor any maturity gaps into the post-investment technical roadmap.

EVIDENCE

checked	dependency manifests

REFERENCES

AWS Well-Architected — https://aws.amazon.com/architecture/well-architected/

HIGH **57. Bus factor of 1 — severe key-person risk** *— ignored: known and accepted, reasonable for size of business*

Engineering Health · scanner git_history

A single contributor authored 65% of all commits. Loss of this person would critically endanger the codebase.

A single contributor authored the majority of commits, concentrating critical product knowledge in one person. Their departure would materially endanger the codebase and roadmap (key-person risk).

REMEDIATION STEPS

1. Confirm the individual's retention, equity/vesting, and notice period.
2. Assess documentation and onboarding maturity — could another engineer take over?
3. Introduce pair programming / mandatory review to spread knowledge.
4. Make a documented succession/continuity plan a condition to close.

EVIDENCE

top_author	lead.dev@example.com
share	0.645

REFERENCES

Bus factor — https://en.wikipedia.org/wiki/Bus_factor

MEDIUM **58. Declining commit velocity** *— ignored: this is expected given the business situation*

Engineering Health · scanner git_history

0 commits in the last 90 days vs 255 in the prior 90 — development is slowing materially.

A signal about team, process, or project continuity.

REMEDIATION STEPS

1. Reconcile with management.
2. Confirm continuity and knowledge distribution.

EVIDENCE

commits_90d	0
commits_prev_90d	255

MEDIUM **59. Backlog growing faster than it is resolved** — *ignored: accepted for project stage*

Process · scanner jira

19 issues created vs 0 resolved in the last 90 days.

An engineering-process signal derived from Jira (sprint flow, rework, cycle time, estimation accuracy). Process maturity affects predictability of delivery and is a standard diligence area.

REMEDIATION STEPS

1. Validate the signal with the engineering lead against their known ways of working.
2. Probe sprint planning, definition-of-done, and QA practices in management interviews.
3. Compare delivery predictability (commit vs complete) against the roadmap's assumptions.

EVIDENCE

created_90d	19
resolved_90d	0

LOW **60. Low recent activity** — *ignored: expected, ignoring*

Engineering Health · scanner git_history

No commits in 98 days.

Development activity has slowed recently. Not necessarily a problem, but worth confirming the cadence reflects the team's actual current output.

REMEDIATION STEPS

1. Confirm whether the slowdown is a release lull, holiday period, or a real decline.
2. Cross-check against the team's stated headcount and velocity.

EVIDENCE

last_commit_age_days	98

LOW **61. No caching layer detected** — *ignored: by design, accepted*

Scalability · scanner architecture

A web service backed by a datastore was detected, but no cache (Redis/Memcached). Read-heavy workloads typically need caching to scale cost-effectively — worth confirming the scaling plan.

An architecture/operational-maturity signal inferred from the technology stack (datastores, caching, async processing, observability, secrets management). These shape how the system scales and how operable it is in production.

REMEDIATION STEPS

1. Confirm the detected stack with the engineering lead (tooling may be configured outside manifests).
2. Validate the scaling plan against projected load — caching and async processing for read/write-heavy paths.
3. Confirm production observability (error tracking, metrics, tracing) and incident response.
4. Factor any maturity gaps into the post-investment technical roadmap.

EVIDENCE

datastores	psycopg
web_frameworks	django, gin, tornado

REFERENCES

AWS Well-Architected — https://aws.amazon.com/architecture/well-architected/

LOW **62. No async/background-processing layer detected** *— ignored: this is by design, ignoring*

Scalability · scanner architecture

No message queue or background-job system (Kafka/RabbitMQ/SQS/Celery/Sidekiq/BullMQ) was found. Heavy work done inline in request handlers limits throughput and resilience as load grows.

An architecture/operational-maturity signal inferred from the technology stack (datastores, caching, async processing, observability, secrets management). These shape how the system scales and how operable it is in production.

REMEDIATION STEPS

1. Confirm the detected stack with the engineering lead (tooling may be configured outside manifests).
2. Validate the scaling plan against projected load — caching and async processing for read/write-heavy paths.
3. Confirm production observability (error tracking, metrics, tracing) and incident response.
4. Factor any maturity gaps into the post-investment technical roadmap.

EVIDENCE

web_frameworks	django, gin, tornado

REFERENCES

AWS Well-Architected — https://aws.amazon.com/architecture/well-architected/