Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Synthetic Clinical Data for Privacy-Preserving Research

Allan Zablon

allanzablon@gmail.com

Aga Khan University

Method Article

Keywords: synthetic clinical data, clinical narratives, electronic health records, privacy-preserving research, configurable generation, data integrity, span annotation

Posted Date: November 1st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11512016/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Abstract

Background

Synthetic clinical records and narratives can support research and information-system testing without routine access to identifiable patient files. A reusable generator must coordinate record relationships, encounter context, and configurable documentation conventions. Fluent prose and relational consistency alone do not establish clinical fidelity, privacy protection, or reliable training labels.

Objective

To describe a configurable framework for linked synthetic clinical records and unstructured notes, evaluate the integrity of archived outputs, and verify reference functions for reproducibility and annotation. We also specify an extension toward a virtual electronic health record and continual, on-demand corpus generation.

Methods

Five stages construct patient anchors, encounters, operational events, demographics, and coverage-guided clinical notes. We examined an archived census of structured exports and two disjoint narrative sets, reproduced isolated implementation fixtures, and tested deterministic seeds, bounded timing, export checks, exact span compilation, and patient-level partitioning in a separate reference module. Python notebooks combine structured sampling with four-bit MedGemma inference. The author-reported generation environment used virtual machines with dual-GPU cloud accelerators. Historical outputs and revised software were evaluated separately; system extensions are presented as proposals.

Results

The archived census recorded 50,000 patients, 3,225,625 unique encounters, and 375,000 operational events covering 250,000 visits. All sampled encounter counts agreed with patient anchors, and operational patient-visit mappings were valid. The 300 narrative documents represented 250 patients and visits. Strict encounter ordering held, although 341 intervals were shorter than 60 seconds. In the 200-document narrative set, 175 texts contained Markdown fences and 30 contained standalone missing-value tokens. All 39 reference-module tests passed, including 1,000 constructed timing cases. This establishes selected software contracts; no new language-model corpus, clinical assessment, or trained de-identification benchmark was produced.

Conclusions

Explicit record contracts and coverage planning provide a reusable basis for synthetic clinical data generation. The evidence establishes selected structural properties and bounded software behaviors in one evaluated configuration. The architecture could supply a virtual EHR and repeated narrative releases, but clinical fidelity, sustained operation, privacy risk, and transfer to other settings require independent evaluation.

1 Introduction

A synthetic clinical note is part of a larger information structure. It describes a patient at a particular encounter, refers to events and services, and expresses identity and contact information in narrative form. A useful generator must coordinate these representations. Producing a convincing paragraph is insufficient if its patient belongs to a different visit, its birthdate follows the encounter, or its identifiers disagree with the structured record.

A reusable generator must separate construction logic from deployment-specific resources. Naming conventions, administrative hierarchies, identifier formats, languages, and documentation practices differ between health systems. Applications also differ: an information-system test may prioritize linkage and schema conformance, whereas an annotation task needs controlled expressions and reviewed text spans. Configuration must therefore identify the intended use, field semantics, and validation rules rather than assume that one set of values or one quality score serves every application.

Clinical de-identification provides the identifier-focused use case examined in this implementation. For that task, a training example comprises final text, an explicit redaction policy, and reviewed character spans. Direct identifiers, contextual quasi-identifiers, and sensitive health attributes require separate definitions. An identifier or contact number has a different task role from height, weight, language, or religion; placing an attribute in a generation payload does not automatically make it a redaction target. Other applications can use the same linked records and narratives with different labels and acceptance criteria.

The framework is intended for adaptation across health systems through versioned resource packs and configuration. These supply field vocabularies, naming pools, geographic hierarchies, identifier rules, document types, and scenario parameters while retaining explicit patient and encounter contracts. Architectural configurability does not establish empirical generalization. The present results concern one implementation and its archived outputs; use in another language, specialty, institution, or jurisdiction requires appropriate resources and a new evaluation.

Existing research establishes important foundations. Synthea constructs synthetic patients and electronic health records through modeled processes. [1] Poulett et al. combine structured patient generation with patient journeys and language-model notes. [2] Yan et al. show why synthetic EHR generators require multiple evaluation measures. [3] These approaches motivate a specific construction question: how can a reusable record-to-narrative workflow control requested entity coverage while preserving clear distinctions between structural validity, clinical fidelity, and downstream usefulness?

