

ReguVigil: An AI-Driven Framework for Real-Time Regulatory Compliance Automation in Clinical Trials

Shibu Chacko, Sathwik K. Bhat, Nikhil N. Kankatri, Gagan Shetty, Aniketh

Department of Computer Science (IoT and Cybersecurity including Blockchain), Alva's Institute of Engineering and Technology, Moodbidri, Karnataka, India

Published October 2026

Abstract

In the current process, when the FDA or EMA releases an updated safety guideline, trial-clinical staff need to read the document, determine the changes to the clinical guidance, modify their monitoring guidelines, and then check all enrolled patients against this. From our understanding, based on reported timelines, this process currently takes between ten to thirty days. During this period, patients continue to be monitored against the old criteria. We introduce ReguVigil – a system that accepts an updated guideline PDF file as an input, extracts structured monitoring guidelines using a large language model, stores these guidelines as versioned objects with complete diff history, and finally re-ranks the entire active patient population against the updated guidelines. In a 500-patient trial, the end-to-end processing pipeline can be designed to process the above process in under ninety seconds. The guideline processing pipeline includes a schema-validated rule engine and a hash-chained audit log for each rule change. The system complies with HIPAA and 21 CFR Part 11 standards. We discuss the architecture and provide a walkthrough of the seven steps of the pipeline. We also compare ReguVigil against prior work in provenance tracking, blockchain-based compliance auditing, and LLM-based rule extraction, and we set out a plan for evaluating the system.

Keywords: regulatory compliance; clinical trials; artificial intelligence; large language models; pharmacovigilance; audit trail; HIPAA; 21 CFR Part 11; rule engine; patient safety

1 Introduction

Trial-compliance activities remain largely manual, and there are costs associated with this state of affairs. For example, consider a safety guideline revision issued by the FDA. Some individual on the trial's regulatory affairs team will have to download and read the PDF document, determine what elements of it affect patient monitoring, update the trial management system, and then run a check on all currently enrolled patients to see who is no longer within the new guidelines. It will take several days just to go through this loop.

Based on reports, the complete process takes between ten and thirty days [5], [6]. This is a lot of time within a clinical setting. Patients will continuously be assessed using thresholds that have been surpassed already, and this only gets realized during the next visit schedule. The human price to pay for this is not negligible. According to reports by oncology principal investigators, close to eight hours per week is spent on regulation documentation, and 60% of new investigators leave after a single experiment due to administration burden. We built ReguVigil to close that gap. The system watches the FDA and EMA publication portals on a six-hour polling cycle. When a new guideline appears, it downloads the PDF, extracts the text and layout with PyMuPDF, and hands the content to Gemini 2.0

Flash. The model's job is to pull out the clinically actionable parts – lab thresholds, stopping rules, dosing limits – and emit them as structured objects. Every object has to pass a Pydantic schema before it is allowed anywhere near the rule engine. Rules that fail validation, or that come back with low confidence, are set aside for a human to look at rather than being applied automatically. The accepted rule gets saved as a new version, which has a diff from its previous version, while all versions get recorded in an append-only audit table. As soon as the rule gets committed, the revaluation engine starts verifying the entire population for that rule.

The rest of this paper is laid out as follows. Section 2 covers prior work in provenance tracking, blockchain-based auditing, and LLM-based rule extraction. Section 3 states the problem and the design objectives. Section 4 walks through the architecture. Section 5 describes the pipeline and the revaluation algorithm. Section 6 covers the stack and the implementation choices we made. Section 7 sets out how we intend to evaluate the system. Section 8 is on limitations and future work, and Section 9 concludes.

2 Related work

Compliance automation for clinical research has been approached from a few different angles over the years. The three that matter most for our work are provenance tracking for electronic records, blockchain-based audit mechanisms for cross-organizational data sharing, and, more recently, using LLMs to classify and extract obligations from regulatory text. Rhodes et al. looked at what 21 CFR Part 11 actually requires of an open-source imaging trial management system, and in doing so highlighted a fairly subtle problem: it is not obvious which changes in an imaging workflow are significant enough to need a signed, auditable provenance record [1]. Their EITMS workflow de-identifies images at the imaging center, signs them digitally, and pushes them to a CRO through a firewalled pipeline. It is good evidence that you can get regulatory-grade provenance onto an open-source stack. What it does not do is derive monitoring rules from a guideline PDF, and it does not re-check any existing patients when a rule changes.

