Quality Requirements

Performance Requirements

Performance is an important quality attribute for AEGIS. The system's primary purpose is to provide a platform that accommodates DFIR teams simultaneous collaboration during ongoing investigations. Thus, reasonable response times are key to maintain workflow continuity without affecting the user's analysis process.

Stimulus Source
Platform user
Stimulus
Users accessing case management features, uploading evidence files, and engaging in real-time collaboration during active investigations
Response
• System provides immediate feedback for user interactions • Evidence upload and case access operations complete without delays • Real-time collaboration features remain responsive during simultaneous use
Response Measure
• Web interface pages load within **2 seconds** • Evidence files up to **1GB** upload within **10 seconds** • Real-time collaboration features respond within **500 milliseconds** • System handles **5 concurrent users** without performance degradation
Environment
Normal operational conditions
Artefact
AEGIS web interface, database queries, evidence upload system, and real-time collaboration features

Performance Strategies

Caching: Store frequently accessed data in memory to reduce database queries and improve response times for common operations.

AEGIS will cache case metadata, user permissions, and frequently accessed evidence listings in memory within the Go backend to avoid repeated database queries, particularly for cases accessed multiple times during active investigations.

Indexing: Create database indexes on frequently queried fields to improve search performance and reduce query response times.

AEGIS will index evidence metadata fields (case ID, file hash, timestamps) in the database to ensure evidence searches and case lookups complete within the 1-second target response time.

Performance Patterns

- Load Balancer
- Throttling

Scalability Requirements

AEGIS should be able to handle multiple users as teams have varying sizes and large amounts of diverse evidence associated with cases. To realise the platform's intended usage of scaling dynamically to accommodate increased system usage for full-scale DFIR operations, AEGIS must be capable of scaling dynamically to accommodate increased demand from the number of users to the data they handle. Scalability is critical to ensure low-latency performance, maintain collaboration fluidity, and preserve data integrity as operational demands increase. This includes supporting high-throughput evidence ingestion, concurrent annotation and messaging workflows, and efficient retrieval of case artifacts.

Stimulus Source
Multiple investigators working on cases simultaneously
Stimulus
Increasing number of concurrent users accessing cases, uploading evidence, and using collaboration features
Response

System maintains functionality as user load increases Case and evidence data can grow without system failure Collaboration features work consistently across multiple users
Response Measure
System supports **5 concurrent users** without failure Handles **20 total evidence files** across all test cases Real-time collaboration supports **3 simultaneous annotation threads** Database maintains response times under **5 seconds** with **100 total evidence records**
Environment
Design of the system and deployment strategy
Artefact
Authentication system, case management features, evidence storage, and real-time collaboration components

Scalability Strategies

Spread load across time: Distribute workload evenly by implementing queuing mechanisms for resource-intensive operations to prevent system overload during peak usage.

When multiple large evidence files are uploaded simultaneously, concurrent users perform complex searches, or multiple chat conversations occur, AEGIS will queue these operations and manage chat message delivery efficiently to prevent system bottlenecks.

Scalability Patterns

- Microservices
- Service-oriented Architecture (SOA)

Security

Security is a critical quality attribute in the AEGIS platform, as it must protect digital evidence from unauthorized access, alteration, or deletion. There is a need to prevent evidence tampering through encryption, role-based access control (RBAC), and exhaustive logging to maintain evidence integrity and confidentiality. Given the sensitivity of digital evidence, especially in legal or criminal investigations, unauthorised access or changes could compromise the entire case. Malicious insiders or external attackers may attempt to breach the system; hence, enforcing security through multiple defence layers is essential. Security mechanisms such as end-to-end encryption, logging of all access attempts, and enforcing strict RBAC ensure only authorised users can access or manipulate the data, while all actions are auditable. By implementing this level of security, AEGIS fulfils the goal of ensuring evidence integrity.

Stimulus Source
External attacker or malicious insider
Stimulus
Attempts to access, alter, or delete stored evidence
Response
System denies unauthorised access and encrypts, and logs all interactions
Response Measure
· All unauthorised access is blocked · All accesses logged · Data at rest and in transit is encrypted · RBAC rules enforces per role
Environment
During normal operation and when evidence is being accessed or uploaded
Artefact
Evidence files and logs

Architectural strategy(s):

Authentication:

Authentication ensures only verified users can interact with the platform. AEGIS requires role-based access and user-specific permissions to limit access to sensitive evidence. Without strong authentication, unauthorised users may bypass controls. Implementing secure authentication guarantees that only legitimate users enter the system.This reinforces the security posture of the system and complements RBAC, a critical requirement for digital forensics platforms.

Authorisation:

Authorization restricts what authenticated users can access or do. The system implements RBAC to limit visibility and access based on assigned roles. Even among trusted users, not all should access all evidence. For instance, a DFIR team member shouldn't have admin privileges. RBAC ensures fine-grained control over actions and data visibility. This aligns with legal compliance requirements for access control and minimises insider threat vectors.

Audit logging:

Audit logs provide a traceable record of all user and system actions. The platform logs every activity for audit and chain-of-custody purposes. Logs ensure that any action, authorised or attempted unauthorised, is recorded for forensic reconstruction, debugging, or legal verification. Logging is essential to maintain chain-of-custody, detect breaches, and support accountability in a legal context.

Hashing and integrity checks:

Hashing validates that data has not been tampered with.While not explicitly named, the proposal’s emphasis on integrity suggests the use of hashing for verifying evidence authenticity. When evidence is uploaded or accessed, hashing (e.g., SHA-256) ensures no bits have changed. This is critical for legal admissibility of digital evidence. Supports the core aim of preventing tampering and enabling provable evidence integrity

End-to-end encryption:

Encryption protects evidence confidentiality during transit and at rest.AEGIS mandates both end-to-end encryption for communication and encryption at rest for stored evidence. Encryption ensures that even if data is intercepted or exfiltrated, it remains unreadable to unauthorized actors. It protects against external breaches and internal data leaks. Fulfills compliance requirements and aligns with best practices for safeguarding forensic data.

