

PROJECT REPORT OF SYSTEM SECURITY



UNIVERSITA DEGLI STUDI DI MESSINA

MIFT DEPARTMENT

DEICMS

Digital Evidence & Integrity Case Management System

Prof. MASSIMO VILLARI

VIDHYA HARINI VIJAYARAJAN SANGEETHA

533994

NUTHAN PULI

540214

ACADEMIC YEAR – 2023/2024

Table of Contents

1. Introduction	4
1.1 Project Background	4
1.2 Problem Statement.....	4
1.3 Objectives.....	4
1.4 Scope	5
2. System Overview	5
2.1 What the System Does	5
2.2 Technology Stack.....	5
2.3 Architecture	5
3. Database Design	6
4. Authentication and Session Management	7
5. Role-Based Access Control (RBAC)	8
6. Cryptographic Foundations	9
6.1 SHA-256 File Hashing	9
6.2 Ed25519 Digital Signatures	9
7. Evidence Lifecycle State Machine.....	9
8. Chain-of-Custody Graph Integrity Engine.....	10
9. Multi-Factor Risk Scoring	11
10. Audit Formula Checks	12
11. AI-Assisted Anomaly Detection	13
12. AI Investigation Assistant	13
13. Immutable Audit Trail	14
14. Security Enhancements	14
14.1 CSRF Protection.....	14
14.2 Login Rate Limiting and Account Lockout.....	14
14.3 Admin-Only Account Creation.....	15
14.4 Case-Level Access Control.....	15
14.5 Magic-Byte File Validation	16
14.6 Tamper-Evident Case Export.....	16
14.7 Supporting Hardening and Schema Migration.....	17
14.8 Transport Security with HTTPS (TLS).....	17
14.9 Password Strength Policy	18
14.10 Idle Session Timeout	19
14.11 Two-Factor Authentication (2FA via TOTP)	19
14.12 Encrypted Private Keys at Rest	20

14.13 Evidence File Encryption at Rest (AES-256-GCM)	20
15. Dashboard and Security Visualisations	20
16. Application Walkthrough: Pages of the Website	20
16.1 Login Page	21
16.2 Dashboard	21
16.3 Cases List.....	22
16.4 Create Case	23
16.5 Case Detail.....	23
16.6 Edit Case	23
16.7 Upload Evidence	24
16.8 Evidence Detail.....	24
16.9 Transfer Custody	25
16.10 Change Lifecycle State.....	25
16.11 Custody Log	25
16.12 Integrity Verification Report.....	26
16.13 Chain Reconstruction Report.....	26
16.14 Risk Dashboard	27
16.15 Evidence Risk Report	28
16.16 Anomaly Detection Dashboard	28
16.17 Audit Trail	29
16.18 Evidence Audit Trail.....	29
16.19 AI Investigation Assistant.....	30
16.20 Administration — User Management	30
16.21 Administration — Create Investigator.....	31
16.22 Security (2FA) Page	31
16.23 Two-Factor Verification Page.....	32
17. Security Concepts Demonstrated	32
18. Limitations and Future Work.....	33
19. Conclusion	33

Note: if the page numbers below do not appear, open the document in Word, select all (Ctrl+A) and press F9 to update this table.

1. Introduction

1.1 Project Background

For our System Security project, we designed and built DEICMS — the Digital Evidence and Integrity Case Management System. ***DEICMS is a web application that simulates the platform a digital forensics laboratory would use to manage investigation cases and the digital evidence attached to them.*** We chose this domain because forensic evidence handling has *usually strict integrity, accountability, and access requirements*, and these requirements map almost one-to-one onto the core concepts of information security: ***confidentiality, integrity, availability, authentication, authorisation, and non-repudiation.*** By building a realistic system around evidence handling, we were able to apply a broad range of security mechanisms in a single, coherent project.

1.2 Problem Statement

When digital evidence is collected, analysed, and moved between investigators, every interaction must be recorded in a way that cannot later be disputed. If a single file is altered, or if a custody record is back-dated or forged, the entire case can collapse in court. The problem we set out to solve was therefore twofold: *first, to make tampering with evidence or its history detectable through cryptography*; and *second, to ensure that only the right people can perform the right actions, and that every action is permanently recorded.* On top of this, the application itself had to be hardened against the common web attacks that could be used to bypass those controls.

1.3 Objectives

We set the following objectives for the project:

- To enforce strong authentication and role-based authorisation so that users can only perform actions appropriate to their role.
- To guarantee the integrity of every evidence file using cryptographic hashing, and the authenticity of every custody event using digital signatures.
- To maintain an immutable, tamper-evident audit trail of every significant action in the system.
- To detect anomalous or high-risk evidence automatically using a multi-factor risk model and machine learning.
- To harden the web application against brute-force attacks, cross-site request forgery, privilege escalation, and malicious file uploads.
- To present all of this information through a clear dashboard with meaningful security visualisations.

1.4 Scope

We implemented DEICMS as **a Flask web application backed by a relational database**. The project covers user management, case management, evidence upload and lifecycle handling, chain-of-custody recording and verification, risk scoring, anomaly detection, an AI assistant, a full audit trail, and a suite of web-application security controls. **We run the system on port 5001.**

2. System Overview

2.1 What the System Does

DEICMS lets an investigator log in, open or be granted access to investigation cases, upload digital evidence files, and move that evidence between investigators while the system records every handoff. *For each piece of evidence, the system computes a cryptographic fingerprint, tracks its position in a formal lifecycle, calculates a risk score, and can verify the complete chain of custody on demand.* Administrators manage user accounts and roles, and every login, file access, state change, and failed attempt is written to an append-only audit log.

2.2 Technology Stack

We built the system with the following technologies:

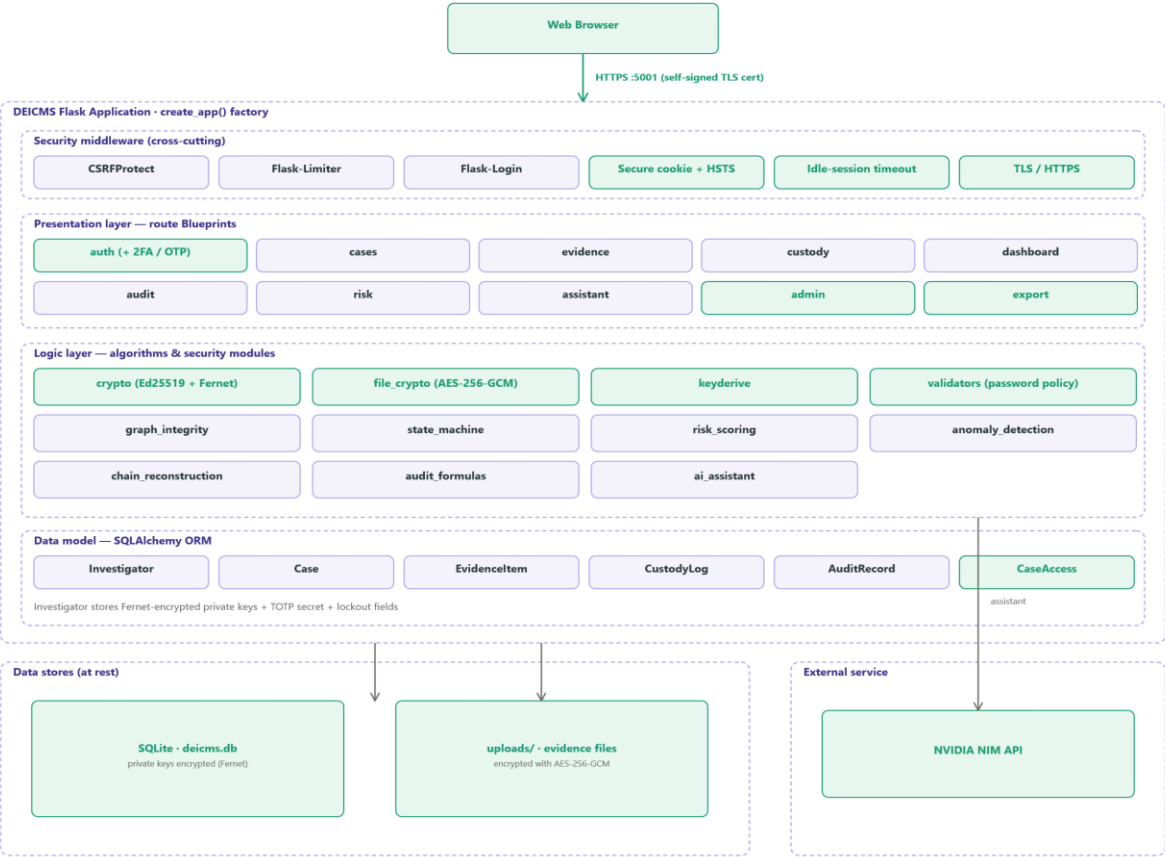
- Python and the Flask web framework, structured using the application-factory pattern and Blueprints.
- SQLAlchemy as the ORM, with SQLite as the development database.
- Flask-Login for session management, and Werkzeug (PBKDF2) for password hashing.
- The cryptography library for Ed25519 digital signatures and SHA-256 hashing.
- Flask-WTF for CSRF protection and Flask-Limiter for rate limiting.
- python-magic for true file-type detection, Pillow for image metadata, and scikit-learn for anomaly detection.
- Bootstrap 5 and Chart.js for the user interface and dashboard visualisations.

2.3 Architecture

We organised the code into three clear layers. *The models layer defines the database tables. The routes layer contains the Flask Blueprints that handle HTTP requests. The logic layer holds the non-trivial algorithms — cryptography, the state machine, the graph integrity engine, risk scoring, and anomaly detection — kept separate from the request handling so that the security-critical logic is isolated and testable.* The application factory wires the extensions, registers the Blueprints, and creates the database tables on start-up.

Component & Architecture Diagram — DEICMS

Green components were added for the new security features.



3. Database Design

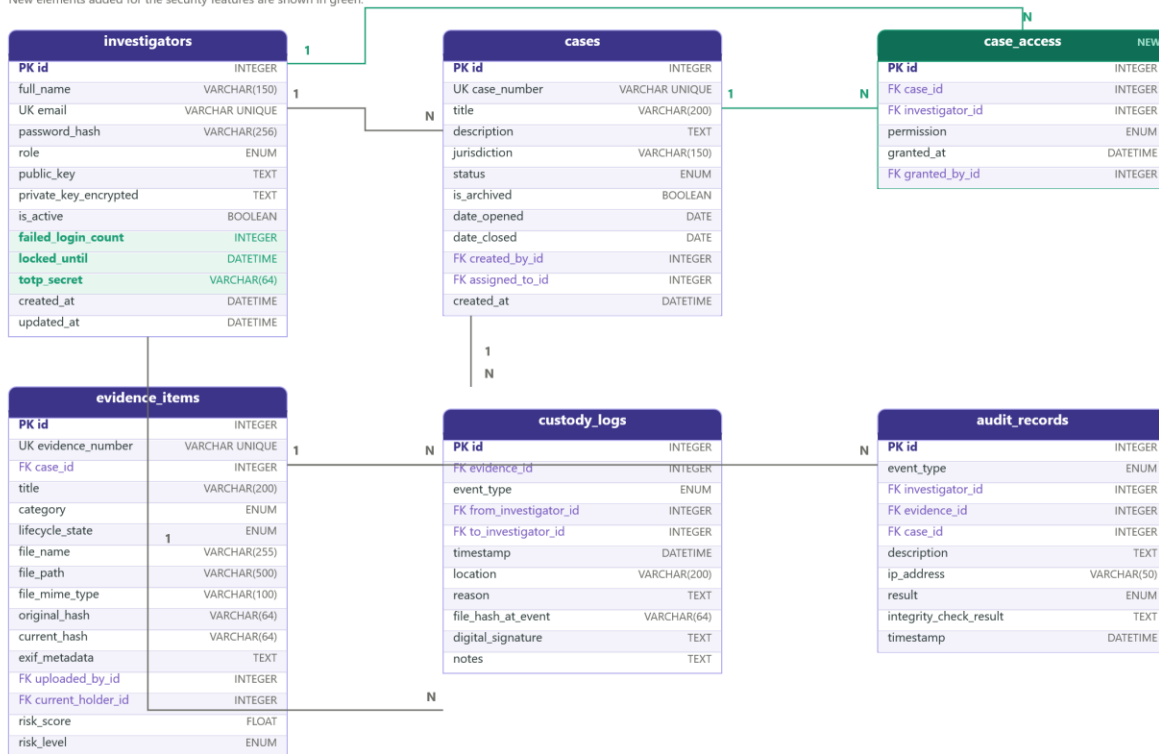
We modelled the system around six database tables. The design deliberately preserves history: *accounts and records are deactivated rather than deleted*, so that the *audit trail and custody chain can never be broken by a deletion*.