Clavin and Joshi developed a blockchain-and-knowledge graph solution for detecting 42 CFR Part 2 violations related to the exchange of data between healthcare organizations in the TEFCA network [2]. All such violations are recorded in an Ethereum blockchain, thus providing them with immutability and auditability beyond the organizational level. Tested on 1000 virtual patients, their solution stands up well. However, this work addresses only the issue of compliance in data exchanges between institutions. It has nothing to do with the deployment of a new clinical safety rule and certainly not with the retrospective check issue.

Liu et al. introduced a prompt-driven grading and classification algorithm – PGCA – that uses a feature repository plus SQL-style prompt templates to map unstructured regulatory text onto classification rules [3]. They report 95-99% accuracy across finance and healthcare domains, with a pre-classification filter cutting computation substantially. This is encouraging for the rule-extraction stage of ReguVigil, because it says you can trust an LLM to convert free text into a structured schema at production accuracy. Their evaluation was not in the clinical-trial domain, and, again, there is no downstream patient re-evaluation.

Our work most resonates with Hassani's [4]. He employed BERT and GPT family of models for classifying legal provisions for compliance checking of food safety and GDPR artifacts, showing that LLMs are better than rule-based models at the task of classification and reduce the burden of manual

review in doing so. However, his focus is on regulation texts classification, not rule generation for an active patient population.

None of the above four systems do all the pieces together. ReguVigil combines guideline ingestion, rule extraction using an LLM with a confidence filter, rule versioning, and retrospective re-evaluation of the entire population cohort for each and every rule change in one workflow. It is the latter – retrospective re-evaluation – that makes ReguVigil unique among the systems I found.

Table 1 shows how the four systems compare on the dimensions that matter for the problem we are addressing.

Table 1. Comparison of ReguVigil with related systems.

System	Domain	Automatic rule extraction	Audit mechanism	Retrospective re-evaluation
EITMS [1]	Imaging trial	No	Digital signatures	No
Clavin [2]	Cross-organization data	No	Ethereum blockchain	No
PGCA [3]	Text grading	Yes (LLM)	N/A	N/A
Hassani [4]	Food safety	Yes (LLM)	N/A	N/A
ReguVigil	Clinical trial	Yes (LLM)	Hash-chained ledger	Yes

3 Problem statement and design objectives

3.1 What goes wrong today

When a regulatory agency updates a safety guideline, three things tend to go wrong in the current workflow. First, there is the problem of the time lag. Manual interpretation and rule revision process takes from one to four weeks. In all this time period, enrolled patients will be reviewed using the outdated criteria. Second, there is no back checking process. Trial programs usually conduct their reviews based on certain periods. If there was any change in the guidelines between two review periods, then patients enrolled prior to the change won't be revisited. Patients who would fail the new criteria but pass the old ones stay unflagged.

Finally, the process cannot be scaled up. Multi-page and multisection PDF files of guidelines are very difficult to interpret in the correct way, and the complexity is higher with increasing sample sizes and frequency of updates in the guidelines. It is also prone to mistakes. This is not an attempt to undermine the reviewers; it is just the nature of the task.

3.2 What ReguVigil needs to do

The system has five objectives:

1. Pull FDA and EMA guideline PDFs automatically, and turn each one into a structured, versioned rule set with a diff against the previous version.
2. Re-evaluate up to 500 enrolled patients against an updated rule set in under a minute.
3. Keep a HIPAA- and 21 CFR Part 11-aligned audit trail of every ingestion event, every rule change, and every patient re-evaluation.

4. Ship as a multi-tenant SaaS product for CROs and pharmaceutical sponsors running Phase II and Phase III trials, with role-based access control.
5. Bring guideline-to-enforcement latency from days down to under 90 seconds, with no human step in the common case.