We present a framework that links relational construction to configurable narrative generation. Record contracts maintain patient-visit dependencies and temporal bounds. Deployment resources supply identity and administrative values. Coverage deficits determine selected narrative requests, and bounded generation attempts retain explicit outcomes. We examine an archived export census and independently test a revised contract module. These evidence streams answer different questions and are reported separately.

The methodological contribution is a contract-based workflow coupling linked record construction, deficit-driven narrative coverage, and recoverable candidate labels. We derive a necessary capacity condition for requested coverage and distinguish planned payloads, implemented validator counts, and independently verified entities. This positions the contribution in coordination and verification rather than in a particular geography or hardware platform. The empirical contribution is an integrity study with bounded software verification; virtual-EHR integration and continual generation are specified as extensions rather than reported deployments.

2 Related work

2.1 Patient simulation and clinical narrative generation

Synthea provides a precedent for constructing linked synthetic patient records before their use in downstream applications. The structured generator described here instead uses explicit workload and temporal sampling rules. Its encounter history represents a relational and administrative scaffold rather than a disease-progression simulation. [1]

Poulett et al. emphasize journey-conditioned notes and consistency across a hospital stay. Their pipeline demonstrates that staged construction, local configuration, and narrative validation are established approaches. Our emphasis is the interface between relational contracts and category-directed generation, including the difference between requested entities and measured coverage. [2]

MedSyn uses structured medical knowledge to guide synthetic text and evaluates utility through a defined prediction task. It illustrates both knowledge-conditioned generation and the importance of task-specific evaluation. The present framework controls patient linkage, administrative context, and narrative entity targets; it does not reproduce MedSyn's knowledge graph or downstream experiment. [4]

2.2 Tabular fidelity and use-specific evaluation

Yan et al. evaluate synthetic EHR generation through multiple measures rather than a universal quality score. [3] For the present framework, linkage agreement, operational coverage, entity realization, clinical plausibility, distributional fidelity, and downstream utility require separate endpoints. A large structured export can satisfy key constraints while its associated narratives remain unsuitable for a particular training or simulation task.

2.3 Entity annotation and privacy

The i2b2/UTHealth annotation work distinguishes identity surrogates from reviewed narrative spans through double annotation and arbitration. A payload provides candidate values, while an annotated corpus identifies where and how those values occur. [5] This distinction is central when a language model can abbreviate, alter, omit, or introduce identifiers.

Synthetic text also requires a defined privacy assessment. Sarkar et al. motivate separating the utility of generated notes from their exposure to inference attacks. [6] The present method constructs fictional structured values, but it does not implement a formal differential-privacy mechanism. Hash-derived seeds support deterministic computation; differential privacy concerns changes in output distributions under neighboring input datasets. [7]

2.4 Synthetic text for deidentification training

Li et al. evaluated synthetic clinical notes through entity recognition on natural clinical text, demonstrating why task performance must be measured separately from apparent readability. [8] Dernoncourt et al. provide an established neural de-identification baseline, with explicit token and entity predictions rather than a document-level realism score. [9] These studies motivate one downstream evaluation for the framework; their performance estimates cannot be transferred to the present implementation without a comparable experiment.

ASQ-PHI combines synthetic identifying queries with hard negatives that resemble PHI, providing a useful precedent for testing over-redaction as well as missed identifiers. [10] RecordTwin generates context around anonymized entities extracted from clinical documents; it differs from the fictional patient-anchor construction used here because it requires information derived from original documents. [11] Cornelius and Rinaldi directly examine synthetic-to-real drift and de-identification transfer, showing why stronger control over a synthetic training distribution still requires external validation. [12]

The comparison with Poulett et al. is organized by purpose. Their journey-conditioned design targets longitudinal narrative coherence. Our design emphasizes configurable entity coverage, relational identity, and recoverable labels. These are different evaluation objectives, not evidence of comparative superiority. A larger encounter table provides no direct evidence that a model trained on its associated notes will generalize better.