Model	Purpose
Investigator	A user account, with a role, a PBKDF2 password hash, an Ed25519 key pair, and login-lockout fields.
Case	An investigation, with a status lifecycle (Open, Active, Pending, Closed, Archived).
EvidenceItem	A single digital file linked to a case, storing its path, original and current SHA-256 hashes, metadata, lifecycle state, and risk score.
CustodyLog	A write-once record of every custody event (Upload, Transfer, Access, Verification), with the hash at that moment and an Ed25519 signature.
AuditRecord	A write-once log of every significant system event, including the IP address of the client.
CaseAccess	Grants a specific investigator a permission level (Owner, Member, ReadOnly) on a specific case.

We designed the relationships so that evidence belongs to a case, custody logs and audit records belong to evidence and cases, and case access links investigators to cases. The diagram below shows the full entity-relationship model.

Entity-Relationship Diagram — DEICMS

New elements added for the security features are shown in green.



■ New (security features)

PK = primary key FK = foreign key UK = unique key 1—N = one-to-many

4. Authentication and Session Management

We implemented *authentication* using *Flask-Login*. **Passwords are never stored in plaintext**; instead, we *hash* them with Werkzeug's PBKDF2-based `generate_password_hash`, and *verify* them with a *constant-time comparison* on login. Once a user authenticates successfully, Flask-Login stores their user ID inside a *cryptographically signed session cookie*, so the session cannot be forged on the client side.

We hardened the session configuration as well. We **set the session cookie to *HttpOnly* so that client-side JavaScript cannot read it** (mitigating session theft through cross-site scripting), and we **set the *SameSite* attribute to *Lax* so the cookie is not sent on cross-site requests**, which further reduces cross-site request forgery exposure. Every successful login and every failed login is written to the audit trail together with the client IP address, giving a complete record of authentication activity.

5. Role-Based Access Control (RBAC)

We **implemented role-based access control with four roles**, each with a **clearly defined set of permissions**. We added helper methods on the Investigator model — such as `is_admin()`, `can_manage()`, and `can_analyse()` — and we call these throughout the route handlers so that every sensitive action is gated by the user's role.

Role	What we allow this role to do
Admin	Full access: manage all users and roles, all cases, archive cases, delete evidence, and export packages.
Lead Investigator	Create and manage cases and evidence, sign off on state transitions, and export case packages.
Analyst	View and analyse evidence on cases they are assigned to.
Read-Only	View permitted cases and evidence, with no ability to modify anything.

Because role alone is a coarse control, we also implemented case-level access control (described in Section 14.4) so that access depends not only on a user's role but on whether the investigator is a member of the specific case — the principle of least privilege.

Use-Case Diagram — DEICMS

Green use cases were added for the new security features. Higher roles inherit the capabilities of lower roles.



6. Cryptographic Foundations

*Cryptography is the backbone of the system's integrity guarantees, so we built a dedicated cryptographic layer. We use two primitives: **SHA-256 hashing for file integrity**, and **Ed25519 digital signatures for authenticity and non-repudiation**.*

6.1 SHA-256 File Hashing

When an evidence file is uploaded, we compute its SHA-256 hash by streaming the file in chunks, and we store this as the original hash — the immutable ground truth for that file. At any later point, the system can re-read the file from disk, recompute the hash, and compare it against the original. If even a single byte has changed, the hashes will not match, and the tampering is detected immediately.

6.2 Ed25519 Digital Signatures

*We generate an Ed25519 key pair for every investigator when their account is created. When an investigator transfers custody of an evidence item, we build a **canonical representation of the custody event** — **the evidence ID, event type, timestamp, parties involved, and the file hash** — **and sign it with the investigator's private key**. The resulting signature is stored on the custody record. Anyone holding the investigator's public key can later verify that the event was genuinely authorised by that investigator and has not been altered. This provides both authenticity and non-repudiation: a signer cannot later deny having authorised a transfer.*

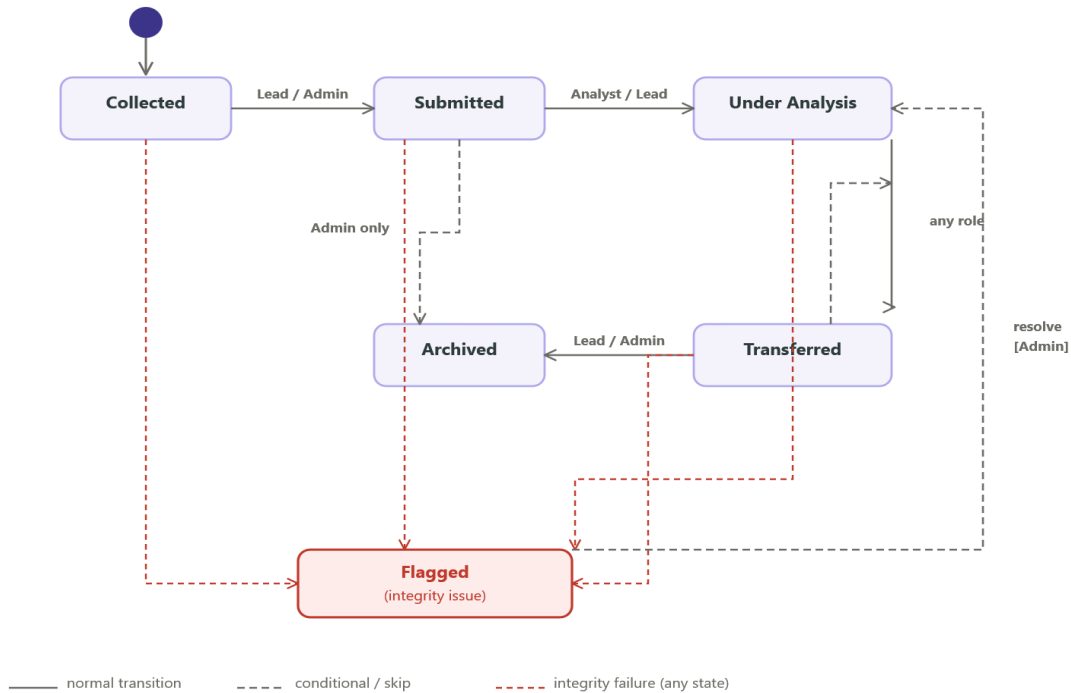
7. Evidence Lifecycle State Machine

*We modelled the legally meaningful lifecycle of an evidence item as a formal finite state machine. **Evidence moves through the states Collected, Submitted, Under Analysis, Transferred, Archived, and Flagged**. We defined a transition table that specifies exactly which transitions are allowed, which roles may perform each transition, and which preconditions must hold first.*

When a transition is requested, we look up the (current state, target state) pair, check that the investigator's role is permitted, and evaluate the preconditions — for example, requiring that an integrity check has passed before evidence may be archived. If every check passes, we update the state, write a success audit record, and recalculate the risk score; if any check fails, we write a failure audit record and return a descriptive error. This guarantees that evidence can never reach an invalid or unauthorised state.

State Diagram — Evidence Lifecycle

Role-gated transitions. Red dashed arrows indicate integrity failure from any active state.



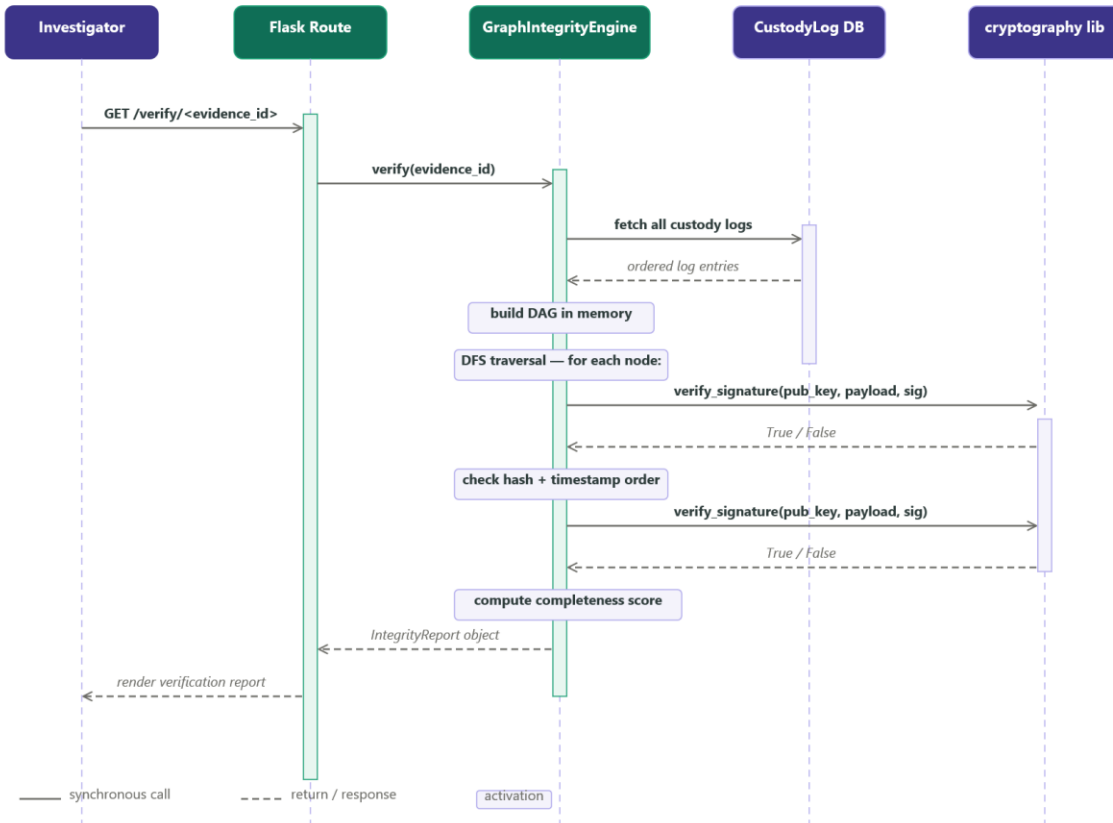
8. Chain-of-Custody Graph Integrity Engine

This is the most sophisticated security component we built. We model the ***complete custody history of an evidence item as a directed acyclic graph***, and we walk it with ***a depth-first search, performing three independent checks at every node***.

The engine produces an integrity report containing a *completeness score (the percentage of nodes that passed)*, an overall verdict of *Intact, Partially Verified, or Broken*, and a detailed list of any per-node failures. If the verdict is *Broken*, we automatically move the evidence into the *Flagged* state and write an audit record, so a compromised item cannot quietly continue through the workflow.

Sequence Diagram — Graph Integrity Verification

Message flow: Investigator → Flask route → GraphIntegrityEngine → CustodyLog DB → cryptography lib



9. Multi-Factor Risk Scoring

We built **a multi-factor risk scorer that assigns each evidence item a risk value between 0 and 1**. We combine six weighted factors, each normalised to a comparable scale, so that no single factor dominates unfairly.