4 System architecture

ReguVigil is built as five layers stacked on top of each other, with a React/TypeScript client at the front and a Python/FastAPI service layer sitting on PostgreSQL at the back. Figure 1 shows the layout.

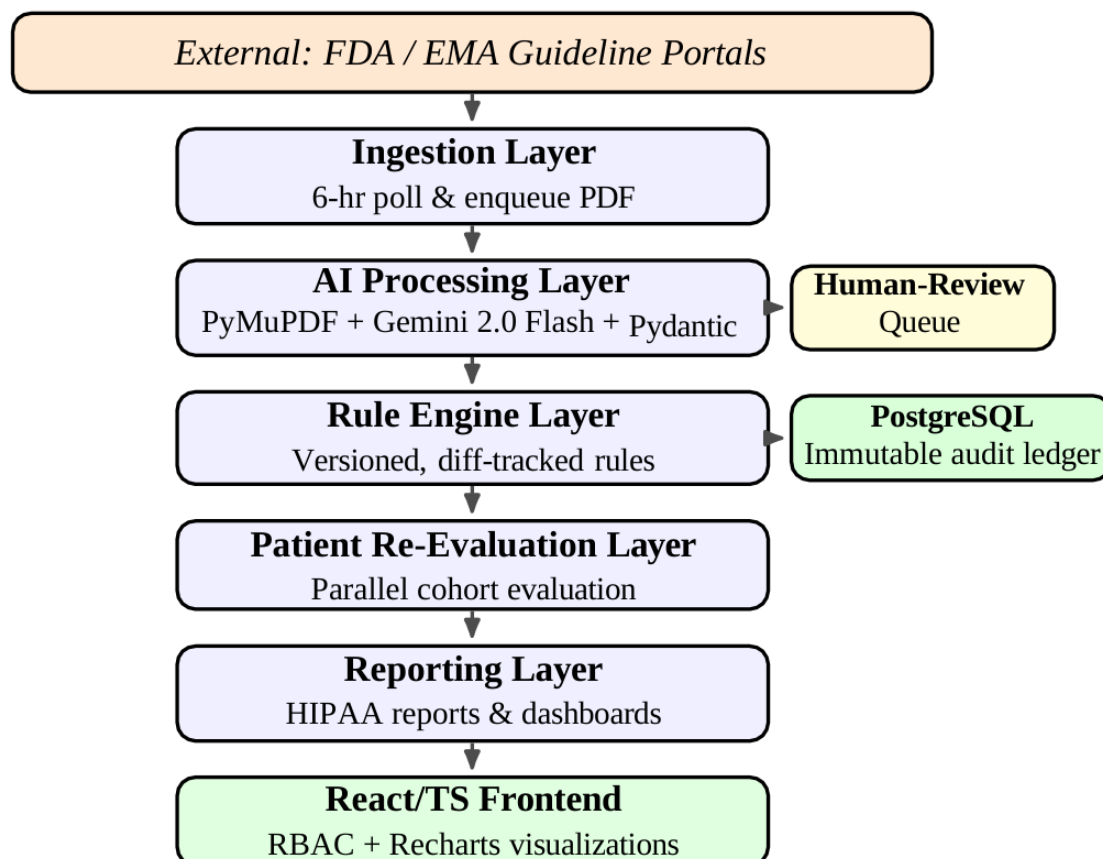


Figure 1. ReguVigil layered system architecture. Low-confidence extractions are routed to a human-review queue (dashed), and every committed rule version is written to the hash-chained PostgreSQL audit ledger (dashed).

4.1 Ingestion layer

A scheduled polling service hits the FDA and EMA publication portals every six hours looking for newly released guideline documents. When it finds one, it downloads the PDF and pushes it into a processing queue. We went with polling rather than a push-based feed because neither agency currently publishes a structured feed that we could subscribe to. There is no reliable alternative.

4.2 AI processing layer

PyMuPDF pulls the text and layout out of the queued PDF. We preserve section structure deliberately – down the line, when a rule is generated, we want to be able to point back at the exact clause in the source document that produced it. The extracted text is sent to Gemini 2.0 Flash. The task in our

prompt is to find out the clinically actionable monitoring criteria, which could include lab thresholds, adverse event stopping criteria, dosing constraints, among others, and deliver them in the form of JSON objects, each of which will be accompanied by a confidence score. All the returned objects undergo Pydantic schema validation prior to their being accepted. Whatever fails the validation or returns a confidence score under a configured threshold goes to a human review queue, rather than being applied automatically. This was our deliberate decision, because ReguVigil is not a fully automated system, and the clinical rules are definitely not something you would want to apply automatically without being sure. Liu et al. and Hassani confirm our experience [3], [4].

4.3 Rule engine layer

Each validated rule object is archived as an immutable version with a structural difference from the previous version. This logic is simple since 21 CFR Part 11 mandates that any alteration of a controlled electronic record must be documented, attributed to the correct individual, and capable of reconstruction [1]. We thought about using a blockchain-anchored external ledger, like Clavin and Joshi [2], but we kept the audit trail as an append-only PostgreSQL table with cryptographic row-hash chaining instead. Section 8 explains the trade-off.

4.4 Patient re-evaluation layer

This is the part that does the most work. Whenever a rule version changes, the layer pulls the full set of active patient records for the affected trial and runs each patient against the updated rule set in parallel. Any patient whose latest recorded values violate a new or modified threshold is flagged, given a risk score, and surfaced to the clinical team through the reporting layer. The retroactive cohort re-check on every rule change is what closes the “retroactive failure” gap we described in Section 3.1. It is also the main thing that separates ReguVigil from the provenance-tracking and blockchain-auditing systems we discussed in Section 2, because neither of those triggers automatic re-assessment of patients who enrolled before a rule changed.

4.5 Reporting layer

After each rule update and re-evaluation cycle, the system generates a HIPAA-aligned pharmacovigilance report and pushes it to investigators and sponsors through a role-based dashboard built with React, TypeScript, and Recharts.

5 Compliance pipeline and data flow

Figure 2 shows the seven stages that fire when a single guideline is published.