3 Framework and reference implementation

3.1 Data model and construction order

Let P denote patients, E encounters, A operational events, D demographics, and N clinical documents. An encounter references one patient; an operational event and a document reference a patient–visit pair. Each document also has a distinct document key. An intermediate entity payload Q specifies candidate narrative values before a note is generated.

The construction order is P → E → A → D → N, with earlier stages supplying direct inputs to later ones. Demographics follows operational assignment because selected service and location signals constrain demographic choices. This is an engineering dependency: the resulting associations reflect the configuration and should not be interpreted as independently learned population relationships.

Table 1
Core contracts between representations.

Artifact	Essential content	Contract
Patient anchor	Patient key, observation endpoint, encounter count	Unique key and defined observation interval
Encounter	Patient key, visit key, start time	Unique visit and valid patient membership
Operational event	Patient–visit pair, event time, facility and setting	Correct pair, temporal relationship, and setting semantics
Demographics	Patient key, identity, birthdate and contextual attributes	One patient record and explicit field authority
Clinical document	Patient–visit pair, document key, narrative and status	Traceable context and defined generation outcome

A valid patient key and a valid visit key are not sufficient: the pair must occur together in E. Similarly, valid operational rows do not establish coverage of every encounter. Pair integrity and upstream completeness are measured separately, using the selected processing set and the full encounter set as different denominators.

3.2 Patient anchors and encounter expansion

The reference cohort configuration samples 50,000 fictional patient anchors. Patient identifiers encode a registration date and a numeric suffix, and the observation endpoint bounds subsequent encounters. The simulated period begins in 2200, with endpoints before the configured end of 2217. Date serialization is part of the inter-stage contract because downstream components recover registration information from the identifier.

Encounter counts follow a mixture of inclusive discrete uniforms with weights weights 0.75, 0.15, 0.08, and 0.02 and ranges 4–60, 60–150, 150–300, and 300–400. These are workload settings rather than fitted population estimates. The component means are 32, 105, 225, and 350, yielding an expected 64.75 encounters per patient and 3,237,500 for 50,000 patients. Actual exported counts are reported separately in Section 5.

For registration date r_i, observation endpoint m_i, and encounter count N_i, normalized beta offsets are sorted and mapped into the observation interval:

$$u_{i,(1)} \le \cdots \le u_{i,(N_i)} \sim \mathrm{Beta}(1.5, 1.2), \quad t_{i,(k)} = r_i + u_{i,(k)}(m_i - r_i), \quad k = 1, \ldots, N_i.$$

The endpoint is an upper bound, not necessarily an event time sampled by the generator. A non-increasing anchor is advanced one minute beyond its predecessor. Positive gaps below a minute remain unchanged, so ordering, minimum separation, and interval membership require distinct checks.

Visit identifiers combine a routing prefix, a timestamp component, and a counter maintained by prefix and second. The counter supports uniqueness within its execution scope. The encounter implementation constructs rows in memory before writing 250,000-row partitions; reproducing identifier assignments therefore requires attention to processing order as well as seeds. Parallel implementations need coordinated counters or identifiers with an explicit shard component.

3.3 Operational context and demographic assignment

The operational stage derives facility and service context from encounters and configured routing rules. Where explicit inpatient information is absent, eligible facilities use an inpatient probability of 0.007. A designated facility receives outpatient location and service context. Its exported setting flag is computed separately, creating the decision-order inconsistency documented in Appendix B. This limitation is retained in the archived source snapshot. The revised module in Section 3.8 supplies a single setting decision for integration; the archived exports were not regenerated from it.

Each selected visit receives one or two operational events. Their times advance cumulatively after service time by 3–89 minutes per increment for outpatient settings and 20–479 minutes for inpatient settings. Outpatient context nulls inpatient-specific fields; inpatient context assigns ward, service, room, bed, and a sampled expected length of stay. Sparse administrative attributes and conditional missingness provide examples of incomplete operational documentation. This stage models operational snapshots, not a complete admission-to-discharge state machine.

The effective development configuration limits operational processing to three encounter files and 250,000 visits. This cap is part of the processing scope. An evaluation should compare operational coverage both with selected encounters and with the full structured history, rather than treating intentionally unprocessed encounters as unexplained missing records.