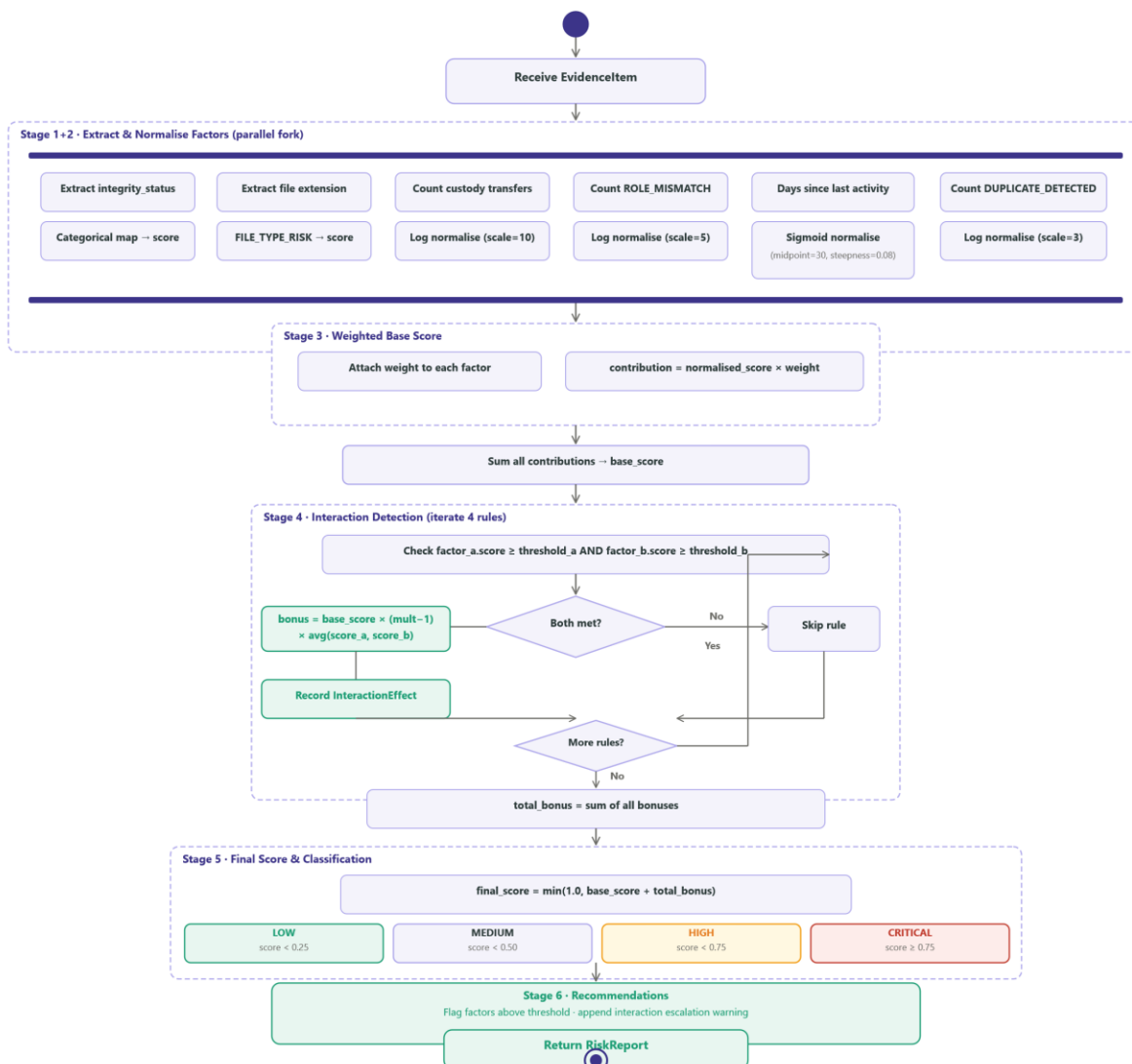
Factor	Weight	How we normalise it
Integrity status (verified / not checked / failed)	30%	Categorical lookup
Inherent file-type risk (e.g. .exe vs .txt)	20%	Categorical lookup
Custody transfer frequency	15%	Logarithmic
Role-mismatch count from audit records	15%	Logarithmic
Time since last activity	10%	Sigmoid (rises sharply past 30 days)
Duplicate custody entries	10%	Logarithmic

After computing the weighted base score, we apply interaction rules that add a bonus when two dangerous factors both cross their thresholds — for example, an integrity failure combined with

role mismatches. We then map the final score to one of four bands: Low, Medium, High, or Critical. This lets an investigator immediately see which evidence items deserve the most scrutiny.

Activity Diagram — Multi-Factor Evidence Risk Scoring

Six factors extracted, normalised, weighted, interaction-adjusted, classified, and annotated with recommendations.



10. Audit Formula Checks

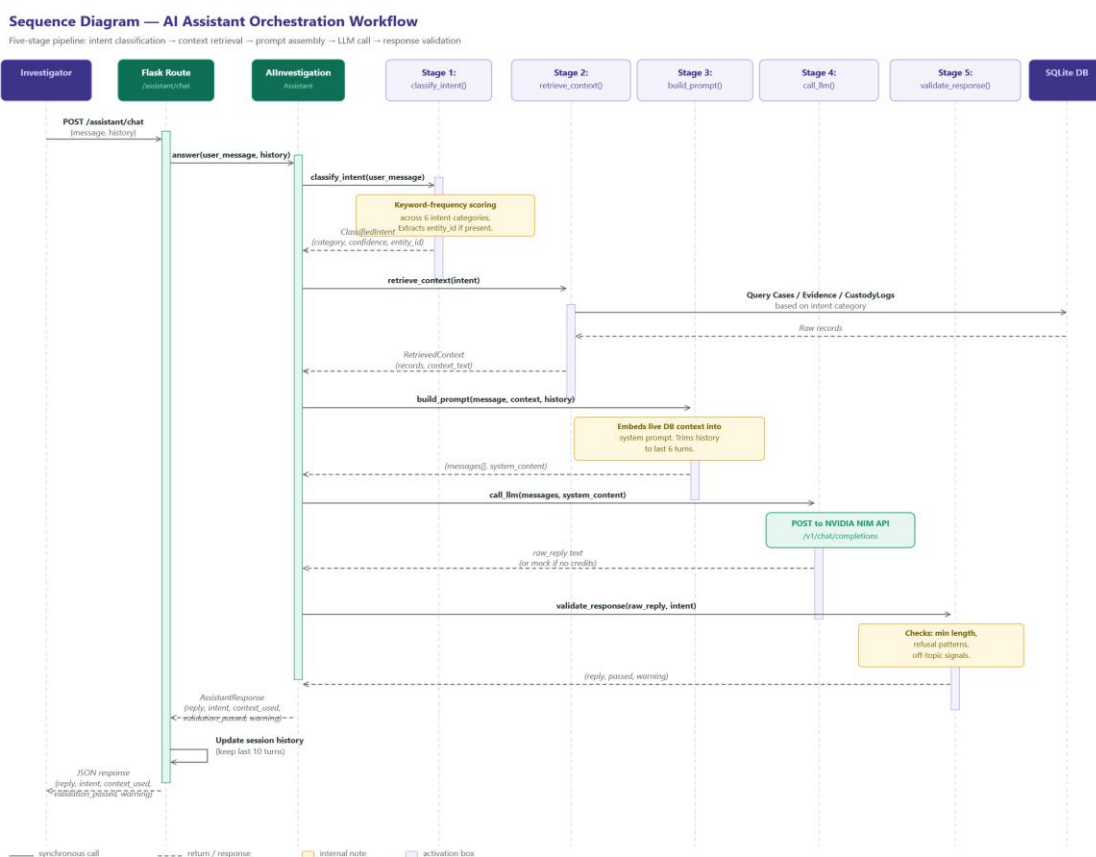
Alongside the risk score, we **implemented three lightweight formula-based checks that can be run on demand to surface suspicious patterns**. The first is **time-gap detection**, which flags any custody gap longer than a configurable threshold (72 hours by default). The second is **duplicate-entry detection**, which finds custody records that share an identical event type, source, destination, and timestamp. The third is **role-mismatch detection**, which inspects the audit records for file accesses performed by investigators whose role is not permitted for that category of evidence. Each check contributes evidence to the risk model and to the anomaly detector.

11. AI-Assisted Anomaly Detection

To catch unusual evidence that the fixed rules might miss, we added an unsupervised machine-learning anomaly detector built on scikit-learn's Isolation Forest. We represent each evidence item as a six-dimensional feature vector — transfer count, transfer rate, number of unique handlers, the largest gap between transfers, an integrity flag, and the inherent file-type risk. The Isolation Forest learns what 'normal' looks like across all evidence and assigns an anomaly score to each item. We invert and normalise these scores so that 1.0 represents the most anomalous item, and we flag anything scoring at or above 0.6. This gives the investigator a data-driven second opinion that complements the deterministic risk score.

12. AI Investigation Assistant

We added a natural-language assistant that lets investigators ask questions about their cases, evidence, risk scores, and custody chains. The assistant is given forensic context drawn from the live database and answers grounded in the real records. We treated the assistant as a security-sensitive surface as well: its asynchronous requests carry the same CSRF protection as the rest of the application (described in Section 14.1), so they cannot be triggered by a forged cross-site request. The conversation is held in the user's session and is replayed when they return to the page, so it persists as they move around the application and is cleared only when they press the Clear Chat button.



13. Immutable Audit Trail

Every significant action in the system — logins, logouts, file accesses, data modifications, integrity checks, state changes, failed attempts, role changes, and key generation — is written to an append-only audit table. We record the event type, the responsible investigator, the related case and evidence, a human-readable description, the client IP address, the outcome (Success, Failure, or Warning), and a timestamp. Because **records are only ever inserted and never updated or deleted, the trail provides non-repudiation**: it is **a permanent, reviewable history of who did what, when, and from where**. **Access to the full system trail is restricted to administrators**; lead investigators and analysts can still review the per-evidence trail for cases they are members of.

14. Security Enhancements

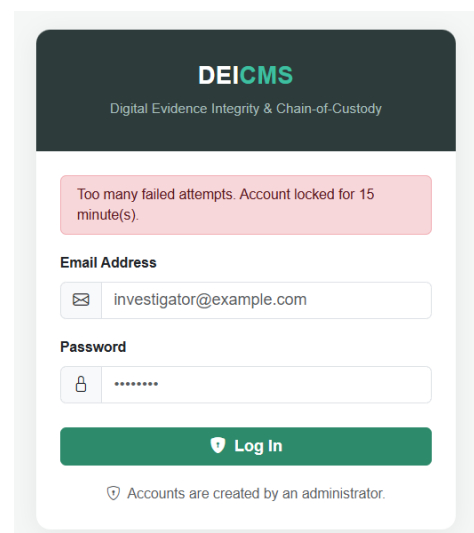
This section **describes the web-application security controls we built into the system**. These are the controls most directly relevant to System Security, and together they defend against the common attacks that could otherwise be used to bypass the integrity and access mechanisms described above.

14.1 CSRF Protection

To defend against cross-site request forgery, we **enabled application-wide CSRF protection using Flask-WTF**. We added a *hidden CSRF token to every form that submits data*, and we *exposed the token through a meta tag* so that the *JavaScript-driven requests in the AI assistant can send it in an X-CSRFToken header*. With this in place, **any state-changing request that does not carry a valid, server-issued token is rejected**. We verified this by submitting a form without a token and confirming the server responded with HTTP 400.

14.2 Login Rate Limiting and Account Lockout

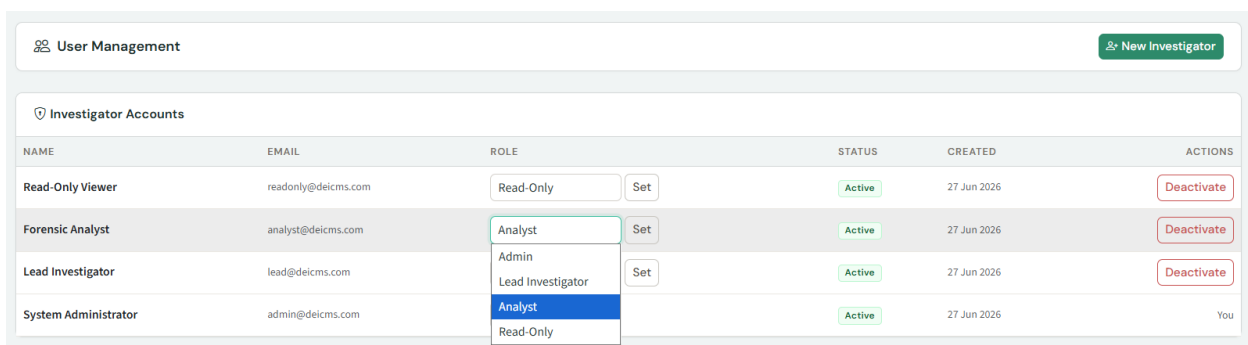
To defend against brute-force password guessing, we implemented two complementary controls. First, we used **Flask-Limiter to cap the login endpoint at ten requests per minute per IP address**. Second, we added **per-account lockout: we track the number of consecutive failed logins, and after five failures we lock the account for fifteen minutes**. We added `failed_login_count` and `locked_until` columns to the Investigator model, helper methods to record failures and clear them, and a clear lockout message on the login page. Every lockout is written to the audit trail. We also **gave administrators the ability to unlock an account manually**. We verified that the account locks after five wrong passwords, that the rate limiter returns HTTP 429 once the limit is exceeded, and that an administrator can clear the lockout.



The screenshot shows the DEICMS login interface. At the top, the header reads "DEICMS" with the subtitle "Digital Evidence Integrity & Chain-of-Custody". A pink error message box states: "Too many failed attempts. Account locked for 15 minute(s)". Below this, there are input fields for "Email Address" (containing "investigator@example.com") and "Password" (masked with "*****"). A green "Log In" button is positioned below the password field. At the bottom, a small note says "Accounts are created by an administrator."