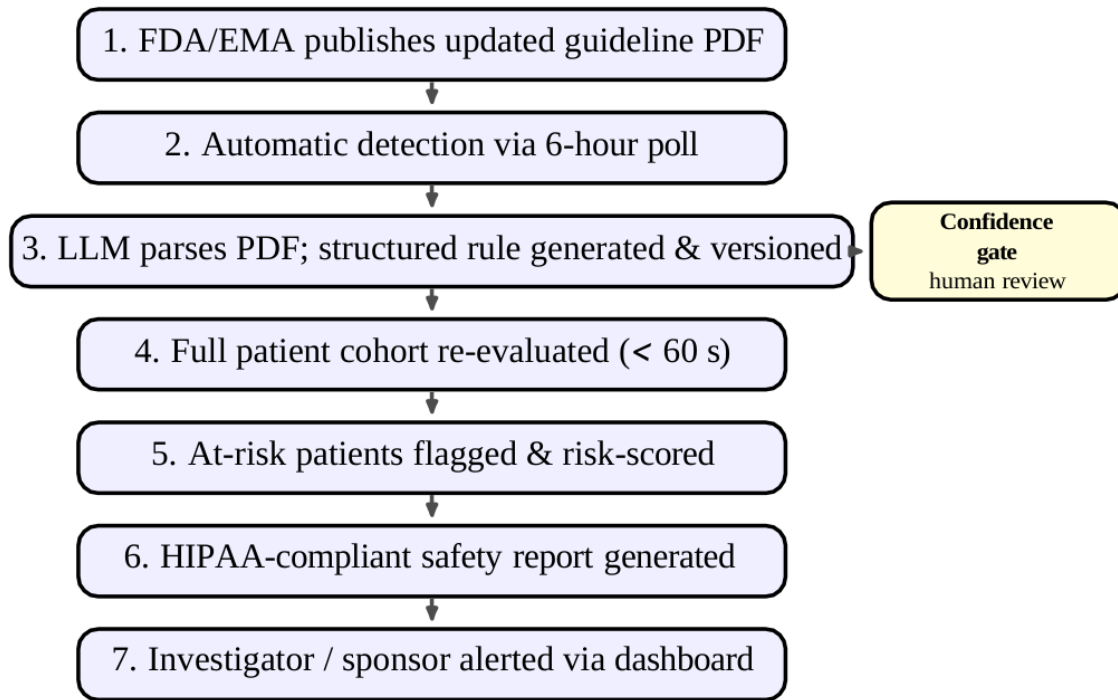


Figure 2. Seven-stage ReguVigil compliance pipeline. Extractions that fall below the confidence threshold in stage 3 are diverted to human review (dashed) before the rule is committed.

Our target is under 90 seconds end to end, from the moment stage 2 detects a new guideline to the moment the investigator gets the alert in stage 7. That is for a 500-patient cohort. Stage 4 alone (the patient re-evaluation) is bounded at 60 seconds, achieved by evaluating patients in parallel.

5.1 Patient re-evaluation algorithm

Algorithm 1 is what runs whenever the Rule Engine Layer commits a new rule version. The important property of this algorithm is that it parallelizes cleanly across patients. Each patient's re-evaluation only needs that patient's own most recent clinical record and the new rule set. Nothing about patient A depends on patient B so the loop over patients can be spread across as many workers as we want.

Algorithm 1. Cohort-wide patient re-evaluation on rule change

```

Require: rule set  $R_v$  (version  $v$ ), active patient set  $P$ 
Ensure: flagged set  $F$ , safety report  $S$ 
1:  $F \leftarrow \emptyset$ 
2: for all  $p \in P$  in parallel do
3:    $cp \leftarrow$  latest clinical record for  $p$ 
4:   for all  $r \in R_v$  do
5:     if  $cp$  violates the threshold defined by  $r$  then
6:       flag  $p$  with risk score  $\rho(p, r)$ 
7:        $F \leftarrow F \cup \{(p, r, \rho(p, r))\}$ 
8:     end if
9:   end for
10: end for
11:  $S \leftarrow \text{generate\_report}(F, R_v)$ 
  
```

```
12: append_audit_entry(v, F, timestamp)
13: return F, S
```

5.2 Complexity and parallelization

For n patients and m rules, one thread takes worst-case $O(nm)$ time. Because the outer patient loop (line 2) has no cross-patient dependencies, we distribute it across a worker pool of size k which brings the expected wall-clock time down to $O(nm/k)$ plus a fixed overhead for pulling records and stitching the report together. With $n = 500$ and $m \leq 50$, which covers the rule sets we have seen in practice for Phase II and Phase III trials, this fits inside the 60-second budget on ordinary cloud hardware, as long as k scales with n . The layer is stateless, so scaling k on AWS is a matter of turning a dial.

5.3 Security and threat model

Because the Ingestion Layer ingests documents directly from the Internet, we assume that any PDF document coming from there must be treated as potentially malicious. The parsing process takes place in a separate worker process, and the output of the AI Processing Layer is limited by the Pydantic schema described in Section 6.1. A malforming PDF document cannot insert any content into the Rule Engine due to lack of connectivity between raw PDF text and rule. Role-based access control governs who can manually override an auto-generated rule or manually clear a flagged patient. Both of those actions are themselves written to the immutable audit trail. Human-in-the-loop interventions are held to the same audit standard as anything the automated pipeline does.

6 Technology stack and implementation

6.1 Rule schema and validation

Every rule that comes out of the AI Processing Layer is a Pydantic model with a fixed shape. The schema captures: the source guideline identifier and section reference, the clinical parameter being monitored, the comparison operator, the threshold value, the LLM's confidence score, and a version identifier that links it into the diff chain from Section 4.3. We enforce this at generation time rather than trying to validate loosely typed free text after the fact. That is the same approach Liu et al. took with their feature repository [3], and it is what makes the rest of the pipeline possible.