The demographic engine aggregates patient-level visit and service signals and uses attribute-specific random streams to assign names, birthdates, geographic categories, and selected social attributes. Geographic base weights can be adjusted by context and normalized. For configured administrative area j, initial weight w□, and context multiplier a□(x□), the sampling rule is:

$$Pr(C□ = j \mid x□) = w□a□(x□) / \Sigma □ w□a□(x□).$$

Geographic weights are scenario parameters; their final conditional probabilities depend on context multipliers and normalization. Naming pools and language, belief, and social-attribute profiles also encode design assumptions. They are replaceable configuration resources, not fitted population estimates. Pediatric and maternity signals constrain selected demographic choices, while birthdates

require validation at registration and service time. Any target population requires separate calibration and assessment of unintended associations.

Operational contacts, demographic values, and note payloads can be generated by different modules. A shared patient key does not make these values identical. A field-authority map should identify which representation supplies the authoritative patient value, which fields may vary by encounter, and which attributes are illustrative. The same distinction determines which cross-table differences are errors and which require contextual interpretation.

3.4 Narrative contexts and entity payloads

The note loader normalizes schema variants and timestamps, resolves duplicate encounters using its configured precedence, and can operate without optional ADT context. The archived reference selection rule retains one visit per patient to favor patient diversity. Its intended weighted ordering and date parsing have boundary behaviors listed in Appendix B. The evaluated narrative exports include another structure with two documents per selected visit; source configuration and archived output sets are therefore described separately.

Missing encounter types receive configured hash-partition labels. These fallback labels are independent of the earlier operational setting rule and need a reconciliation policy when both are available. Document-type assignment likewise needs to agree with the selected scenario. A stable deterministic label is a generated attribute, not an observed clinical event.

The payload combines patient-level identity with visit- and document-level values. It reuses patient names and selected demographic fields where recognized, sets the medical-record identifier from the patient key, and generates additional provider, contact, account, device, vehicle, and network values. Adaptation concerns the formats and meanings of these values, not translation of the prose prompt alone. A resource pack must document its allowed values, provenance, and intended clinical context.

The evaluated planner prioritizes sixteen additional categories, with document-count and distinct-value thresholds given in Appendix A. They include direct identifiers, quasi-identifiers, and sensitive contextual attributes. Their roles differ: height and weight are included for coverage experimentation rather than treated as universally equivalent to a name or identifier. The category set and thresholds describe this implementation; another use case must specify its own policy and validate any changed taxonomy.

3.5 Coverage planning and capacity

For category c, the planner maintains the number d(c) of accepted documents credited for that requested category and a set S(c) of credited representative payload values. Given document target D(c) and distinct-value target V(c), the deficits are:

$$g_{\square}(c) = max(0,\ D(c) - d(c)), \quad g_{\square}(c) = max(0,\ V(c) - |S(c)|).$$

Categories are ranked lexicographically by descending document deficit, descending distinct-value deficit, and category name. This greedy rule makes the next request inspectable. Because it uses raw deficits, larger absolute thresholds influence priority; it is not a proof of optimal target allocation.

The reference style schedule assigns rounded sequential shares of 45% baseline, 35% enriched, and the remaining identifier-heavy contexts. These are phases rather than independent draws. Baseline contexts receive selected social or anthropometric targets. Enriched and identifier-heavy contexts prioritize categories according to their rules, with a final cap of six targets. The final list after deduplication and capping defines the actual request.

The configured thresholds imply a useful capacity check. For an initially empty ledger, the sum of minimum document targets is 337 category–document assignments. An accepted document can credit at most six requested categories. Even an ideal allocation therefore requires:

$$6\,n_{\text{acc}} \;\geq\; 337, \quad n_{\text{acc}} \;\geq\; \mathrm{ceil}(337 \,/\, 6) \;=\; 57.$$

Here n_{acc} is the number of accepted documents. With at most one representative value credited per requested category and document, each category additionally requires at least max(D(c), V(c)) credited documents. These are necessary ledger-capacity conditions, not sufficient guarantees of convergence. Style restrictions, failed generation, omitted targets, and repeated values can increase the required budget. A two-document execution can test orchestration but cannot satisfy the default thresholds.