14.3 Admin-Only Account Creation

We made **account creation an administrator-only action to prevent privilege escalation**. If users could register themselves and choose their own role, anyone could simply grant themselves administrator rights, so we designed the system so that new accounts can only be created from an administrator-only user-management area. **We built an admin panel where administrators can list all accounts, create new investigators, activate or deactivate accounts, unlock locked accounts, and change roles**. We protected every one of these routes with an administrator guard, and we placed an ‘Admin Panel’ link in the navigation that is visible only to administrators. As a result, **the only path to a new account is through an administrator, which closes off self-service privilege escalation**. We verified that a non-administrator is denied access to these routes while an administrator can manage accounts normally.

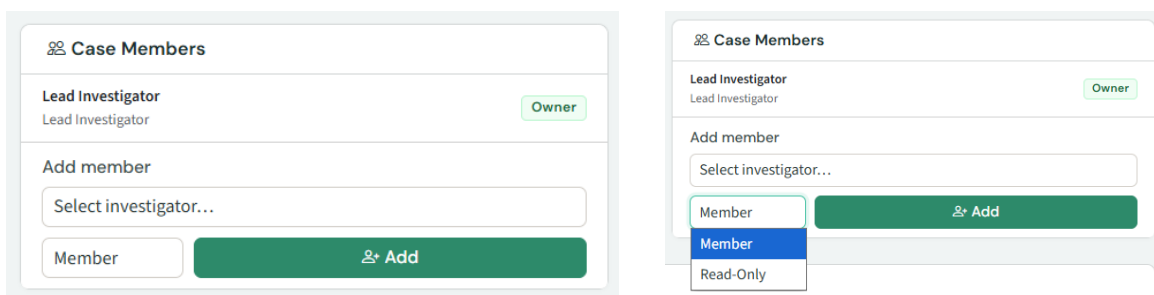


The screenshot shows the 'User Management' interface. At the top right is a 'New Investigator' button. Below is a section titled 'Investigator Accounts' containing a table with columns: NAME, EMAIL, ROLE, STATUS, CREATED, and ACTIONS. The table lists four accounts: Read-Only Viewer, Forensic Analyst, Lead Investigator, and System Administrator. The 'ROLE' column has a dropdown menu open for the 'Forensic Analyst' row, showing options: Analyst, Admin, Lead Investigator, and Read-Only. The 'STATUS' column shows 'Active' for all accounts. The 'ACTIONS' column has 'Deactivate' buttons for the first three accounts and 'You' for the System Administrator.

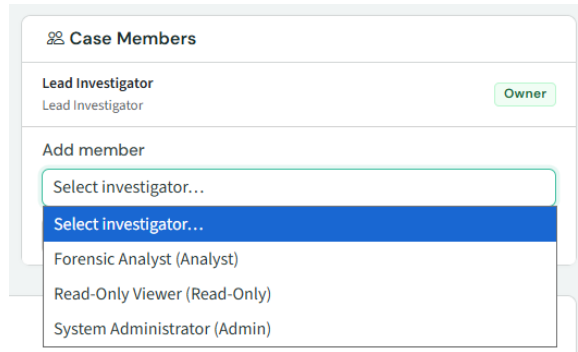
NAME	EMAIL	ROLE	STATUS	CREATED	ACTIONS
Read-Only Viewer	readonly@deicms.com	Read-Only	Active	27 Jun 2026	Deactivate
Forensic Analyst	analyst@deicms.com	Analyst	Active	27 Jun 2026	Deactivate
Lead Investigator	lead@deicms.com	Admin	Active	27 Jun 2026	Deactivate
System Administrator	admin@deicms.com	Lead Investigator	Active	27 Jun 2026	You

14.4 Case-Level Access Control

To enforce the principle of least privilege, we **added case-level access control on top of the global roles**. We created a *CaseAccess* table that grants an investigator a permission level — Owner, Member, or ReadOnly — on a specific case. When an investigator creates a case, we automatically make them its Owner. **The case list now shows administrators every case but shows everyone else only the cases they have been granted access to**. We added access checks to the case, evidence, custody, and export routes so that a user who is not a member of a case cannot view or act on it, and we **built a ‘Case Members’ panel where an owner or administrator can add and remove members**. To keep existing data usable, we wrote an idempotent start-up migration that backfills an Owner record for every pre-existing case. We verified that a non-member is denied access to a case and that, once added as a member, the same user gains access.



The two screenshots show the 'Case Members' panel. The left screenshot shows a 'Lead Investigator' as the 'Owner' with an 'Add member' section containing a search input and an 'Add' button. The right screenshot shows the same panel with a dropdown menu open for the 'Add member' section, showing options: Member, Member, and Read-Only. The 'Add' button is also visible.



14.5 Magic-Byte File Validation

Checking only a file's extension is insufficient, because an attacker can rename a malicious executable to look like an image. To prevent this MIME-type spoofing, we **added magic-byte validation to the upload handler using python-magic**. After a file is saved, we read its real content type from its leading bytes and compare it against the declared extension. **We reject any file whose true type does not match its extension, and we reject executables outright regardless of their name. Rejected uploads are deleted from disk and logged to the audit trail as failed attempts.** We verified this by renaming an executable to a .jpg and confirming it was rejected, while a genuine text file uploaded successfully.

14.6 Tamper-Evident Case Export

For legal admissibility we built a tamper-evident export feature. When an administrator or lead investigator exports a case, we **generate a ZIP archive in memory containing all of the case's evidence files, a manifest listing the SHA-256 of every file together with case metadata, a full text audit trail for the case, and an Ed25519 signature of the manifest produced with the exporting user's private key**. A recipient can independently verify the package by re-hashing each file against the manifest and verifying the manifest's signature against the exporter's public key. **Any modification to the package after export is therefore detectable.** We verified that the export downloads a valid ZIP containing the manifest, the signature file, and the audit trail.

Desktop > CASE-2026-008_export_20260627_163834							
<div> <div> <div></div> <div></div> <div></div> <div></div> </div> <div>Sort</div> <div>View</div> <div>Extract all</div> <div></div> </div>							
Name	Type	Compressed size	Password p...	Size	Ratio	Date modified	
audit_trail	Text Document	1 KB	No	1 KB	34%	6/27/2026 06:38 PM	
manifest	JSON Source File	1 KB	No	1 KB	43%	6/27/2026 06:38 PM	
manifest_signature	Text Document	1 KB	No	1 KB	20%	6/27/2026 06:38 PM	

14.7 Supporting Hardening and Schema Migration

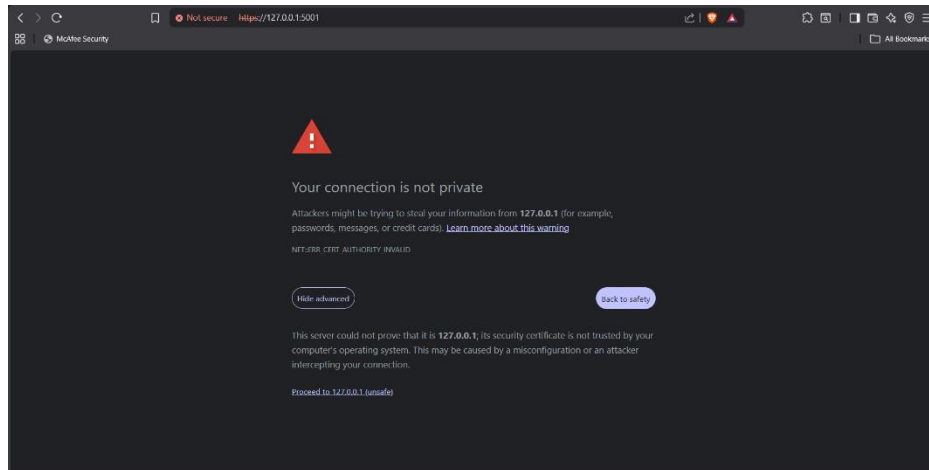
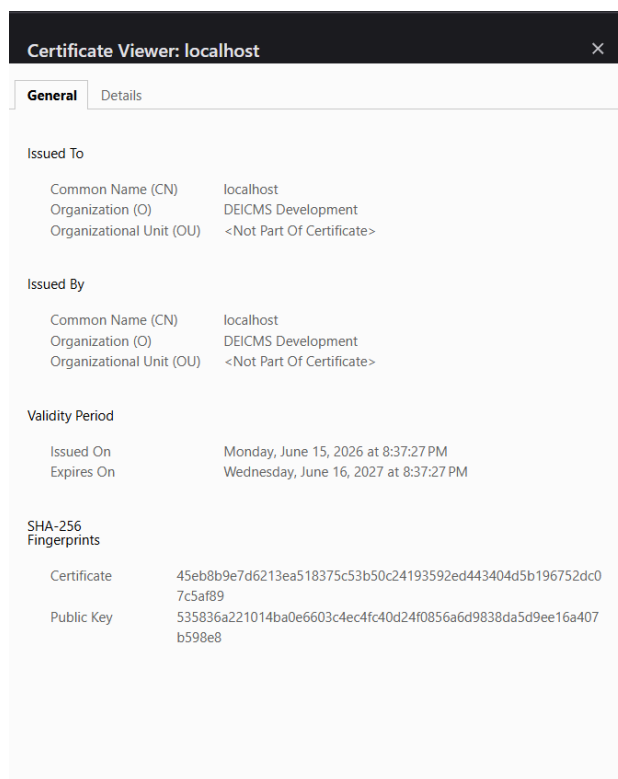
Alongside the major controls we added several supporting measures. We **configured secure session cookies (HttpOnly and SameSite), and centralised the security settings — maximum login attempts, lockout duration, and the MIME-to-extension map** — in the configuration file. Because adding new security columns to an existing database can break older data, we wrote a small, idempotent migration that runs automatically at start-up: it **adds the lockout columns to the user table if they are missing and backfills case ownership, so the system upgrades itself safely without losing the existing seeded data.**

14.8 Transport Security with HTTPS (TLS)

Without transport-layer security, the traffic between the browser and the server would travel over plain HTTP, which would mean that sensitive data — including the password typed at login and the session cookie that authenticates every subsequent request — would be transmitted unencrypted, and anyone able to observe the network could read or tamper with it (a classic man-in-the-middle exposure). To prevent this, we **served the whole application over HTTPS.**

We implemented this without adding any new libraries. We wrote a small script, **generate_cert.py, that uses the cryptography library already in the project to create a 2048-bit RSA key and a self-signed X.509 certificate for localhost.** We then changed `run.py` so that it generates this certificate automatically if it is missing and starts Flask with an SSL context, serving the application over TLS at **`https://localhost:5001`**. We **also set the session cookie's Secure attribute, so the cookie is only ever transmitted over an encrypted connection,** and we added a **Strict-Transport-Security (HSTS) response header so that once a browser has connected securely it will not silently fall back to plain HTTP.**

After this change the application is reached at `https://localhost:5001`, and the login credentials and session cookie are encrypted in transit. The session cookie is now flagged Secure in addition to HttpOnly and SameSite, and every secure response carries the HSTS header. Because the certificate is self-signed rather than issued by a trusted Certificate Authority (CA), the browser shows a one-time **'your connection is not private' warning** on the first visit; this is expected for local development, and in a real deployment we would replace the self-signed certificate with one issued by a trusted authority such as Let's Encrypt. We verified the change by serving the application over TLS and confirming that the login page loads over HTTPS, that the response includes the Strict-Transport-Security header, and that the session cookie is marked Secure.



14.9 Password Strength Policy

We added a password-strength policy so that weak passwords cannot be chosen. When an account is created, we check the chosen password against **a set of rules — a minimum length of twelve characters and at least one uppercase letter, one lowercase letter, one digit, and one special character** — and we reject it with specific feedback if any rule is not met. Enforcing complexity makes passwords far harder to guess or brute-force. We verified that a weak password is rejected while a strong one is accepted.

14.10 Idle Session Timeout

We **added an idle session timeout so that an unattended session does not stay logged in indefinitely**. On every request we record the time of the user's last activity, and **if more than thirty minutes pass without any activity we log the user out, clear the session, and return them to the login page with an explanatory message**. This limits the window in which an abandoned but still logged-in browser could be misused. We verified that a session whose activity time is stale is logged out on its next request.

14.11 Two-Factor Authentication (2FA via TOTP)

We added **time-based one-time-password (TOTP) two-factor authentication (2FA)** so that a stolen password alone is not enough to sign in. **A user enrolls from a security page that shows a QR code; after scanning it with an authenticator app such as Google Authenticator and confirming a code, we store their TOTP secret**. From then on, once the user has entered a correct password, we hold them in a pending state and require the six-digit code from their app before completing the login. We **used the pyotp library to generate and verify the codes and the qrcode library to render the enrolment QR code**. We verified that a user with 2FA enabled is sent to the code page after their password, that an incorrect code is rejected, and that the correct code completes the login.

```
def mfa_setup():
    """Enrol the current user in TOTP multi-factor authentication."""
    import pyotp, qrcode, io, base64

    if request.method == 'POST':
        secret = session.get('mfa_setup_secret')
        code = request.form.get('otp', '').strip().replace(' ', '')
        if secret and pyotp.TOTP(secret).verify(code, valid_window=1):
            current_user.totp_secret = secret
            session.pop('mfa_setup_secret', None)
            audit = AuditRecord(
                event_type='Role Change',
                investigator_id=current_user.id,
                description=f'{current_user.full_name} enabled two-factor authentication (2FA).',
                ip_address=request.remote_addr,
                result='Success'
            )
            db.session.add(audit)
            db.session.commit()
            flash('Two-factor authentication (2FA) is now enabled on your account.', 'success')
            return redirect(url_for('mfa_setup' if False else 'auth.mfa_setup'))
        flash('That code was not valid. Please scan the QR code and try again.', 'danger')

    if current_user.is_mfa_enabled():
        return render_template('auth/mfa_setup.html', enabled=True, qr_data=None, secret=None)

    # Generate a provisional secret and a QR code for enrolment
    secret = session.get('mfa_setup_secret') or pyotp.random_base32()
    session['mfa_setup_secret'] = secret
    uri = pyotp.totp.TOTP(secret).provisioning_uri(
        name=current_user.email, issuer_name='DEICMS')
    img = qrcode.make(uri)
    buf = io.BytesIO()
    img.save(buf, format='PNG')
    qr_data = 'data:image/png;base64,' + base64.b64encode(buf.getvalue()).decode('ascii')
    return render_template('auth/mfa_setup.html', enabled=False, qr_data=qr_data, secret=secret)
```

14.12 Encrypted Private Keys at Rest

Each investigator has an Ed25519 private key that is used to sign custody events, so the confidentiality of those key matters. We encrypt every private key before it is stored in the database, using Fernet authenticated encryption with a key derived from the application secret. The key is decrypted only in memory and only at the moment it is needed to sign an event, so a copy of the database on its own does not reveal any usable private keys. We **added a start-up step that encrypts any keys that were previously held in plaintext**. We verified that no plaintext private key remains in the database and that signing and signature verification still work correctly through the transparent decryption.

14.13 Evidence File Encryption at Rest (AES-256-GCM)

To protect the confidentiality of the evidence itself, we **encrypt every uploaded file on disk using AES-256-GCM, an authenticated cipher that also detects tampering with the stored ciphertext**. When a file is uploaded, we *first compute its SHA-256 hash and validate it, then encrypt the file in place with a key derived from the application secret and a fresh random nonce*. **Wherever the original content is genuinely needed — when verifying integrity or producing a signed export — we decrypt it transparently in memory**. This means that even *if the server's storage were copied, the evidence files would be unreadable*. We verified that an uploaded file is stored as ciphertext on disk, that it decrypts back to the exact original, that integrity verification still returns an Intact verdict, and that the exported package still contains the original files.

15. Dashboard and Security Visualisations

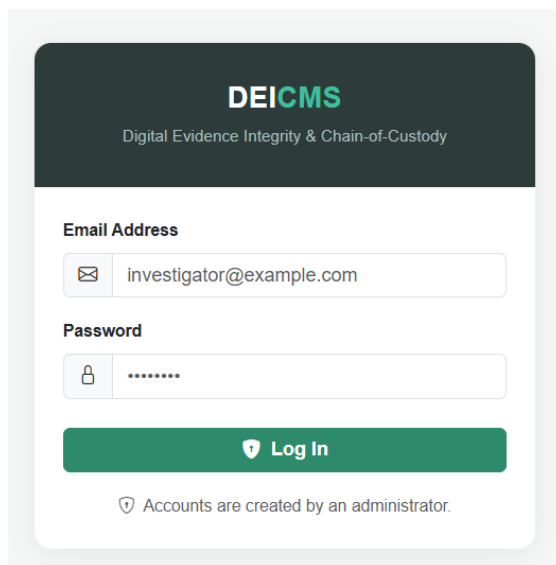
To make the system's security posture visible at a glance, we **built four data visualisations into the dashboard using Chart.js and the application's colour scheme**. The first is a *doughnut chart showing the distribution of evidence by computed risk level*. The second is a *horizontal bar chart of evidence by lifecycle state, with flagged items highlighted in red*. The third is a *line chart of security activity over the last thirty days, plotting successful events against security alerts (failures and warnings) so that spikes in failed logins or warnings stand out*. The fourth is a *bar chart of audit events by type, with failed attempts highlighted*. All four charts read live from the database, so they update as the system is used.

16. Application Walkthrough: Pages of the Website

In this section we describe every page of the DEICMS web interface and what each one allows the user to do. The navigation sidebar is visible on every page once a user has logged in and it adapts to the user's role — for example, the Admin Panel link appears only for administrators.

16.1 Login Page

The login page is the entry point to the system. We present a single form asking for the investigator's email and password. If an account has been locked by repeated failed attempts, a red banner explains that the account is locked and for how long. A note reminds users that accounts are created by an administrator, because we removed open self-registration. The page is served over HTTPS, so the credentials are encrypted as they are submitted.



The login page features a dark header with the DEICMS logo and tagline. Below is a form with fields for Email Address and Password, a Log In button, and a note about administrator-created accounts.

DEICMS
Digital Evidence Integrity & Chain-of-Custody

Email Address
investigator@example.com

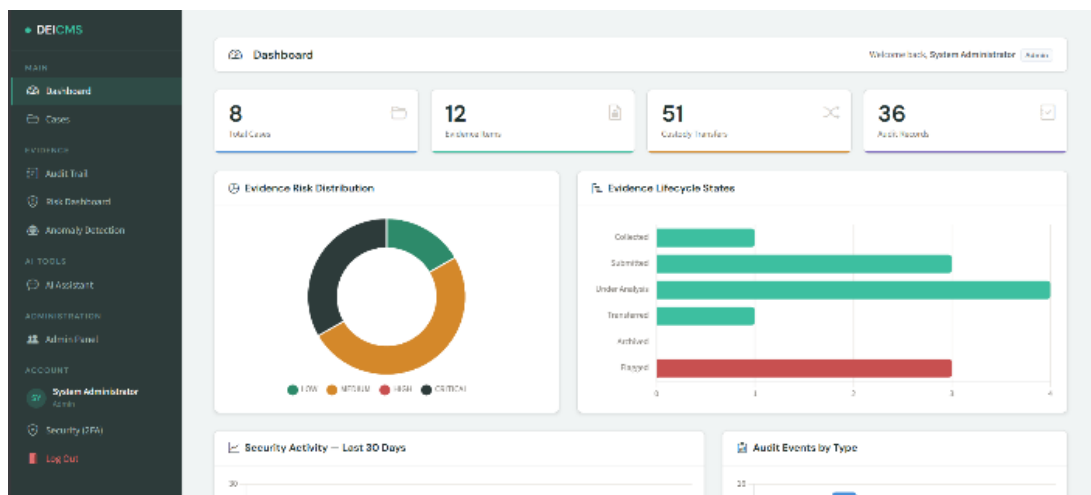
Password

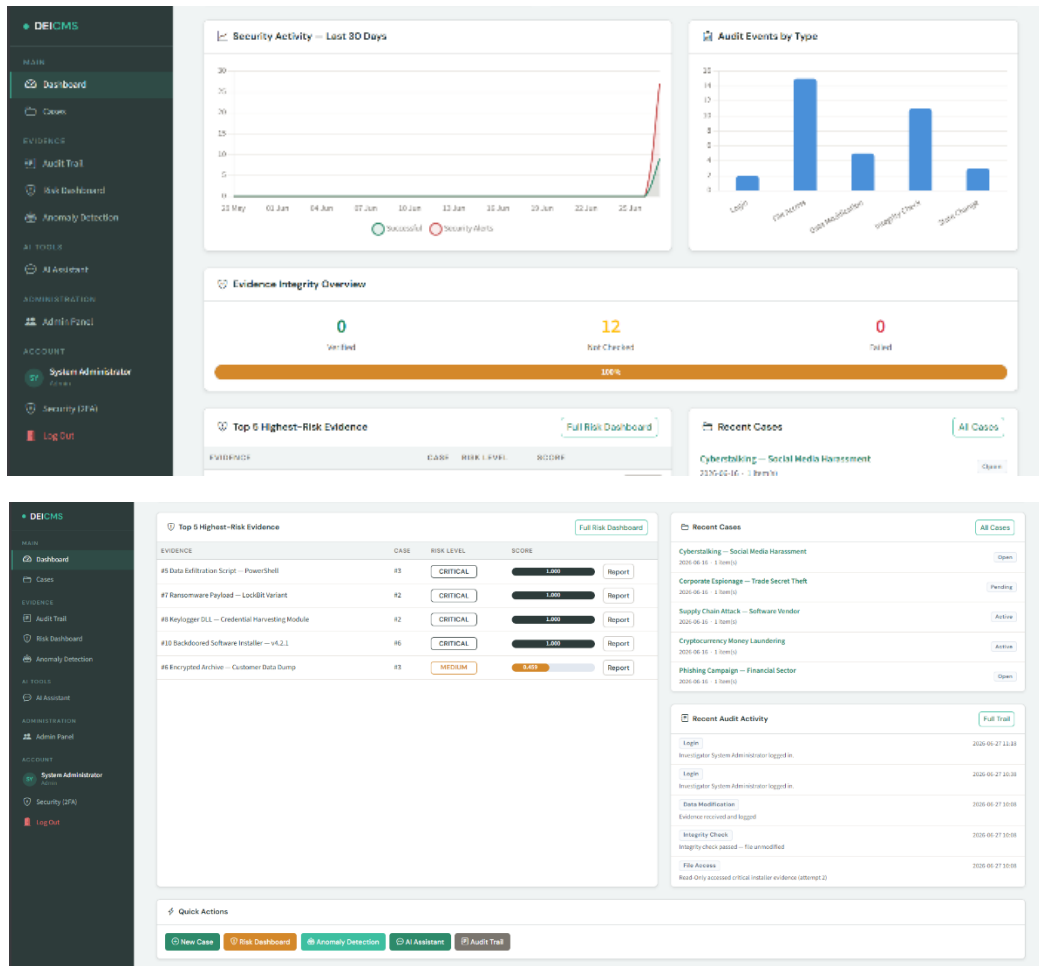
Log In

Accounts are created by an administrator.

16.2 Dashboard

The dashboard is the landing page after login. At the top we show **four summary cards — total cases, evidence items, custody transfers, and audit records**. Below them we added four security visualisations: the evidence risk distribution, the evidence lifecycle states, security activity over the last thirty days, and audit events by type. The page also shows an integrity overview bar, a table of the five highest-risk evidence items, the most recent cases and audit activity, and a row of quick-action buttons.





16.3 Cases List

The cases page lists the investigation cases the user is allowed to see. Administrators see every case, while other investigators see only the cases they have been granted access to, in line with the case-level access control. Each row shows the case number, title, jurisdiction, status, opening date, and evidence count, with a link to open the case.

• DEICMS

MAIN

Dashboard

Cases

EVIDENCE

Audit Trail

Risk Dashboard

Anomaly Detection

AI TOOLS

AI Assistant

ADMINISTRATION

Admin Panel

ACCOUNT

System Administrator

Security (2FA)

Log Out

Investigation Cases

+ New Case

CASE NO.	TITLE	JURISDICTION	STATUS	OPENED	EVIDENCE	
CASE-2025-008	Cyberstalking – Social Media Harassment	Italy, Messina	Open	06 Jun 2025	1	<div>View</div>
CASE-2025-007	Corporate Espionage – Trade Secret Theft	Italy, Messina	Pending	02 May 2025	1	<div>View</div>
CASE-2025-006	Supply Chain Attack – Software Vendor	Italy, Messina	Active	13 Jun 2025	1	<div>View</div>
CASE-2025-005	Cryptocurrency Money Laundering	Italy, Messina	Active	08 Jun 2025	1	<div>View</div>
CASE-2025-004	Phishing Campaign – Financial Sector	Italy, Messina	Open	11 Jun 2025	1	<div>View</div>
CASE-2025-003	Insider Threat – Data Exfiltration	Italy, Messina	Pending	17 Apr 2025	2	<div>View</div>
CASE-2025-002	Ransomware Attack – MediCorp Hospital	Italy, Messina	Active	02 Jun 2025	2	<div>View</div>
CASE-2025-001	Corporate Fraud Investigation	Italy, Messina	Active	17 May 2025	3	<div>View</div>

16.4 Create Case

This page provides a form for creating a new investigation case. We capture the title, description, and jurisdiction, and on submission the system generates a unique case number, records the creator as the case Owner, and writes an audit entry. **Only Administrators and Lead Investigators can reach this page.**

The screenshot shows the 'Create New Case' form in the DEICMS system. The form is located in the main content area, with a 'Back' button in the top right corner. The form fields are:

- Case Title ***: A text input field with a placeholder 'e.g. Corporate Fraud Investigation'.
- Description**: A text area with a placeholder 'Brief description of the case...'.
- Jurisdiction**: A text input field with a placeholder 'e.g. Italy, Messina'.

At the bottom of the form are two buttons: 'Create Case' (with a green checkmark icon) and 'Cancel'.

16.5 Case Detail

The case detail page is the hub for a single investigation. It shows the case's status, jurisdiction, opening date, and description, alongside a Case Members panel where an owner or administrator can grant or revoke other investigators' access. From here a Lead Investigator or Administrator can edit the case, upload evidence, or produce a signed export package. The lower half of the page lists every evidence item attached to the case with its state and risk level.

The screenshot shows the 'Case Detail' page for 'CASE-2026-003: Insider Threat - Data Exfiltration'. The page is divided into several sections:

- Case Information**: A table showing the case's status, jurisdiction, opening date, and description.
- Case Members**: A panel showing the case's members, including the Lead Investigator (Owner) and an 'Add member' button.
- Evidence Items (2)**: A table listing the evidence items attached to the case.

NO.	TITLE	CATEGORY	STATE	RISK	UPLOADED	
EV-2026-005	Data Exfiltration Script - Powershell	Other	Under Analysis	LOW	06 May 2026	View
EV-2026-006	Encrypted Archive - Customer Data Dump	Database	Transferred	LOW	03 May 2026	View

16.6 Edit Case

This page lets an authorised investigator update a case's title, description, jurisdiction, and status. When a case is moved to the Closed status the system records the closing date automatically, and the change is written to the audit trail.

DEICMS

MAIN

Dashboard

Cases

EVIDENCE

AI TOOLS

ADMINISTRATION

ACCOUNT

Audit Trail

Risk Dashboard

Anomaly Detection

AI Assistant

Admin Panel

System Administrator

Security (2FA)

Log Out

Edit CASE-2026-003

Back

Case Title

Insider Threat — Data Exfiltration

Description

Employee suspected of exfiltrating sensitive customer data via USB drives and cloud upload scripts.

Jurisdiction

Italy, Messina

Status

Pending

Open

Active

Pending

Closed

16.7 Upload Evidence

On this page an investigator attaches a digital file to a case. When the file is uploaded the system computes its SHA-256 hash, extracts metadata, and runs the magic-byte validation that checks the file’s true type against its extension and rejects spoofed or executable files. A first custody record is created automatically to mark the upload.

DEICMS

MAIN

Dashboard

Cases

EVIDENCE

AI TOOLS

ADMINISTRATION

ACCOUNT

Audit Trail

Risk Dashboard

Anomaly Detection

AI Assistant

Admin Panel

System Administrator

Security (2FA)

Log Out

Upload Evidence

Back to CASE-2026-003

Evidence Title *

e.g. Suspect Hard Drive Image

Description

Brief description...

Category

Image

Collection Location

e.g. Suspect's office, Room 3B

Evidence File *

Choose File No file chosen

Max 100 MB. Supported: images, video, audio, documents, databases, logs.

A SHA-256 hash will be computed automatically after upload to establish the integrity baseline.

Upload Evidence

Cancel

Evidence Title *

e.g. Suspect Hard Drive Image

Description

Brief description...

Category

Image

Image

Video

Audio

Document

Database

Log File

Other

A SHA-256 hash will be computed automatically after upload to establish the integrity baseline.

Upload Evidence

Cancel

16.8 Evidence Detail

The evidence detail page shows everything known about a single item: its evidence number, category, lifecycle state, risk level, file information, original and current SHA-256 hashes, and extracted metadata. It also displays the complete custody log for the item and provides the actions available to the user — transferring custody, verifying integrity, reconstructing the chain, changing the lifecycle state, editing the metadata or checking the audit trail.

EV-2026-005: Data Exfiltration Script — PowerShell

Edit

Transfer

Change State

Verify Integrity

Reconstruct Chain

Audit Trail

Risk Score

Evidence Details

Case

CASE-2026-003

Category

Other

File Name

sync_backup.ps1

File Size

41 KB

MIME Type

application/x-powershell

Uploaded By

System Administrator

Current Holder

Forensic Analyst

State

Under Analysis

Risk Level

Low

Integrity Hashes

Original Hash (SHA-256)

a8e80d51d722ef6d24cc0162435bedd8a1ddfc8ca29aef1b175d02ef3865449f

Current Hash (SHA-256)

a8e80d51d722ef6d24cc0162435bedd8a1ddfc8ca29aef1b175d02ef3865449f

Hashes match — file integrity confirmed.

DEICMS — System Security Project | Page 24

16.9 Transfer Custody

This page records the handoff of an evidence item from one investigator to another. The current holder selects the recipient and gives a reason; the system then builds the canonical event payload, signs it with the holder’s Ed25519 private key, writes the signed custody record, and advances the lifecycle state. Only the current holder, or an administrator, can perform a transfer.

Transfer Evidence

Transferring EV-2026-005: Data Exfiltration Script — PowerShell. Current holder: Forensic Analyst.

Transfer To *

Reason for Transfer

Transfer Location

Confirm Transfer

Transferring EV-2026-005: Data Exfiltration Script — PowerShell. Current holder: Forensic Analyst.

Transfer To *

Lead Investigator (Lead Investigator)

Forensic Analyst (Analyst)

Read-Only Viewer (Read-Only)

Transfer Location

Confirm Transfer

16.10 Change Lifecycle State

This page drives the evidence state machine. It shows only the transitions that are valid from the current state and that the user’s role is permitted to make, and it requires a justification. The system enforces the preconditions before applying the change and records the result in the audit trail.

Change Evidence State: EV-2026-005

Current state: Under Analysis. Only legally valid transitions for your role are shown below.

Target State *

Justification (recommended)

Apply Transition

Cancel

Current state: Under Analysis. Only legally valid transitions for your role are shown below.

Target State *

Transferred

Archived

Flagged

Apply Transition

Cancel

16.11 Custody Log

The custody log page presents the full, ordered history of custody events for an evidence item — uploads, transfers, accesses, and verifications — each with its timestamp, the parties involved, the file hash at that moment, and the digital signature where present.

Custody Log						
#	EVENT	FROM	TO	LOCATION	REASON	TIME
1	Upload	—	System Administrator	Digital Lab	Recovered from suspect USB	08 May 2026 10:08
2	Transfer	System Administrator	Lead Investigator	Digital Lab	Urgent escalation to lead	08 May 2026 12:08
3	Transfer	Lead Investigator	Forensic Analyst	Digital Lab	Assigned for dynamic analysis	08 May 2026 16:08
4	Transfer	Forensic Analyst	Lead Investigator	Digital Lab	Returned to lead for sign-off	08 May 2026 22:08
5	Transfer	Lead Investigator	System Administrator	Digital Lab	Admin review required	09 May 2026 04:08
6	Transfer	System Administrator	Forensic Analyst	Digital Lab	Re-assigned to analyst	09 May 2026 10:08

16.12 Integrity Verification Report

This page shows the result of running the chain-of-custody graph integrity engine on an evidence item. It **reports the overall verdict (Intact, Partially Verified, or Broken), the completeness score, and a per-event breakdown of the hash, signature, and timestamp checks**, so the investigator can see exactly where any problem lies.

Integrity Verification: EV-2026-005

Back

Partially Verified

5 of 6 nodes verified. Chain completeness: 83.33%. 1 failure(s) detected.

Summary

Total Nodes	6
Verified Nodes	5
Failed Nodes	1
Completeness	83.33%
Run At	27 Jun 2026 11:34

Failures Detected

NODE ID	EVENT	FAILURE TYPE	DETAIL
19	Upload	Hash Mismatch	Evidence file not found at path: /uploads/evidence/evidence.ppt

Custody Node Timeline

POSITION	NODE ID	EVENT	STATUS	TIMESTAMP
1	19	Upload	Failed	2026-05-08T10:08:23
2	11	Transfer	Verified	2026-05-08T12:08:23
3	12	Transfer	Verified	2026-05-08T16:08:23
4	13	Transfer	Verified	2026-05-08T22:08:23
5	14	Transfer	Verified	2026-05-09T04:08:23
6	15	Transfer	Verified	2026-05-09T10:08:23

16.13 Chain Reconstruction Report

The **chain reconstruction page rebuilds the custody timeline for an item and reports a confidence score together with any unresolved gaps it found**, helping an investigator judge how complete and trustworthy the recorded history is.

Chain Reconstruction: EV-2026-005

Back

Chain fully confirmed. All 6 links are verified with no gaps.

6

Total Links

6

Confirmed

0

Inferred

0

Unresolved

Reconstructed Custody Chain

#	EVENT	INVESTIGATOR	LABEL	CONFIDENCE	TIMESTAMP	NOTES
1	Upload	System Administrator	Confirmed	100%	08 May 2026 10:08	
2	Transfer	Lead Investigator	Confirmed	100%	08 May 2026 12:08	
3	Transfer	Forensic Analyst	Confirmed	100%	08 May 2026 16:08	
4	Transfer	Lead Investigator	Confirmed	100%	08 May 2026 22:08	
5	Transfer	System Administrator	Confirmed	100%	09 May 2026 04:08	
6	Transfer	Forensic Analyst	Confirmed	100%	09 May 2026 10:08	

16.14 Risk Dashboard

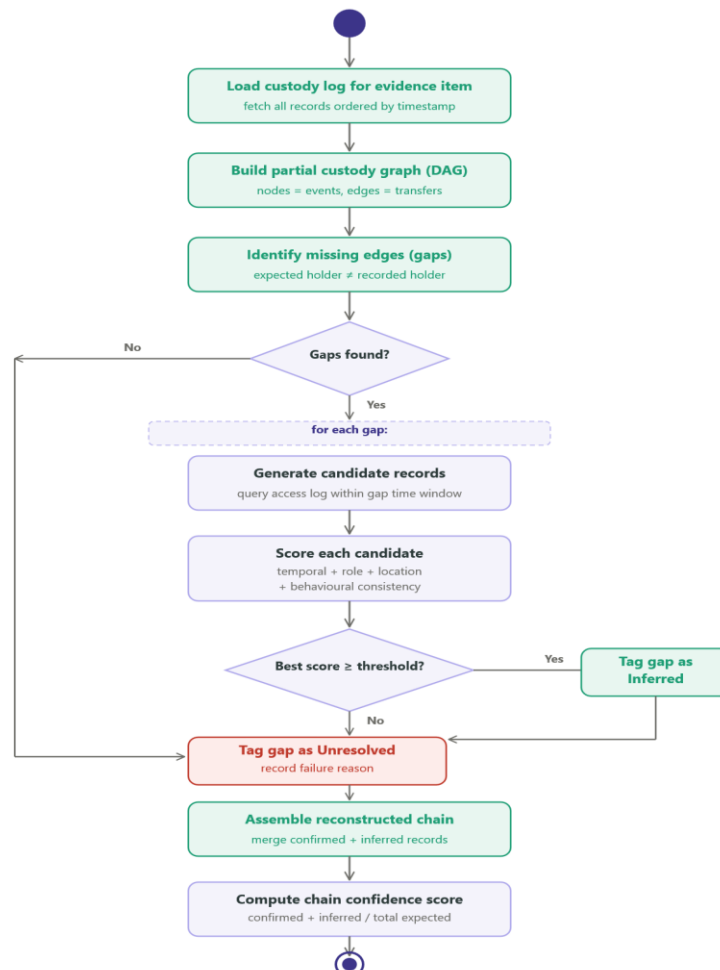
The risk dashboard ranks evidence items by *their computed risk score and shows the band — Low, Medium, High, or Critical — for each*. It lets an investigator quickly focus attention on the items that the multi-factor model considers most dangerous.

The screenshot shows the DEICMS Risk Dashboard. On the left is a sidebar with navigation links: MAIN (Dashboard, Cases), EVIDENCE (Audit Trail, Risk Dashboard, Anomaly Detection), AI TOOLS (AI Assistant), ADMINISTRATION (Admin Panel), and ACCOUNT (System Administrator, Security (2FA), Log Out). The main content area is titled 'Risk Dashboard' and includes a table of evidence items.

RANK	EVIDENCE	CASE	RISK LEVEL	SCORE	INTERACTIONS	ACTIONS
1	#5 — Data Exfiltration Script — PowerShell	#3	CRITICAL	1.000	3 Interactions(3)	Full Report
2	#7 — Ransomware Payload — LockBit Variant	#2	CRITICAL	1.000	3 Interactions(3)	Full Report
3	#9 — Keylogger DLL — Credential Harvesting Module	#2	CRITICAL	1.000	3 Interactions(3)	Full Report
4	#10 — Backdoored Software Installer — v4.2.1	#6	CRITICAL	1.000	3 Interactions(3)	Full Report
5	#6 — Encrypted Archive — Customer Data Dump	#3	MEDIUM	0.400	—	Full Report
6	#4 — Phishing Email Attachment — Invoice Macro	#4	MEDIUM	0.400	—	Full Report
7	#9 — Blockchain Wallet Export — 14 Addresses	#5	MEDIUM	0.347	—	Full Report
8	#12 — Instagram Data Export — Victim Account	#8	MEDIUM	0.333	—	Full Report
9	#11 — USB Activity Log — Suspect Workstation	#7	MEDIUM	0.321	—	Full Report
10	#3 — Financial Transaction Records Q3 2026	#1	MEDIUM	0.204	—	Full Report
11	#1 — Interview Notes — Suspect A	#1	LOW	0.113	—	Full Report
12	#2 — CCTV Screenshot — Server Room Entry	#1	LOW	0.090	—	Full Report

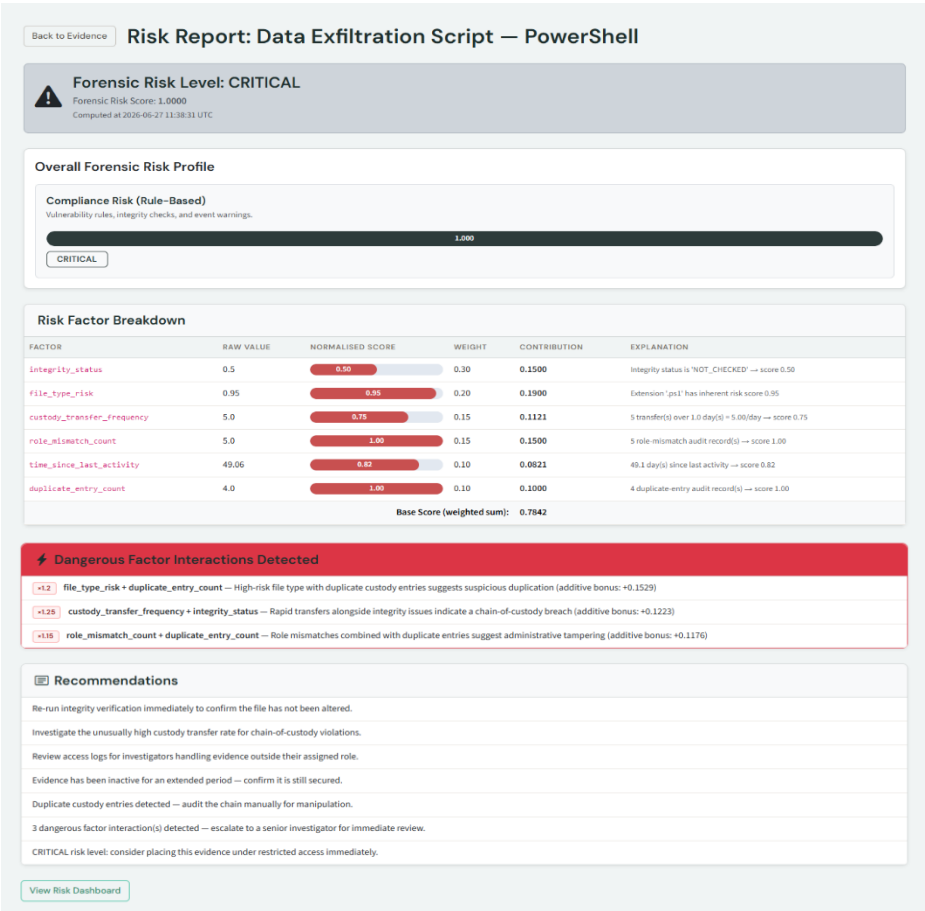
Activity Diagram — Custody Chain Reconstruction

Gap detection → candidate generation → plausibility scoring → confidence labelling



16.15 Evidence Risk Report

This page breaks down the risk score for a single evidence item, *showing how each of the six weighted factors contributed and which interaction rules, if any, fired*. It makes the scoring transparent rather than a black box.



Recommendations

Re-run integrity verification immediately to confirm the file has not been altered.

Investigate the unusually high custody transfer rate for chain-of-custody violations.

Review access logs for investigators handling evidence outside their assigned role.

Evidence has been inactive for an extended period — confirm it is still secured.

Duplicate custody entries detected — audit the chain manually for manipulation.

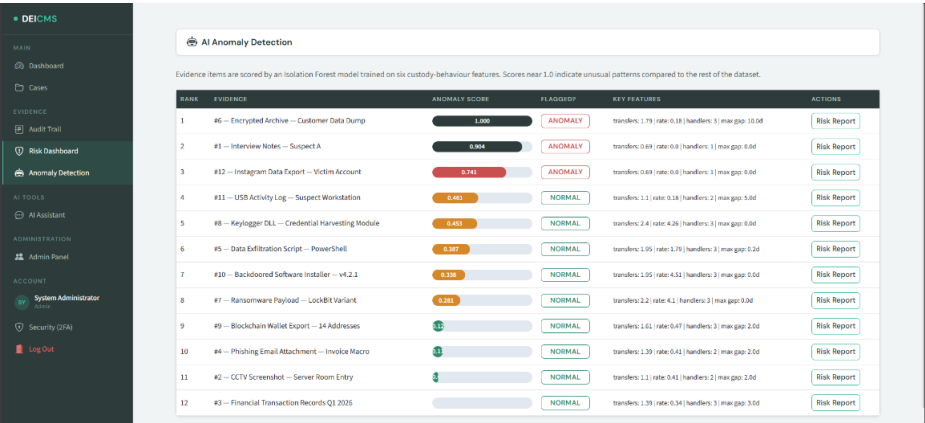
3 dangerous factor interaction(s) detected — escalate to a senior investigator for immediate review.

CRITICAL risk level: consider placing this evidence under restricted access immediately.

View Risk Dashboard

16.16 Anomaly Detection Dashboard

The anomaly dashboard presents the items that *the Isolation Forest model has flagged as statistically unusual, together with their anomaly scores*. It complements the deterministic risk score with a data-driven view of which items stand out from the rest.



16.17 Audit Trail

The audit trail page displays *the immutable, append-only log of every significant action in the system*. Each entry shows the event type, the responsible investigator, the related case or evidence, the outcome, the client IP address, and the timestamp. *Because records are never edited or deleted, this page is the system's primary evidence of accountability*. We *restricted this global trail to administrators-only*; other investigators can still view the per-evidence trail for the cases they belong to.

System Audit Trail					
EVENT	INVESTIGATOR	DESCRIPTION	RESULT	IP	TIME
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	127.0.0.1	27 Jun 2026 16:39:53
File Access	System Administrator	Case package exported by System Administrator. 2 evidence item(s). Signed: No.	Success	127.0.0.1	27 Jun 2026 16:39:49
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	127.0.0.1	27 Jun 2026 16:39:22
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	127.0.0.1	27 Jun 2026 16:39:07
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	127.0.0.1	27 Jun 2026 16:39:04
File Access	System Administrator	System Administrator viewed evidence EV-2026-012	Success	127.0.0.1	27 Jun 2026 16:38:56
File Access	System Administrator	System Administrator viewed evidence EV-2026-012	Success	127.0.0.1	27 Jun 2026 16:38:50
File Access	System Administrator	Case package exported by System Administrator. 1 evidence item(s). Signed: No.	Success	127.0.0.1	27 Jun 2026 16:38:34
Login	System Administrator	Investigator System Administrator logged in.	Success	127.0.0.1	27 Jun 2026 16:33:42
Logout	System Administrator	Investigator System Administrator logged out.	Success	127.0.0.1	27 Jun 2026 16:33:34
Failed Attempt	System	Failed login attempt for email:	Failure	127.0.0.1	27 Jun 2026 16:29:18
Login	System Administrator	Investigator System Administrator logged in.	Success	127.0.0.1	27 Jun 2026 16:25:23
Login	System Administrator	Investigator System Administrator logged in.	Success	127.0.0.1	27 Jun 2026 16:21:43
Login	System Administrator	Investigator System Administrator logged in.	Success	127.0.0.1	27 Jun 2026 14:26:08
Logout	Lead Investigator	Investigator Lead Investigator logged out.	Success	127.0.0.1	27 Jun 2026 14:25:59

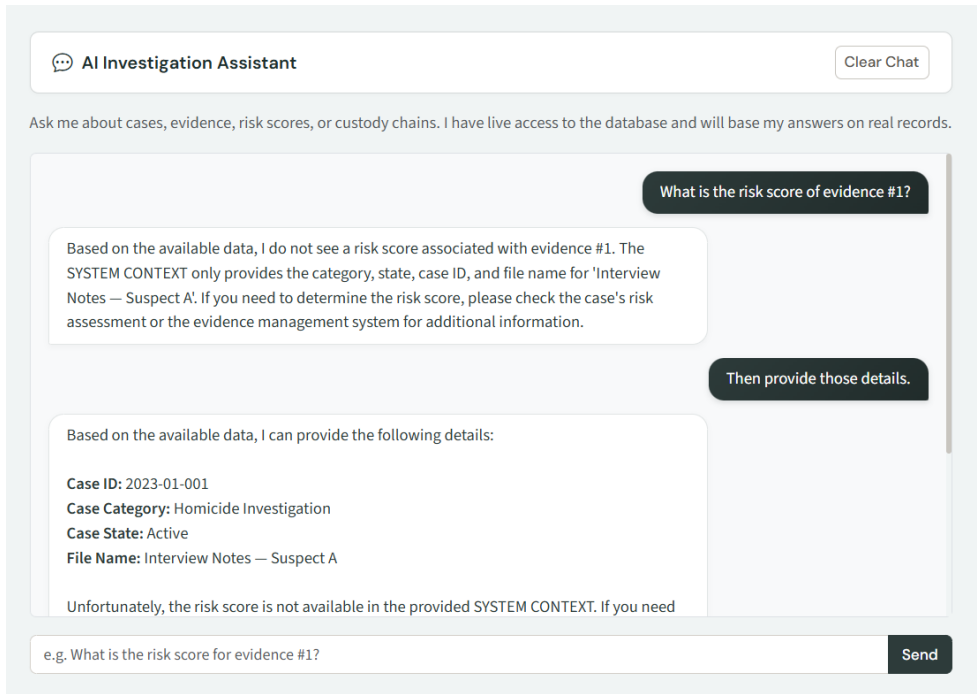
16.18 Evidence Audit Trail

This is a focused view of the audit trail filtered to a single evidence item, so an investigator can see the complete record of who accessed, modified, or verified that specific item and when.