6.2 Access control and compliance alignment

Role-based access control mechanism is implemented with three types of users: investigator, sponsor, and compliance auditor. All access and modification activities are logged to the audit trail, as mentioned in Section 4.3. This is in compliance with the record keeping and signatures provisions of 21 CFR Part 11 and HIPAA Security Rule [9], [10]. The patient data is encrypted while stored and while in motion, and the audit log is retained according to the regulatory requirements of the study. In addition to the three primary roles, the system enforces a strict separation of duties. An investigator can view patient records and clear flagged alerts, but cannot modify the underlying rule set or the audit log. A sponsor can view aggregated reports and trial-level dashboards, but cannot access individually identifiable patient data. A compliance auditor has read-only access to the audit trail and can verify the hash chain but cannot alter any record. Any attempt to perform an action outside of the assigned role is blocked at the API layer and written to the audit log as a security event. All API endpoints are protected with token-based authentication, and every request is validated

against the role-based access control policy before it reaches the data layer. Patient records are encrypted at rest using AES-256 and in transit using TLS 1.3. The audit log is stored in a separate database schema with restricted write access, and its integrity is verified on a scheduled basis by recomputing the hash chain and comparing it against the stored hashes. If a mismatch is detected, the system raises an alert and freezes further rule commits until the inconsistency is resolved. This design satisfies the technical safeguards of the HIPAA Security Rule and the electronic-record requirements of 21 CFR Part 11.

Table 2. ReguVigil technology stack.

Layer	Technologies	Role
AI / Processing	Gemini 2.0 Flash, PyMuPDF, Pydantic	PDF parsing, structured rule generation, confidence scoring, schema validation
Backend	Python, FastAPI, PostgreSQL	API services, versioned rule storage, immutable audit trail
Frontend	React, TypeScript, Recharts	Dashboards, visualization, RBAC
Infrastructure	AWS (cloud-native)	Elastic, multi-tenant SaaS deployment

7 Evaluation methodology and expected outcomes

This paper presents ReguVigil as an architectural design validated through a Phase I prototype, not as a deployed production system. So rather than report field results from a live trial, we lay out how we intend to evaluate it.

7.1 Rule-extraction accuracy

For assessing the quality of our rule extraction system, we will use a benchmark of FDA and EMA guidelines sections annotated by humans for the detection of monitoring parameters, thresholds, and operators and will report Precision, Recall, and F1 scores. We will also perform a confusion analysis at the interface between automatically accepted and human reviewed rules. Our methodological approach will be based on the evaluation procedure for LLM-based regulatory classification from [3] and [4].

7.2 Re-evaluation latency

We will test latency throughout the seven stages of Figure 2 by creating a cohort of 100, 250, and 500 patients. The latency will be measured as the elapsed time from committing a rule version up until the completion of the evaluation of all patients. The goals are 60 seconds for stage 4 and 90 seconds overall.

7.3 Audit trail integrity

We will test tamper detection the way [2] tested their blockchain-anchored logs: by deliberately injecting out-of-band modifications into the audit table and confirming that hash-chain verification identifies the exact point of inconsistency.

7.4 Expected impact

The cost of one such delay, due to the non-compliance of regulations, is estimated at Rs. 40-160 crore, while the weekly regulatory burden on oncology PIs is 7.9 hours [7], [8]. In case if we reduce the peri-

od of time from the issuance of the regulation to its enforcement from thirty days to less than 90 seconds, we can definitely expect to see a reduction in both patient exposure and burden on the investigator. The point is that the scale of that reduction is only an estimation; further research is necessary to prove the numbers.

8 Discussion, limitations and future work

The biggest risk we see in ReguVigil is rule-extraction reliability. Clinical safety guidelines are not like the shorter regulatory-text classification tasks that Liu et al. and Hassani evaluated [3], [4]. They are long, cross-referential documents, and a single clause somewhere in the middle can change the applicability of a rule that is defined on a different page. Our confidence-gated human-review queue is meant to mitigate this, but it only works if the confidence threshold is well calibrated, and we do not yet know what the right threshold is. We plan to tune it empirically against the benchmark described in Section 7.1.