In the archived planner, only accepted requested targets with positive hits update the ledger. Incidental categories are not exhaustively counted. Furthermore, broad-family matches can credit a narrower target even when its specific payload value is absent. The ledger is therefore an intervention record; realized corpus coverage requires an independent annotation or matching procedure with category-specific semantics.

3.6 Prompting validation and bounded generation

The prompt combines patient and visit context, calculated age, document type, available operational events, the entity payload, and instructions for requested categories. Embedding instructions connect identifiers to plausible documentation contexts such as registration, communication, transport, and sign-off. The model is asked to express specified values within a clinical document, while document structure guides the form of the note.

The reference inference component is configured for the Unsloth four-bit quantization of MedGemma-27B text instruction weights. The foundation model and distributed artifacts are documented separately. [13] [14] [15] The wrapper specifies temperature 0.85, nucleus-sampling probability 0.95, at most 4,000 new tokens, and an input limit of 4,096 tokens. These are configuration settings; model and tokenizer revisions and actual token counts belong in the execution manifest.

The programmatic validator counts planned-value matches by broad family. Required families include patient and provider names, medical-record identifier, date, phone, and address; configured gates request at least eight families and eighteen candidate matches. Additional gates include 2,200–9,000 characters, repeated record-identifier occurrence across document regions, requested target hits, punctuation and shorthand thresholds, and a duplicate-sentence ratio no greater than 0.30. Candidate matches are not gold annotated spans, and these rules do not substitute for clinical review.

Up to four attempts use the same prompt and payload. Passing all configured gates produces an accepted candidate. When attempts fail, a heuristic combines mention, family, target, and issue counts to retain a diagnostic candidate; terminal generation exceptions remain separate. Retained text is consequently not synonymous with accepted text.

Algorithm 1

Archived coverage planning and bounded candidate selection.

Input ordered contexts X, coverage state C, model G, validator V, attempt limit R

For each context x at position i

Choose the style from rounded sequential cutoffs

Rank category deficits and form the capped target list T

Construct payload Q and a document identity for x

Assemble prompt P from x, Q, T and available operational events

Set history H empty and accepted candidate absent

For attempt a from 1 through R

Generate y from G using the same prompt P

Evaluate y with V against Q and T

Record attempt diagnostics or the generation exception

If y passes all gates

Select y as accepted and stop attempting

Otherwise retain the candidate and its score in H

If accepted candidate exists

Export with accepted status

Update C for requested categories with positive recorded hits

Else if H contains a generated candidate

Select its highest scoring candidate and export with failure status

Else

Record terminal generation failure without a text candidate

Output documents with status, attempt reports, coverage ledger and run summary

3.7 Reproducibility and configuration portability

The reference writer supports text files, document metadata, attempt reports, planning reports, and a run summary. A reproducible release must preserve requested and realized context counts, document status, model settings, effective configuration, and output checksums. The presence of writer code does not establish that those files accompanied the archived narrative sets. Patient-based sharding also requires global reconciliation: document counts can be summed over disjoint documents, whereas distinct values require set unions.

Many operational and demographic attributes use cryptographic digest-derived random streams with stage-specific serialization and reduction rules. Other components depend on sequential state, processing order, or interpreter configuration. Python string-hash behavior is documented independently. [16] Structured reproducibility and language-model sampling reproducibility therefore require different checks.

Adapting the method begins with intended use, field authority, reference-resource provenance, and the semantics of administrative and clinical fields. Domain experts should assess social-attribute assumptions, document eligibility, identifier density, and documentation conventions. A resource pack should be versioned separately from construction code so that changed outputs can be traced to changed resources or changed algorithms. Model cards and dataset datasheets provide established reporting structures. [17] [18] Boundary tests and pilot evaluation must accompany a new configuration.

3.8 Revised software contracts and annotation export

We implemented a separate reference module for the failure conditions that can be resolved without repeating model inference. It supplies stable record seeds, patient-level partition assignment, bounded encounter anchors, field-label access for sampling weights, strict parsing of the cohort identifier format, a shared setting decision, omission of missing prompt fields, and character-span compilation. The module is an integration target for the generation pipeline. Its tests do not imply that archived notebooks or narrative outputs have been repaired in place.

Stable seeds are derived from the first 64 bits of SHA-256 over a UTF-8 JSON serialization of a namespace, record identifier, and integer seed. