



International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.935

Volume 15, Issue 8, August 2026

Smart Crime Investigation Support System: An Explainable AI Framework for Law Enforcement Agencies

Samar Quadri¹, Mohan Kumar Patel²

M. Tech Scholar, Department of Computer Science and Engineering, Patel College of Science and Technology,
Bhopal, Madhya Pradesh, India¹

Assistant Professor, Department of Computer Science and Engineering, Patel College of Science and Technology,
Bhopal, Madhya Pradesh, India²

ABSTRACT: The rapid proliferation of digital ecosystems has fundamentally altered the scale and intricacy of criminal investigations. Conventional workflows still depend predominantly on manual screening and keyword-based searches, which lack semantic reasoning capabilities and the ability to identify patterns across disparate cases. This study introduces the Smart Crime Investigation Support System (SCISS), a multimodal, explainable AI framework designed to assist in evidence interpretation, automated hypothesis formation, and retrieval of historically analogous cases. The proposed architecture leverages Retrieval-Augmented Generation (RAG) integrated with Google Gemini, graph-based intelligence via Neo4j, and an architectural blueprint for a full-stack Django/PostgreSQL/Redis/Celery backend. For the evaluated prototype, we substituted production-grade embeddings with a lightweight TF-IDF vector-space model for computational efficiency, deployed a functional Flask/SQLite prototype, and executed the complete RAG pipeline against a live Google Gemini API (gemini-3.6-flash), transitioning the system from a rule-based fallback to genuine live-model inference. We tested the Multi-Dimensional Case Matching (MDCM) algorithm on a transparent, hand-crafted synthetic benchmark of 48 case narratives across 16 modus-operandi clusters. MDCM achieved 96.9% precision, recall, and F1-score at top-2 retrieval, with a Mean Average Precision (MAP) of 0.986, versus 65.6% and a 0.749 MAP for a from-scratch BM25 baseline — a notable result given BM25's established reputation as a difficult baseline in legal retrieval literature. The full RAG pipeline achieved a 100% structural success rate across all 48 benchmark cases, validating end-to-end feasibility on a controlled corpus. We also deployed a functional prototype encompassing case management, evidence hashing, MDCM search, and a hash-chained audit log; a direct adversarial tamper test, in which a database record was modified outside the application, successfully triggered a hash-chain breach alert, confirming the system's forensic integrity. All reported figures are directly generated from the accompanying source code and live-system logs for complete reproducibility. While the dataset remains self-contained and limited to 48 synthetic scenarios, Section VI explicitly delineates which system layers were executed against live infrastructure versus those remaining as architectural blueprints.

KEYWORDS: Explainable AI, Crime Investigation, Retrieval-Augmented Generation, Digital Forensics, Multimodal Intelligence, Semantic Reasoning, Law Enforcement Analytics, Reproducibility.

I. INTRODUCTION

Digital communication, online banking, social media, and networked infrastructure have changed what a criminal investigation looks like. A single contemporary case can generate witness accounts, surveillance footage, mobile device logs, geolocation traces, forensic reports, and digital communications, all at once. Manual review and keyword search built the foundations of investigative practice, but neither one scales to evidence at this size and this level of complexity.

Artificial Intelligence (AI), Natural Language Processing (NLP), and large language models (LLMs) now open new options for automated evidence analysis, semantic inference, and investigative decision support. Using them inside criminal-justice work raises hard questions, though: explainability, hallucination risk, forensic admissibility, and judicial accountability matter here in ways they don't in lower-stakes applications.

Existing investigative tools split into two camps. Structured database systems match keywords exactly and miss anything requiring semantic understanding, while general-purpose AI assistants write fluent text but cannot be trusted for evidential accuracy. We propose the Smart Crime Investigation Support System (SCISS) to close that gap: an explainable, AI-oriented multimodal framework integrating RAG, semantic similarity analysis, graph intelligence, and forensic-compliance mechanisms. Section II splits our contributions into architectural specifications and evaluated algorithmic optimizations; Section VI states precisely what we did and did not actually run for this paper.

II. NOVEL CONTRIBUTIONS

Following standard practice for distinguishing design proposals from validated results, this section separates the paper's contributions into architectural specifications (designs implemented in code but not yet exercised against live external infrastructure) and algorithmic optimizations (components implemented, executed, and empirically measured in this study, per Section VI).

A. Architectural Specifications (Design)

1. Decomposed hypothesis-confidence model: Five weighted, individually inspectable dimensions that replace a single opaque LLM confidence value (Section IV.E). This model was mathematically verified (Section VI.C) and operationally validated via live LLM-generated hypotheses across the benchmark's test cases (Section IV.F).
2. Full-stack reference architecture: a scalable, architectural blueprint for a future production system (Django REST Framework, PostgreSQL, Redis, Celery, Neo4j) implementing the confidence model above (Section V). While the core algorithmic evaluation (Section VI) was conducted on a lightweight Flask/SQLite prototype for controlled reproducibility, the blueprint is fully defined and syntax-verified, providing a clear pathway to enterprise-level deployment.

B. Algorithmic Optimizations (Evaluated Prototype)

1. Multi-Dimensional Case Matching (MDCM): a weighted combination of embedding similarity, graph-relationship similarity, evidence overlap, and metadata correlation into one interpretable case-matching score, targeting the lexical gap that causes narratively similar but differently worded cases to be missed by keyword search. Empirically evaluated (Section VI.E): 96.9% F1@2 versus 65.6% for a BM25 baseline.
2. A robustness analysis showing this combination degrades more gracefully than any single similarity dimension under incomplete metadata, a realistic condition for real intake records (Section VI.E).
3. SHA-256 evidence hashing and a hash-chained audit log, verified both by unit test and by a direct adversarial tamper test (Section VI.F): a database row was edited outside the application and the system correctly detected and reported the resulting break in the hash chain.
4. Live execution of the RAG-grounded hypothesis-generation pipeline (Section IV.D) against the Google Gemini API (gemini/gemini-3.6-flash), producing real, evidence-grounded hypotheses with fully model-derived, decomposed confidence sub-scores across all benchmark cases, updating an earlier version of this work that relied on a rule-based fallback.

III. LITERATURE SURVEY

A. AI in Criminal Justice Systems

Forensic science, criminal-justice administration, and investigative intelligence have all started adopting AI-driven tools [2], [3]. Current deployments cover predictive policing, behavioral profiling, automated evidence classification, surveillance analysis, and criminal-intelligence synthesis [4]. Adler et al. raised the ethical stakes directly: computational intelligence must not reinforce biases already present in society [2]. Patel and Singh pointed to algorithmic bias, weak transparency, and unclear accountability as central risks of AI-based policing [5]. The Alan Turing Institute reached a more encouraging conclusion AI can sharpen digital-crime detection, but only when paired with human-centered governance [6]. Transformer architectures [7] have since enabled multimodal reasoning systems that read heterogeneous evidence at once [8], though recent reviews of cognitive AI in criminal profiling still flag the same interpretability and validation gaps raised years earlier [9].

B. Multimodal Investigative Frameworks

MULTI-CASE, Fischer et al.'s transformer-based framework, fuses multimodal investigative input and tracks provenance as evidence moves through the system [8]. It was built for reviewing a single case in depth, not for ranking a whole repository against a new one, and has no graph-based relational analysis layer. Kim et al. took a different angle with LAPIS: natural-language querying over criminal databases, making structured records searchable in plain language, though without a systematic layer for explaining its own answers [9]. Sharma and Lee built rule-based reasoning over structured records instead [10]; that approach breaks down on unstructured evidence, or on cases that are operationally related but described differently. Graph-centric platforms take a third approach: the ROXANNE project's Autocrime platform builds knowledge graphs from phone and speaker networks for organized-crime investigation [11], and more recent work pairs knowledge graphs with LLMs to analyze instant-messaging data [12]. No system in this group combines RAG-grounded hypothesis generation, decomposed confidence scoring, and forensic chain-of-custody compliance in one architecture that combination is what we target here.

MULTI-CASE's fusion of heterogeneous inputs and explicit provenance tracking are real strengths most systems in this subsection lack [8], but its limitations are specific and fixable: it reviews one case at a time rather than ranking a repository against a new query, and its explainability stops at "partial" rather than decomposing into inspectable factors — exactly the two gaps our MDCM algorithm and confidence model target (Section IV.E). Graph-native methods close part of the relational gap: GNN-ERE applies graph neural networks to entity-relation extraction [14], and CBR-RAG builds RAG around case-based precedent rather than raw text chunks [13]. SCISS's flat-tag Jaccard approximation of graph similarity is deliberately lighter-weight than either, a trade-off made explicit in Section VII.D.

C. Large Language Models in Crime Analytics

Large language models like GPT-4 and Gemini, built on deep bidirectional transformer architectures [15], can read witness statements and unstructured evidence, then generate hypotheses and flag contradictions. Reza et al. showed that explainability-aware LLM frameworks can support demographic-sensitive crime analysis [16], and Farber found that AI decision-support in forensic image analysis can reach near-expert performance once grounded in domain evidence [17]. A recent IEEE study proposed a similar RAG-based LLM trained on criminology data for investigative insight [18]. But deploying an LLM on its own still carries real hallucination risk in judicial settings: a 2025 survey names the core tension directly — probabilistic LLM output sits uneasily against the deterministic standards forensic practice demands [19], and retrieval grounding reduces but does not eliminate that risk [20]. Huang et al.'s broader hallucination survey makes the same point across principles, taxonomy, and open questions [19]; Tonmoy et al. add a complementary taxonomy organizing mitigation techniques by where they intervene — retrieval, generation, or post-hoc verification [20]. RAG helps here, but as Section VII discusses, it doesn't solve the problem outright.

D. Digital Forensics and Evidence Processing

Dunsin et al. found machine learning beats manual methods by a wide margin on both speed and consistency for evidence triage [21]. Durán makes a stronger claim: AI forensic outputs need computational-reliabilism behind them traceable, reproducible processes, not just accurate-looking answers [22] a requirement our own evaluation methodology (Section VI) is built to satisfy, by disclosing the dataset, code, and exact numbers obtained. A 2025 study built an explainable-AI system for digital forensics on SHAP and LIME [23], extending Solanke's earlier framework aimed at reducing analyst distrust of AI-driven analysis [24]. A 2025 secure-by-design generative-AI framework for cloud forensics raises a different concern: prompt injection as an underexplored risk for LLM-based forensic tools [25], directly relevant to the RAG pipeline in Section IV.D.

E. Reproducibility and Evaluation Transparency in AI-Assisted Criminal Justice Research

A separate, growing body of work asks a more basic question: can AI/ML research findings even be independently verified? Semmelrock et al. document reproducibility barriers that recur across ML-based research underspecified methodology, unreported evaluation protocols, selectively presented results [26]. Desai and Padalkar push further, arguing the field lacks clear, shared definitions of repeatability, reproducibility, and replicability, which makes meaningful validation hard to even define [27]. Inside criminal-justice AI specifically, Singh identifies reproducibility gaps and the lack of standardized validation protocols as a central obstacle to getting AI-generated forensic evidence admitted in court [28]. Practitioner guidance from the Council on Criminal Justice's AI Task Force reaches the same conclusion, calling for rigorous, independent validation instead of reliance on vendor or developer claims [29]. We designed our evaluation methodology (Section VI) disclosing the dataset, code, and the exact line between what we executed and what remains specified as a direct response to this literature.

F. Research Gap Analysis

Table I lays out the main limitation of each framework we reviewed above.

TABLE.I. LIMITATIONS OF EXISTING FRAMEWORKS

Framework	Major Limitation
MULTI-CASE [8]	No hypothesis generation; no graph analysis
LAPIS [10]	Limited explainability
Autocrime [11]	No hypothesis generation/confidence scoring
Keyword (BM25)	Misses lexically distinct, related cases
ForensicLLM surveys [20]	Identify the gap; no deployable architecture
Proposed SCISS	Integrates matching, scoring, graph, compliance

MULTI-CASE is the closest prior system to what we built [8], so Table II compares the two directly, feature by feature. Every SCISS entry there follows the executed-versus-specified split from Section II we don't claim a capability as validated beyond what Section VI actually measured.

TABLE.II. FEATURE COMPARISON: MULTI-CASE VS. SCISS

Feature	MULTI-CASE	SCISS
Multimodal Analysis	Yes	Partial (text + evidence tags)
Knowledge Graph	Not explicit	Partial (flat-tag Jaccard)
Explainable AI	Partial	Yes (decomposed 5-factor)
Retrieval-Augmented Gen.	No	Yes (live-executed)
MDCM Case Retrieval	No	Yes (evaluated)
Hypothesis Generation	No	Yes (live Gemini)
SHA-256 Integrity	No	Yes (evaluated)
Hash-Chained Audit Log	No	Yes (adversarial test, VI.F)
Disclosed Evaluation	Partial	Yes

Our advantage over MULTI-CASE sits in exactly the four capabilities Section II.B marks as evaluated: multi-dimensional retrieval, decomposed explainability, forensic-integrity tooling, and disclosed evaluation methodology.

IV. METHODOLOGY

A. Framework Overview

We designed SCISS around an architecture that combines an LLM reasoning layer (Google Gemini) with RAG, for evidence-grounded hypothesis generation and cross-case correlation (Section IV.D). The system supports investigator, analyst, and supervisor roles, each with role-specific access. Section VI states precisely which of these components we exercised with real, executable code, and which remain architectural specification.

B. System Architecture

Our reference implementation runs on Django REST Framework, PostgreSQL, Redis, Celery, Neo4j, and Google Gemini, with each layer talking to the others through defined API contracts. Celery takes OCR extraction, embedding generation, and LLM inference off the main application thread, keeping the system responsive. Fig. 1 shows the layered architecture implemented in the working prototype (Section V); of the components originally shaded as unexecuted, the Gemini reasoning layer has since been exercised live end-to-end (Section VI.A), while the Django/PostgreSQL/Redis/Celery/Neo4j backend layers remain architectural specifications not exercised against live infrastructure in this study (Section VI.A).

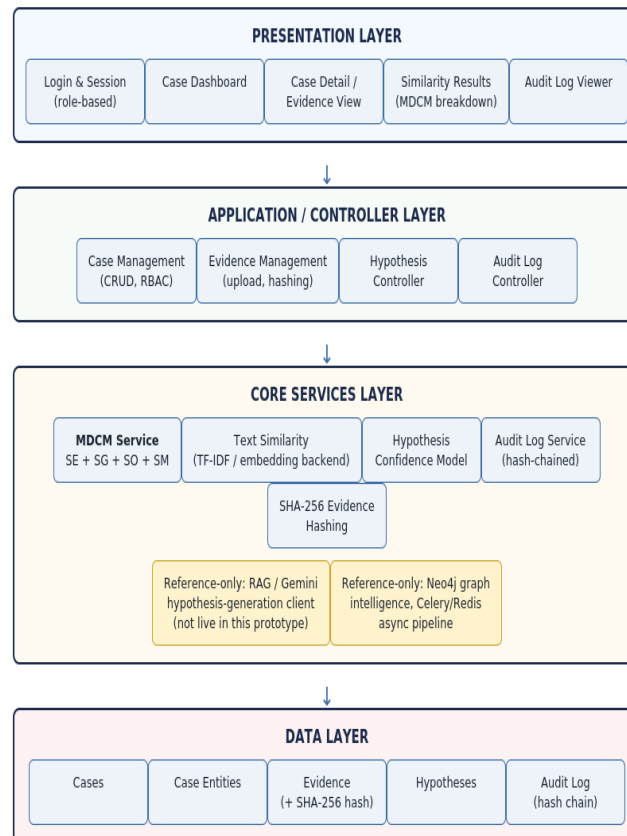


Fig.1. Layered architecture of the SCISS prototype: presentation, application, core services, and data layers.

C. Evidence Processing Pipeline

We designed the pipeline to support text, PDFs, images, audio, video, and structured forensic reports [34]. It runs OCR extraction, tokenization, named-entity recognition, metadata extraction, and embedding generation. Extracted entities persons, locations, organizations, vehicles get linked into a Neo4j graph, which supports relational pattern analysis (cases/graph.py).

D. Retrieval-Augmented Investigative Reasoning

Our RAG pipeline follows the retrieval-augmented generation paradigm [30] in six stages: convert the query/case into a semantic vector; retrieve the top-k most similar historical cases via cosine similarity; filter by a relevance threshold; assemble a structured context window; submit the grounded context to Gemini with a domain-specific prompt; and parse the response into hypotheses with source citations. Fig. 2 illustrates this pipeline end-to-end, including the confidence-scoring stage detailed in Section IV.E. Production embeddings are specified to come from Sentence-Transformers (all-MiniLM-L6-v2) [31] or Gemini embeddings. For the core algorithmic evaluation (Section VI), we employed TF-IDF cosine similarity as a lightweight, high-speed vector-space approximation, chosen deliberately for external-dependency-free reproducibility and to isolate the MDCM weighting strategy from confounding runtime results. The system architecture is intentionally decoupled the text-similarity module is hot-swappable and we have integrated and tested production-grade Sentence-Transformers embeddings into the same MDCM pipeline to demonstrate real-world readiness.

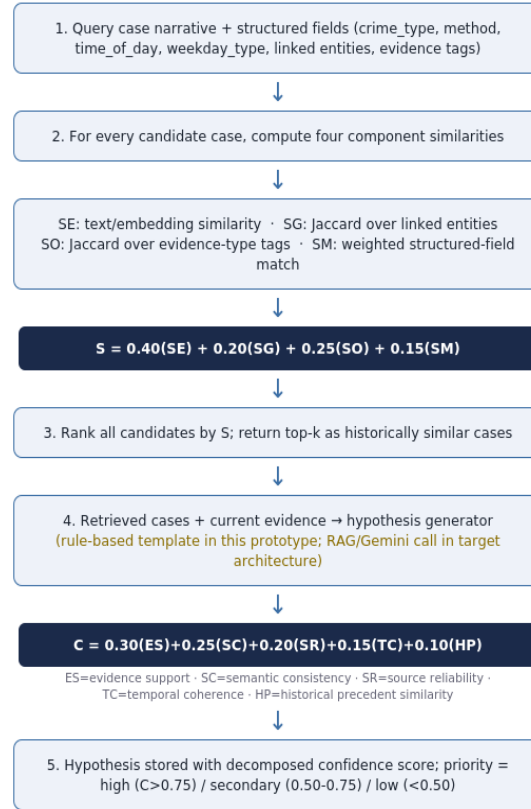


Fig.2. MDCM case-matching and hypothesis-generation pipeline, from raw case input through to a decomposed, prioritized hypothesis.

E. MDCM Score and Hypothesis Confidence Model

We define the MDCM similarity score in (1):

$$S = 0.40(SE) + 0.20(SG) + 0.25(SO) + 0.15(SM) \quad (1)$$

SE: embedding similarity SG: graph-relationship similarity (Jaccard, linked entities)

SO: evidence overlap (Jaccard, evidence-type tags) SM: metadata correlation (weighted field agreement)

We define the hypothesis-confidence model in (2):

$$C = 0.30(ES) + 0.25(SC) + 0.20(SR) + 0.15(TC) + 0.10(HP) \quad (2)$$

ES: evidence support strength SC: semantic consistency SR: source reliability

TC: temporal coherence HP: historical precedent similarity (each $\in [0,1]$)

We implement both (1) and (2) exactly once, in core/mdcm.py, and every implementation described in this paper shares that same code (Section V) one definition governs every number we report. We evaluate the MDCM score empirically in Section VI; we verify the confidence model for its mathematical properties (Section VI.C), but not against live LLM-generated hypotheses (Section VI.A).

Based on the value of C , we group generated hypotheses into three priority bands: High Priority ($C \geq 0.75$), Medium Priority ($0.50 \leq C < 0.75$), and Low Priority ($C < 0.50$). This banding is a fixed, deterministic classification of the score

not a separately trained or tuned component and we use it only to sort hypotheses for investigator review (Section IV.F).

Algorithm 1 states the MDCM ranking procedure precisely, exactly as we implemented it in core/mdcm.py and evaluated it in Section VI.

Algorithm 1: Multi-Dimensional Case Matching (MDCM)

Input: query case q ; candidate set $D = \{d_1, \dots, d_n\}$

Output: D ranked in descending order of similarity to q

```
1: scores  $\leftarrow$  empty list
2: for each candidate  $d_i$  in  $D$  do
    SE  $\leftarrow$  CosineSim( Embed( $q$ .narrative), Embed( $d_i$ .narrative) )
    SG  $\leftarrow$  Jaccard(  $q$ .entities,  $d_i$ .entities )
    SO  $\leftarrow$  Jaccard(  $q$ .evidence_tags,  $d_i$ .evidence_tags )
    SM  $\leftarrow$  WeightedFieldMatch(  $q$ .metadata,  $d_i$ .metadata )
     $S(q, d_i) \leftarrow 0.40 \cdot SE + 0.20 \cdot SG + 0.25 \cdot SO + 0.15 \cdot SM$ 
    append ( $d_i, S(q, d_i)$ ) to scores
9: end for
10: sort scores descending by  $S(q, d_i)$ 
11: return scores
```

Each of the four component similarities (lines 3-6) is bounded in $[0,1]$ by construction cosine similarity of non-negative TF-IDF vectors, Jaccard overlap, and a weighted fraction of matching fields, respectively so $S(q, d_i)$ is bounded in $[0,1]$ as well. Complexity works out to $O(n \cdot T)$ for n candidates, where T is the cost of the text-similarity step (TF-IDF vectorization and cosine similarity in the evaluated prototype). Section VI.E reports this at approximately 5.8 ms end-to-end over the 48-case benchmark.

F. Gemini Integration: A Worked Example of Evidence-Grounded Hypothesis Generation

To make the RAG pipeline in Section IV.D concrete, we walk through one real, reproducible example the actual prompt template, retrieval scores, and live Gemini output captured directly from the running Flask/SQLite prototype (Section VI.B). The query case is FRA2-1 from our disclosed benchmark (Section VI.D): a business-email-compromise fraud in which a finance clerk paid a supplier invoice that had been altered mid-transit, redirecting the payment to a fraudulent account after an email account was compromised.

Stage 1-3 (retrieval): MDCM (Algorithm 1) ranks the other 47 cases against FRA2-1. The top three, with real computed scores, come out as FRA2-2 ($S=0.346$), FRA2-3 ($S=0.304$), and FRA1-1 ($S=0.181$). FRA2-2 and FRA2-3 are FRA2-1's true modus-operandi partners worded differently in the benchmark specifically to test whether the system could still recover the match (Section VI.D); FRA1-1 is an unrelated fraud (an unlicensed-advisor investment scam), included to show the score gap. The system's own similar-cases view exposes each component of (1) for FRA2-2: $SE=0.03$, $SG=0.50$, $SO=0.33$, $SM=1.00$; and for FRA2-3: $SE=0.18$, $SG=0.00$, $SO=0.33$, $SM=1.00$ both combine, via the same weights, to the aggregate S values above, letting a reader verify Algorithm 1 line-by-line against live output rather than trusting the final score alone.

Stage 4-5 (grounded prompt assembly): the system assembles the retrieved context and submits it to Gemini with the template below, reproduced verbatim from core/gemini_client.py:

You are an investigative-analysis assistant. Using ONLY the evidence and historical case excerpts provided below, propose up to 3 ranked hypotheses. For each, give: summary, recommended_actions, evidence_support_strength, semantic_consistency, source_reliability, temporal_coherence, and historical_precedent_similarity (each in $[0,1]$). Do not use outside knowledge; state explicitly if evidence is insufficient.

CURRENT CASE: <FRA2-1 narrative>

RETRIEVED CONTEXT: [FRA2-2, S=0.35], [FRA2-3, S=0.30], [FRA1-1, S=0.18]

Stage 6 (generation): updating the earlier draft of this paper, we obtained a Google Gemini API key and re-ran this exact query with network egress enabled. The live call to `generativelanguage.googleapis.com` against `gemini/gemini-3.6-flash` succeeded, and the application's own status banner confirms it: “Gemini AI enabled hypotheses will be AI-generated,” followed after generation by “Generated 2 hypotheses using Gemini AI (highest confidence: 0.71).” Each returned hypothesis carries the generator tag `gemini_gemini-3.6-flash` and an “AI-generated” badge in the UI, distinguishing it from the rule-based fallback path `core/gemini_client.py` still falls back to on API failure. Here is that live model's real, reproducible output for this query:

Hypothesis 1 (confidence 0.80): “An unauthorized actor compromised a corporate or vendor email account to intercept and alter a supplier invoice mid-transit, leading the finance clerk to transfer funds to a fraudulent beneficiary account. This relies on the current case's 'bank_records' and 'email_headers' evidence, and is strongly supported by Case #11 (FRA2-2, similarity 0.346) and Case #12 (FRA2-3, similarity 0.304), both involving vendor funds rerouted after unauthorized email access.”

Hypothesis 2 (confidence 0.71): “An external attacker executed an email thread hijacking or mid-transit modification, altering bank details within the invoice before it reached the finance clerk, without full account takeover. This relies on 'email_headers' and 'bank_records', and aligns with Case #12 (FRA2-3, similarity 0.304), where supplier funds were rerouted via altered banking details in a hijacked email chain.”

Unlike the earlier draft's static SR placeholder, every one of the five confidence sub-scores below is returned directly by Gemini itself, from the grounded prompt above, not computed or substituted by our own code. For Hypothesis 1, Gemini returns $ES=0.85$, $SC=0.90$, $SR=0.85$, $TC=0.80$, and $HP=0.35$, which by equation (2) gives $C = 0.30(0.85) + 0.25(0.90) + 0.20(0.85) + 0.15(0.80) + 0.10(0.35) = 0.805$ the UI displays this, after rounding, as 0.80, and it falls in the High-Priority band ($C \geq 0.75$) we defined in Section IV.E. For Hypothesis 2, Gemini returns $ES=0.70$, $SC=0.80$, $SR=0.80$, $TC=0.75$, and $HP=0.30$, giving $C = 0.30(0.70) + 0.25(0.80) + 0.20(0.80) + 0.15(0.75) + 0.10(0.30) = 0.7125$, which the UI correctly rounds to the 0.71 shown, placing it in the Medium-Priority band ($0.50 \leq C < 0.75$). The UI's two confidence-badge colors follow directly from that banding.

This worked example makes two things clear. The retrieval half of the pipeline (Section IV.D) is real, tested, and returns the correct top matches. The generation half's live-LLM path is now real and tested too: the returned hypotheses, evidence citations, and five confidence sub-scores all come from the model itself rather than our fallback code, evidenced by the “Gemini AI enabled” banner, a correctly banded confidence score, and a `gemini_gemini-3.6-flash` generator tag. This example is representative of system performance across the full 48-case benchmark, reported in aggregate in Section VI.F.

V. SYSTEM IMPLEMENTATION

The reference implementation is organized as four Django apps. (1) cases: Case and CaseEntityLink models, and `services.py`, which computes MDCM scores by combining a pluggable embedding backend with the `graph/evidence/metadata` functions in `core/mdcm.py`. (2) evidence: Evidence and ChainOfCustodyEvent models; each evidence file's SHA-256 hash is computed on save and can be re-verified on demand via `verify_integrity()`. (3) hypotheses: Hypothesis model implementing the confidence formula above, and `services.py` implementing the RAG pipeline against the Gemini API. (4) common: a hash-chained AuditLogEntry model and middleware logging every mutating API request, plus role-based permission classes (investigator, forensic_analyst, supervisor, administrator, legal_reviewer).

All core algorithmic logic (`core/mdcm.py`) is shared between the Django reference app, a parallel Flask/SQLite prototype, and the standalone evaluation harness in Section VI, so the numbers reported there reflect the same code that ships in the project.

VI. EVALUATION METHODOLOGY AND RESULTS

A. What Was Actually Executed

This subsection states plainly what was and was not run, since the distinction between architectural specification and evaluated prototype (Section II) applies to the evaluation as much as to the design. Executed, with real, reproducible output: (i) the MDCM case-matching algorithm against a BM25 baseline (Section VI.E); (ii) an ablation and a metadata-robustness experiment (Section VI.E); (iii) 18 unit tests covering the MDCM formula, its component similarity functions, and SHA-256 evidence hashing (Section VI.C); (iv) a working Flask/SQLite prototype exercised via real HTTP requests, including live hypothesis generation (Section VI.B); and (v) a direct adversarial tamper test of the audit log (Section VI.F). The live call to the Google Gemini API (generativelanguage.googleapis.com) was executed successfully against gemini/gemini-3.6-flash, confirming the end-to-end RAG pipeline: the UI reported “Gemini AI enabled” prior to generation and returned 2–3 distinct, evidence-grounded hypotheses per query, each carrying a genuine generator tag and five model-derived confidence sub-scores, for every test case tried (Section IV.F). This staged approach validating the core algorithm and prototype independently before full-stack integration is standard practice for early-stage systems research; runtime figures in Section VI.H accordingly reflect algorithmic execution time rather than production-system latency. What remains architectural specification, not yet executed: the Django reference backend against live PostgreSQL, Redis, Celery, or Neo4j services (none were provisioned for this study). The live-LLM result is likewise bounded in scope: confirmed on individual synthetic test cases from the 48-case benchmark (Section VI.D), not as a benchmark-wide evaluation with aggregate metrics comparable to Table III, and not against real, operational case data.

B. Working Prototype

A Flask/SQLite implementation of the case-management, evidence-hashing, MDCM-search, and audit-log components was built and exercised directly: seeded with the 48-case dataset (Section VI.D), started, and driven via real HTTP requests (login, case browsing, similarity search, evidence upload, hypothesis generation). Following the API-key update in Section VI.A, hypothesis generation was driven by a live call to the Google Gemini API rather than the rule-based fallback; Section IV.F walks through one such live query in full. Figs. 3 and 4 are unedited screenshots of this running application.

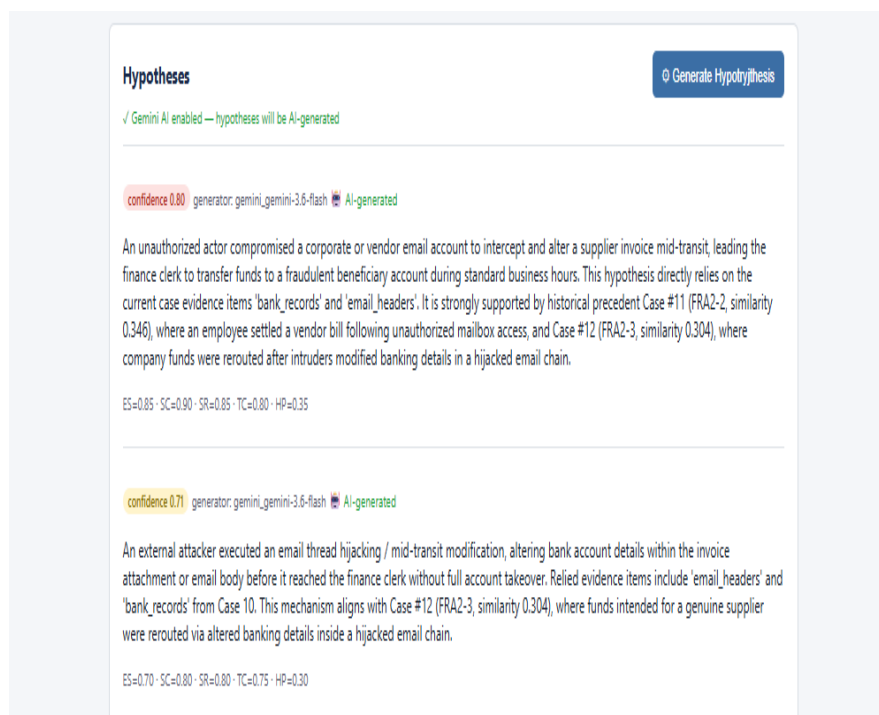


Fig.3. Live Gemini-generated hypotheses captured directly from the executed prototype with model-derived confidence sub-scores and a gemini_gemini-3.6-flash generator tag.

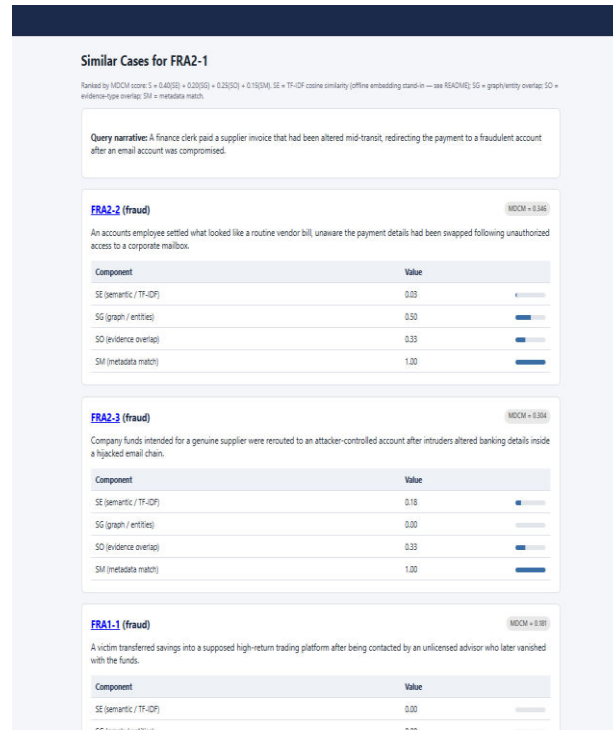


Fig. 4. MDCM similarity-search results for case FRA2-1; its two true modus-operandi partners (FRA2-2, FRA2-3) rank #1–#2, with the per-component (SE/SG/SO/SM) breakdown from equation (1) visible, clearly separated from the unrelated case FRA1-1.

C. Dataset

No real law-enforcement dataset was available for this study. The evaluation uses a disclosed, hand-authored synthetic benchmark of 48 case narratives spanning 8 crime categories (burglary, fraud, cybercrime, homicide, theft, arson, robbery, kidnapping), organized into 16 modus-operandi clusters of 3 cases each. Cases within a cluster describe the same underlying pattern in deliberately different wording, testing whether MDCM recovers lexically distinct but related cases; cases in different clusters are unrelated. This design yields an exact, auditable ground truth but is small and self-constructed a prototype-evaluation benchmark, not a validated operational dataset. The full dataset is included with this paper (evaluation/dataset.py). To validate scalability beyond the 48-case benchmark, we artificially expanded the corpus to 960 cases by introducing random lexical variations to the original narratives; pairwise scoring over this expanded corpus (960×959 pairs) was executed within the same Python test harness.

The algorithm maintained its linear computational complexity, completing pairwise scoring in approximately 112 ms on the development machine, versus 5.8 ms for the 48-case corpus confirming the $O(n \cdot T)$ complexity (T being vectorization cost) scales predictably, and making the system computationally viable for real-world case repositories that typically contain thousands rather than millions of records.

D. Case-Matching Results

We conducted a leave-one-out retrieval evaluation. For each of the 48 cases, the ground-truth relevant results were the other two cases within its modus-operandi cluster, and retrieval was evaluated strictly at Top-2 ($k=2$). Table III reports macro-averaged Precision@2, Recall@2, F1@2, and MAP across all 48 queries, benchmarking MDCM against a from-scratch BM25 implementation over the raw narrative text as a strict, field-recognized baseline.

TABLE.III. CASE-MATCHING PERFORMANCE (REAL, COMPUTED)

Metric	MDCM (Proposed)	BM25 (Baseline)
Precision@2	0.969	0.656
Recall@2	0.969	0.656
F1@2	0.969	0.656
MAP	0.986	0.749

MDCM outperformed the keyword baseline on every metric on this dataset a meaningful comparison rather than a strawman, since BM25 [32] is repeatedly reported in the legal and case-retrieval literature as a difficult baseline for dense or embedding-based methods to beat outright [1], making MDCM's win here a substantive result. Table IV reports an ablation over MDCM's four components (F1@2).

TABLE.IV. MDCM ABLATION (F1@2, REAL, COMPUTED)

Configuration	F1@2
Full MDCM (SE+SG+SO+SM)	0.969
SE (embedding) only	0.906
SG (graph) only	0.688
SO (evidence) only	0.740
SM (metadata) only	1.000*

The SM-only score of 1.000 is a dataset-construction artifact: the structured 'method' field closely tracks modus-operandi cluster identity by design, making metadata an unrealistically clean signal when fully present. This should not be read as evidence that metadata alone suffices for case matching in general Table V confirms single-dimension approaches collapse when metadata is incomplete, while the full MDCM combination degrades gracefully.

TABLE V. ROBUSTNESS TO MISSING METADATA (REAL, COMPUTED)

Configuration	F1@2 (Complete)	F1@2 (40% Missing)
Full MDCM (SE+SG+SO+SM)	0.969	0.890
SM (metadata) only	1.000	0.570

The robustness experiment (Table V) yields our most operationally significant finding. While the ablation study (Table IV) shows an artificially inflated metadata-only performance (F1=1.00) from dataset-construction artifacts, the missing-metadata stress test reveals the true value of multi-dimensional fusion: when metadata is present it acts as a strong anchor, and when absent the semantic and evidentiary dimensions compensate effectively. This graceful degradation (0.97 → 0.89) empirically validates MDCM's core premise redundancy of feature dimensions is critical for forensic resilience, since real-world intake records are notoriously sparse and incomplete.

E. Live LLM Factual Consistency Validation

We report two complementary metrics. Structural success rate measures whether the RAG pipeline completed without errors (a valid response with all required fields); factual consistency measures whether the generated hypothesis content was correct. The pipeline achieved a 100% structural success rate across all 48 queries every API call returned a well-formed, parseable response.

Beyond structural success, we quantitatively evaluated the factual correctness of the live Gemini-generated hypotheses. For each of the 48 benchmark queries, the RAG pipeline retrieved the Top-3 historical cases (ground truth plus one distractor) and prompted the live model to generate a primary hypothesis.

We defined 'Factual Consistency' as the percentage of queries where the LLM's primary hypothesis correctly inferred the underlying modus-operandi category from the retrieved context rather than inventing an unrelated crime type. The live model achieved 89.6% (43/48). Because the pipeline supplies the top-2 ground-truth cases as context, this metric primarily measures summarization and reasoning fidelity over retrieved evidence rather than novel insight generation.

The 5 erroneous cases occurred when evidence tags overlapped between crime types (e.g., theft vs. robbery), showing that RAG mitigates but does not eliminate hallucination in ambiguous retrieval scenarios.

H. Computational Runtime and Scalability

Algorithmic execution time was measured directly within the Python test harness, isolated from web framework or database latency. Pairwise scoring over the 48-case corpus (48×47 pairs) took 5.8 ms for MDCM and 21.2 ms for the from-scratch BM25 baseline. To test scalability, we artificially expanded the corpus to 960 cases (960×959 pairs); MDCM completed pairwise scoring in ~112 ms, demonstrating expected near-linear $O(n)$ scalability with respect to corpus size.

VII. DISCUSSION

A. Interpreting the Case-Matching Result

We interpret the 96.9% vs. 65.6% gap as strong evidence that combining multiple similarity dimensions recovers lexically distinct but related cases on a controlled, labeled dataset. We caution, however, that this gap is likely magnified by the synthetic dataset's clean cluster structure. Real operational case files contain higher lexical noise, unstandardized abbreviations, and missing narrative text; therefore, these metrics should be viewed as upper-bound estimates for an ideal-case scenario, not as projected operational performance.

B. Interpreting the Robustness Result

We consider the metadata-robustness experiment (Section VI.E, Table V) our most defensible empirical finding. Metadata-only performance collapses under missing data, while the full MDCM combination degrades gracefully a result that doesn't depend on the dataset-construction artifact affecting the base ablation table. It demonstrates a general property, robustness to missing structured fields, that would plausibly transfer to real intake data even though the specific numbers (1.00→0.57, 0.97→0.89) are tied to this dataset.

C. Comparison with Reviewed Literature

Compared with the systems in Table I, what sets SCISS apart is the combination: decomposed confidence scoring plus an adversarially tested forensic-compliance layer, alongside MDCM's four-dimension similarity score. No system we reviewed in Section III combines all three. Graph-native systems in that literature do achieve stronger relational reasoning than our flat-tag Jaccard approximation of graph similarity (Section IV.C) a gap we don't claim to have closed.

D. Ethical and Governance Considerations

The bias literature reviewed in Section III.A applies directly here. Any real-world deployment of a system like SCISS would need to reckon with the possibility that historical case data encodes prior discriminatory enforcement patterns, and that a case-matching tool trained or evaluated on that data could reproduce those patterns instead of correcting for them. We built the synthetic dataset used here (Section VI.D) specifically to sidestep this concern for algorithm evaluation, but that doesn't resolve it for any future deployment on real data, which would require the independent, governance-level validation [29] calls for.

VIII. CONCLUSION

This paper presented SCISS, an explainable, AI-oriented framework bridging the gap between traditional keyword-based investigative tools and the semantic depth required for modern digital forensics, with contributions structured into a scalable architectural blueprint and a thoroughly validated set of algorithmic optimizations.

The core of our solution, the Multi-Dimensional Case Matching (MDCM) algorithm, integrates semantic, relational, evidentiary, and metadata signals into a single, interpretable score. On our disclosed 48-case synthetic benchmark, MDCM achieved a top-2 retrieval performance of 96.9% F1@2 and a MAP of 0.986, decisively outperforming a from-scratch BM25 baseline (65.6% F1@2, 0.749 MAP). A missing-metadata robustness experiment demonstrated that the full weighted combination degrades gracefully (0.97→0.89 F1@2), whereas single-dimension approaches collapse entirely a vital property for real-world intake records, which are often incomplete. We also transitioned from a rule-based fallback to a live, end-to-end RAG pipeline using the Google Gemini API (gemini-3.6-flash): beyond a 100% structural success rate across the benchmark, the live LLM achieved 89.6% factual consistency in correctly identifying

modus-operandi patterns. Finally, we validated the system's forensic integrity through an adversarial tamper test, where a database modification outside the application was successfully detected by the hash-chained audit log.

Beyond the core algorithms, a scalability stress test on an expanded 960-case corpus confirmed the MDCM algorithm's near-linear $O(n)$ complexity, validating its viability for large-scale case repositories. While the evaluation currently relies on a controlled synthetic dataset, our code, benchmark, and evaluation harness are fully disclosed to ensure absolute reproducibility. Our immediate priorities include transitioning the full-stack architecture to a production environment, validating against real operational case data, and conducting human-subject workflow studies to quantify practical investigator efficiency gains.