Audit Trail: EV-2026-005					Back
EVENT	INVESTIGATOR	DESCRIPTION	RESULT	TIME	
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	27 Jun 2026 16:39:53	
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	27 Jun 2026 16:39:22	
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	27 Jun 2026 16:39:07	
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	27 Jun 2026 16:39:04	
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	27 Jun 2026 11:52:01	
Integrity Check	System Administrator	Graph integrity verification run on EV-2026-005. Verdict: Partially Verified. Completeness: 83.33%.	Warning	27 Jun 2026 11:51:57	
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	27 Jun 2026 11:51:56	
Integrity Check	System Administrator	Chain reconstruction run on EV-2026-005. Confidence: 100.0%. Unresolved gaps: 0.	Success	27 Jun 2026 11:51:53	
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	27 Jun 2026 11:51:11	
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	27 Jun 2026 11:51:00	
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	27 Jun 2026 11:38:29	
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	27 Jun 2026 11:36:10	
Integrity Check	System Administrator	Chain reconstruction run on EV-2026-005. Confidence: 100.0%. Unresolved gaps: 0.	Success	27 Jun 2026 11:35:42	
File Access	System Administrator	System Administrator viewed evidence EV-2026-005	Success	27 Jun 2026 11:35:40	
Integrity Check	System Administrator	Graph integrity verification run on EV-2026-005. Verdict: Partially Verified. Completeness: 83.33%.	Warning	27 Jun 2026 11:34:42	

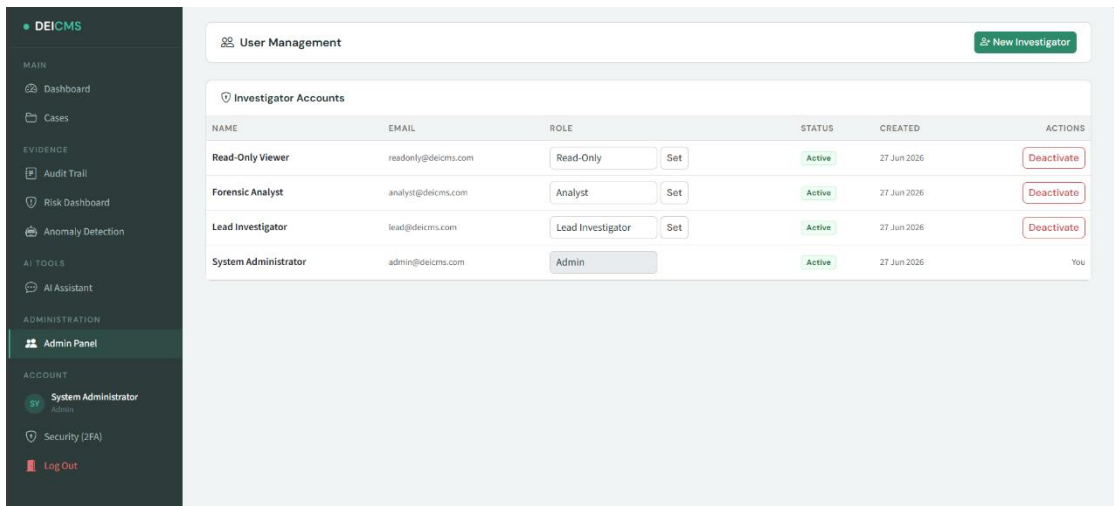
16.19 AI Investigation Assistant

The assistant page provides a chat interface where *an investigator can ask natural-language questions about the cases and evidence*. The assistant answers using *live data from the database, and its requests are protected by the same CSRF token mechanism as the rest of the application*. The conversation stays in place as the user navigates between pages and is only cleared when they press the Clear Chat button.



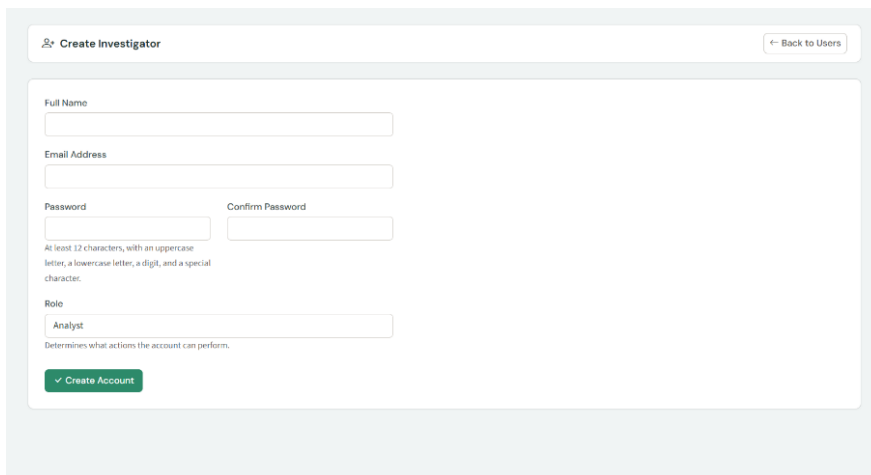
16.20 Administration — User Management

This administrator-only page lists every investigator account with its role, status, and creation date. From here *an administrator can create new accounts, activate or deactivate accounts, unlock accounts that have been locked by failed logins, and change a user's role. Every one of these actions is recorded in the audit trail*.



16.21 Administration — Create Investigator

This page gives administrators a form to create a new investigator account, choosing the person's name, email, password, and role. Because we removed open registration, this is the only way new accounts can be created, which ***prevents privilege escalation through self-registration***.

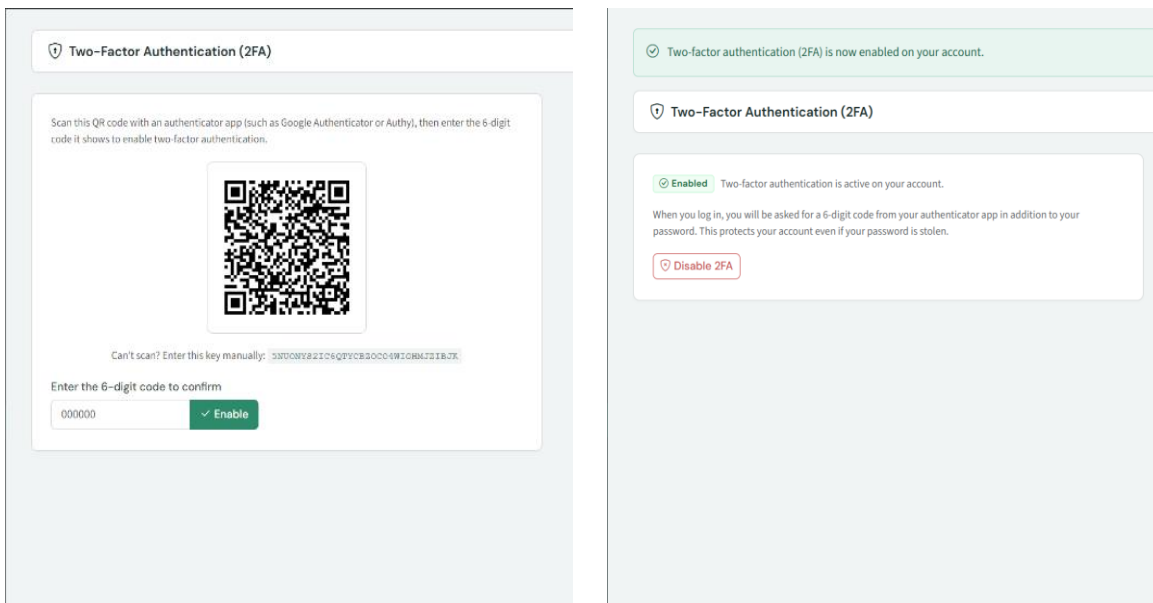


The screenshot shows a web form titled "Create Investigator" with a "Back to Users" link in the top right. The form contains the following fields and instructions:

- Full Name:** A text input field.
- Email Address:** A text input field.
- Password:** A text input field with a note: "At least 12 characters, with an uppercase letter, a lowercase letter, a digit, and a special character."
- Confirm Password:** A text input field.
- Role:** A dropdown menu with "Analyst" selected. A note below reads: "Determines what actions the account can perform."
- Create Account:** A green button with a checkmark icon.

16.22 Security (2FA) Page

This page lets ***a logged-in user manage two-factor authentication for their own account***. If ***2FA is not yet enabled it shows a QR code and a confirmation field for enrolment***; if it is already enabled it shows the active status and ***an option to disable it***.



The image displays two states of the "Two-Factor Authentication (2FA)" page:

Left Screenshot (2FA Not Enabled):

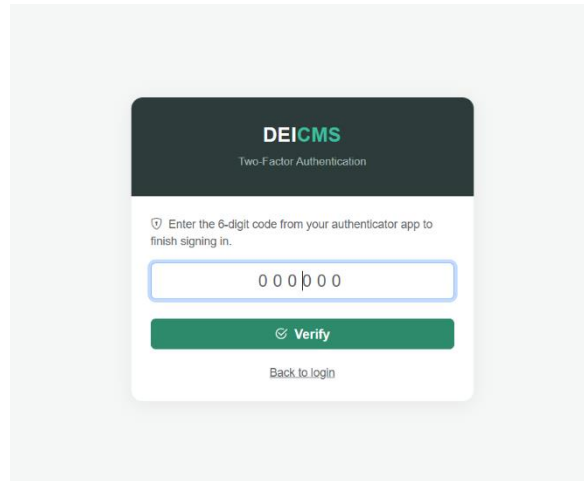
- Section: "Two-Factor Authentication (2FA)"
- Text: "Scan this QR code with an authenticator app (such as Google Authenticator or Authy), then enter the 6-digit code it shows to enable two-factor authentication."
- QR Code: A square QR code for scanning.
- Text: "Can't scan? Enter this key manually: 33D0NY82106QPVCR30004WJ0RMJ218J7"
- Text: "Enter the 6-digit code to confirm"
- Input Field: A text box containing "000000".
- Button: A green "Enable" button with a checkmark icon.

Right Screenshot (2FA Enabled):

- Notification: A green banner at the top stating "Two-factor authentication (2FA) is now enabled on your account."
- Section: "Two-Factor Authentication (2FA)"
- Status: "Enabled" with a green checkmark icon. Text: "Two-factor authentication is active on your account."
- Text: "When you log in, you will be asked for a 6-digit code from your authenticator app in addition to your password. This protects your account even if your password is stolen."
- Button: A red "Disable 2FA" button with a shield icon and a minus sign.

16.23 Two-Factor Verification Page

This page appears during login for accounts that have 2FA enabled. After the password has been accepted it asks for the six-digit code from the user's authenticator app, and it only completes the login once a valid code is entered.



17. Security Concepts Demonstrated

The table below maps the security concepts we demonstrated in the project to the mechanisms we implemented for each.

Security concept	How we demonstrated it
Confidentiality	TLS/HTTPS in transit; AES-256-GCM evidence-file encryption and Fernet-encrypted private keys at rest; signed, HttpOnly and Secure session cookies; access by role and case membership.
Integrity	SHA-256 file hashing; graph-based custody verification; tamper-evident signed export.
Availability	Account lockout and rate limiting protect the login service from brute-force abuse.
Transport security	Self-signed TLS (HTTPS), the Secure cookie flag, and an HSTS header protect data in transit and resist downgrade to HTTP.
Two-factor authentication (2FA)	TOTP second factor from an authenticator app, required after the password.
Encryption at rest	AES-256-GCM for evidence files and Fernet encryption for investigators' private keys.
Session management	Sliding idle-session timeout plus a Secure, time-limited session cookie.
Credential strength	Password-complexity policy enforced when accounts are created.
Authentication	Flask-Login sessions with PBKDF2 password hashing.
Authorisation	Four-role RBAC plus case-level least-privilege access control.

Security concept	How we demonstrated it
Non-repudiation	Ed25519 signatures on custody events; immutable audit trail with IP logging.
Brute-force defence	Login rate limiting (Flask-Limiter) and five-strike account lockout.
CSRF prevention	Application-wide CSRF tokens on all forms and AJAX requests (Flask-WTF).
Privilege-escalation prevention	Removal of open registration; admin-only account creation.
Upload security	Magic-byte (true MIME) validation and executable rejection.
Intrusion detection	Risk scoring, audit-formula checks, and Isolation Forest anomaly detection.

18. Limitations and Future Work

We were careful to understand the remaining weaknesses of our own system, because knowing them is part of good security practice. With HTTPS, encryption of evidence files and private keys at rest, and two-factor authentication (2FA) in place, the system now addresses confidentiality both in transit and at rest, in addition to integrity, authentication, authorisation, and auditing. The honest remaining limitations are that the keys protecting the data at rest are derived from the application secret rather than from a dedicated hardware security module or a per-user passphrase; that the development certificate is self-signed rather than issued by a trusted Certificate Authority; and that browser security headers such as a Content-Security-Policy are not yet set. If we were to extend the project further, we would manage the encryption keys with a dedicated key-management service, obtain a CA-signed certificate for production, add a Content-Security-Policy and related headers, and introduce continuous background integrity monitoring.

19. Conclusion

In this project, we designed and built a realistic digital-evidence management system and used it as a vehicle to implement a broad range of information-security mechanisms. We implemented authentication with hashed passwords, four-role access control together with case-level least privilege, cryptographic file integrity with SHA-256, authenticity and non-repudiation with Ed25519 signatures, a graph-based chain-of-custody verifier, multi-factor risk scoring, machine-learning anomaly detection, and an immutable audit trail. We protected the application against common web attacks with CSRF protection, login rate limiting and account lockout, admin-only account creation, case-level access control, magic-byte upload validation, a tamper-evident signed export, transport security over HTTPS, two-factor authentication (2FA), a password-strength policy, an idle-session timeout, and encryption of evidence files and private keys at rest. Finally, we made the system's security posture visible through a dashboard of live visualisations. Taken together, the project demonstrates the core principles of the System Security syllabus — confidentiality, integrity, availability, authentication, authorisation, and non-repudiation — through working, verified implementations rather than description alone.