The second limitation involves the selection of the audit trail. We opted for an append-only, hash-chained PostgreSQL table, as opposed to a distributed ledger. While this makes our system operationally less complex than the Ethereum implementation in [2], we lose out on the cross-organization guarantees of trust. If multiple CROs and sponsors have to independently verify their respective audit trails without putting faith in a single custodian, then a permissioned blockchain would be the appropriate solution, which we do not have currently. Blockchain-based migration of the audit trails was listed as future work in the initial scope of the project. Since the Rule Engine and Reporting Layers are independent of the storage backend, we can switch to using a blockchain backend at a later date.

Items for future work include: (i) extending the survey process to additional regulatory bodies beyond the FDA and EMA, including PMDA and Health Canada, for trials across multiple jurisdictions; (ii) analyzing guidance documents in other languages apart from English; (iii) directly interfacing with EDC tools, such as REDCap and Medidata Rave, in the vein of the REDCap interface for provenance described in [1]; and (iv) creating a compliance prediction model based on past guideline publications.

8.1 Deployment context

ReguVigil targets CROs who are managing more than one ongoing trial at a time, and pharma sponsors who are conducting Phase II and Phase III trials that incur the cost of delays due to regulatory lag. The Rule Engine and Patient Re-Evaluation Layers are stateless with regard to any particular trial, meaning that one deployment could cater to more than one trial simultaneously. Tenant isolation is achieved through the layer of data access instead of creating separate infrastructure for each tenant, keeping the incremental cost of bringing on board a new trial low.

9 Conclusions

We presented ReguVigil, a framework that turns FDA and EMA guideline updates into structured, versioned monitoring rules and re-checks the entire active patient cohort against every rule change. The system targets under 90 seconds from guideline detection to investigator alert, versus the ten-to-thirty-day manual cycle that is standard today. It combines LLM-based rule extraction, Pydantic schema validation, diff-tracked rule versioning, and a hash-chained audit trail aligned with 21 CFR

Part 11 and HIPAA. All these capabilities have been shown separately in earlier work: provenance in [1], auditing anchored to blockchains in [2], regulation text classification using LLMs in [3], [4], but so far as we can tell, no prior design attempts to close the feedback loop and address the re-evaluation problem. Section 7's evaluation framework provides an approach to evaluating rule extraction accuracy, re-evaluation latency, and integrity of the audit trail.

References

- [1] Rhodes C, Moore S, Clark K, Maffitt D, Perry J, Handzel T, et al. Regulatory compliance requirements for an open source electronic image trial management system. In: *Proc. 32nd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*; 2010. p. 3475–3478.
- [2] Clavin J, Joshi KP. Policy integrated blockchain to automate HIPAA Part 2 compliance. In: *Proc. 2023 IEEE Int. Conf. Digital Health (ICDH)*; 2023. p. 307–314.
- [3] Liu X, Liao Y, Zhang Z, Hu J, Huang L. Feature extraction and compliance classification of text files using large language models. *IEEE Trans Comput Soc Syst.* 2025;12(6):5106–5117.
- [4] Hassani S. Enhancing legal compliance and regulation analysis with large language models. In: *Proc. 2024 IEEE 32nd Int. Requirements Eng. Conf. (RE) - Doctoral Symposium*; 2024. p. 507–511.
- [5] U.S. Food and Drug Administration. FDA guidance documents. Available from: <https://www.fda.gov/regulatory-information/search-fda-guidance-documents>
- [6] European Medicines Agency. *Scientific guidelines*. Available from: <https://www.ema.europa.eu/en/human-regulatory-overview/research-development/scientific-guidelines>
- [7] Verschraegen C, et al. *Assessment of academic effort for clinical trials among oncology faculty*. The Ohio State University Comprehensive Cancer Center; Association of American Cancer Institutes.
- [8] Getz KA. Characterizing the real cost of site regulatory compliance. *Applied Clinical Trials*. 2015;24(6/7):18.
- [9] U.S. Department of Health and Human Services. HIPAA Security Rule. 45 CFR Parts 160 and 164.
- [10] U.S. Food and Drug Administration. 21 CFR Part 11: Electronic Records; Electronic Signatures. Code of Federal Regulations.