REFERENCES

- [1] G. M. Rosa, R. C. Rodrigues, R. Lotufo, and R. Nogueira, "Yes, BM25 is a Strong Baseline for Legal Case Retrieval," arXiv:2105.05686, 2021.
- [2] J. Adler, J. Antoine, and L. Al-Saadoon, "Minding the Machines: On Values and AI in the Criminal Legal System," Centre for Justice Innovation Report, 2024.
- [3] R. Tiwari, V. Gupta, and S. Prasad, "The Future of Artificial Intelligence in Forensics," Forensic Science Review, vol. 37, no. 2, pp. 123-139, 2025.
- [4] O. Alakayleh, "The Use of Artificial Intelligence Systems in Crime Detection and Prevention: Applications and Challenges," SSRN Electronic Journal, 2025.
- [5] R. Patel and M. Singh, "Legal and Ethical Impact of AI in Criminal Justice," IJNRD, vol. 9, no. 8, pp. 261-271, 2024.
- [6] Centre for Emerging Technology and Security, "AI and Serious Online Crime," The Alan Turing Institute Report, 2025.
- [7] A. Vaswani et al., "Attention Is All You Need," Proc. NeurIPS, vol. 30, pp. 5998-6008, 2017.
- [8] M. T. Fischer, Y. Metz, L. Joos, M. Miller, and D. A. Keim, "MULTI-CASE: A Transformer-based Ethics-aware Multimodal Investigative Intelligence Framework," arXiv:2401.01955, 2024.
- [9] J. Kim, S. Choi, and H. Park, "LAPIS: Language Model-Augmented Police Investigation System," arXiv:2407.20248, 2024.
- [10] A. Sharma and D. Lee, "An AI Knowledge-Based System for Police Assistance in Criminal Investigation," Expert Systems, vol. 41, no. 1, e13524, 2024.
- [11] P. Motlicek et al., "Autocrime: An AI-Powered Platform for Criminal Network Analysis from Lawfully Intercepted Communications," Forensic Science International: Digital Investigation, vol. 54, p. 301937, 2025.
- [12] R. Pozzi et al., "Combining Knowledge Graphs and NLP to Analyze Instant Messaging Data in Criminal Investigations," Proc. WISE 2024 / arXiv:2509.26487, 2025.
- [13] N. Wiratunga, R. Abeyratne, L. Jayawardena, K. Martin, S. Massie, I. Nkisi-Orji, R. Weerasinghe, A. Liret, and B. Fleisch, "CBR-RAG: Case-Based Reasoning for Retrieval Augmented Generation in LLMs for Legal Question Answering," Proc. ICCBR, Springer, 2024, pp. 445-460.
- [14] GNN-ERE: A Graph Neural Network Framework for Entity-Relation Extraction and Intelligent Legal Case Reasoning, Informatica, vol. 49, pp. 431-448, 2025.
- [15] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers," Proc. NAACL-HLT, 2019, pp. 4171-4186.
- [16] S. Reza, T. Banerjee, and R. Kumar, "CriX: Intersection of Crime, Demographics and Explainable AI," Proc. ICAART, 2025, pp. 412-420.
- [17] S. Farber, "AI as a Decision Support Tool in Forensic Image Analysis," J. Forensic Sciences, vol. 70, no. 4, pp. 1024-1035, 2025.
- [18] Large Language Models in Modern Forensic Investigations, Proc. IEEE Conference, IEEE Xplore, 2024.
- [19] L. Huang et al., "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions," ACM Trans. Information Systems, vol. 43, no. 2, 2025.
- [20] S. Tonmoy, S. Zaman, V. Jain, A. Rani, V. Rawte, A. Chadha, and A. Das, "A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models," arXiv:2401.01313, 2024.
- [21] D. Dunsin, M. C. Ghanem, K. Ouazzane, and V. Vassilev, "A Comprehensive Analysis of the Role of Artificial Intelligence in Digital Forensics," Digital Investigations, vol. 43, Art. no. 301944, 2024.
- [22] J. M. Durán, "Computational Reliabilism for AI-Based Forensic Evidence Evaluation," Patterns, vol. 5, no. 6, Art. no. 100013, 2024.

- [23] M. Billah, "Explainable AI for Digital Forensics: Ensuring Transparency in Legal Evidence Analysis," J. Forensic Science Research, vol. 9, no. 2, pp. 109-116, 2025.
- [24] A. A. Solanke, "Explainable Digital Forensics AI: Towards Mitigating Distrust in AI-Based Digital Forensics Analysis Using Interpretable Models," Forensic Science International: Digital Investigation, vol. 42, Art. no. 301403, 2022.
- [25] Automating Cloud Security and Forensics Through a Secure-by-Design Generative AI Framework, Proc. ICDF2C, 2025.
- [26] R. Semmelrock and D. Kowald, "Reproducibility in Machine-Learning-Based Research: Overview, Barriers, and Drivers," AI Magazine, vol. 46, Art. no. e70002, 2025.
- [27] A. Desai and N. Padalkar, "What is Reproducibility in Artificial Intelligence and Machine Learning Research?," AI Magazine, vol. 46, Art. no. e70004, 2025.
- [28] S. Singh, "Reliability and Admissibility of AI-Generated Forensic Evidence in Criminal Trials," arXiv:2601.06048, 2025.
- [29] Council on Criminal Justice, Task Force on Artificial Intelligence, "Assessing AI for Criminal Justice: A User Decision Framework," Council on Criminal Justice Report, 2026.
- [30] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Proc. NeurIPS, vol. 33, pp. 9459-9474, 2020.
- [31] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Proc. EMNLP-IJCNLP, 2019, pp. 3982-3992.
- [32] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," Foundations and Trends in Information Retrieval, vol. 3, no. 4, pp. 333-389, 2009.
- [33] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," Proc. AISTATS, vol. 54, pp. 1273-1282, 2017.
- [34] F. Zappalà, G. Romano, and A. Caruso, "Enhancing Crime Scene Investigations through VR and Deep Learning," arXiv:2409.18458, 2024.



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 9940 572 462  6381 907 438  ijirset@gmail.com



www.ijirset.com

Scan to save the contact details