Architectural pattern:

Microkernel

This pattern promotes modular security services that can be independently updated.

Availability

Availability is a vital quality attribute in the AEGIS platform, as uninterrupted access to evidence and services is critical during time-sensitive investigations. The system must have high uptime, rapid fail over, and minimal recovery time, while ensuring no data loss or corruption during failures. In forensic operations, even short outages can disrupt workflows, delay evidence analysis, and jeopardize legal procedures. High availability ensures that parts of the system remain functional during compartmental failures. This includes real-time collaboration, uninterrupted access to evidence files, and responsive communication features. Automated recovery mechanisms are essential to mitigate service disruptions. Ensuring availability directly supports the platform's mission, which is to facilitate efficient, collaborative forensic investigations.

Stimulus Source
User (investigator), automated monitoring service, or another subsystem
Stimulus
A request is made to access the system or evidence while a component is down
Response
The system reroutes the request, activates a backup service, or degrades gracefully
Response Measure
· System uptime > 98% · Has a Mean Time To Repair (MTTR) of at most 2 hours. · No data loss ot corruption
Environment
During normal operation or when the system is under heavy load
Artefact
Evidence storage, collaboration module (i.e., secure chat, annotation threads), communication service

Architectural strategy(s):

Health monitoring:

Health monitoring continuously tracks system performance and detects failures early.The system is expected to initiate failover or restart components when a service becomes

unavailable. This allows rapid detection and recovery, supporting the MTTR and failover targets, thus maintaining high availability.

Automated restarts:

Automated restarts allow failed components to be rebooted without manual intervention. The platform requires minimal downtime and quick recovery, with MTTR < 2. Restarting containers, services, or nodes automatically ensures service continuity without waiting for human operators. This is particularly useful during temporary crashes or memory leaks.

Architectural pattern(s):

Microservices

This pattern maximises availability by ensuring separation on concern and support for redundancy and failover mechanisms. The client initiates the request and the server can replicate that request in high availability clusters.

Reliability

Reliability is critical for AEGIS because digital forensic investigations require the system to operate consistently and without failure, preserving evidence integrity and ensuring uninterrupted workflow.Forensic systems like AEGIS handle time-sensitive and legally sensitive tasks. If the system corrupts evidence, fails mid-operation, or cannot quickly recover from faults (like power surges or failed uploads), it can compromise the entire investigation. By ensuring reliable operation with data validation, auto-recovery, and fault isolation, the platform can maintain investigator trust and deliver consistent performance, even under stress or partial system failure.

Stimulus Source
Power surges, corrupted backups, transmission errors, internal component failures
Stimulus
A failure occurs during evidence upload, access, storage, or collaboration
Response
The system prevents data corruption, recovers from failure, and resumes operation
Response Measure

· No loss of uploaded evidence · Auto-recovery time < 30 seconds · Error logged · Has a mean time between failures of at least 1 week
Environment
While the system is actively being used in an ongoing investigation
Artefact
Evidence database, file storage system, logging mechanism

Architectural Strategy(s):

Data validation and integrity checks:

These techniques ensure data is correct and unaltered during upload, storage, and access. AEGIS is required to prevent data corruption and maintain evidence integrity even during transmission errors or component failures. Validation (e.g., file type checks, hashing) detects anomalies, and integrity checks (e.g., checksums, signatures) ensure data hasn't been modified or corrupted. In turn reliable evidence handling, a cornerstone requirement in forensic systems is accomplished.

Backup systems and transactional operations:

Transactions allow the system to restore to avoid partial or inconsistent states. The system must avoid any data loss or corruption—even during power surges or interrupted uploads. Transactional operations (e.g., ACID-compliant database operations) ensure that incomplete actions don't corrupt the system. On failure, the system can rollback or restore a previous valid state.

Architectural pattern:

Repository:

The Repository pattern centralizes access to data and manages interactions with the underlying storage. The reliability target involves no evidence loss and consistent data handling across the file storage and logging systems.Repositories abstract data access and can include validation, transaction control, and failover support. This makes it easier to

enforce consistency and data recovery logic. This ensures consistent, reliable access to and modification of data, which is foundational for forensic reliability.

Architectural Constraints

Architectural constraints define non-negotiable design boundaries that the AEGIS system must respect due to compliance, technical environment, and project limitations.

1. The system must handle sensitive forensic evidence, so it must meet legal and regulatory compliance.
    a. There is a compliance constraint to data privacy laws i.e., POPIA, may prohibit storing sensitive evidence off-site, pushing for on-premises hosting.

2. The chain of custody must be enforced, requiring tamper-proof logs and documented data handling.
    a. Chain-of-custody enforcement demands immutability and full traceability. You can't just overwrite data or delete logs.

3. It must run in an on-site deployment environment, not the public cloud.

4. The system uses PostgreSQL, MongoDB, IPFS, and Docker for all components.
    a. IPFS doesn't support standard database features like transactions. This means we can’t rely on it to safely undo or roll back operations. So, we must make sure that actions like uploading evidence are repeatable without causing problems, and we must check that files haven't been changed, using things like hashes.

    b. Since we’re using Docker to run everything in containers, we have to choose tools that work well in containers. Docker also means we need to plan carefully for how services will talk to each other, how they will be scaled up or down, and how we’ll restart them automatically if something fails.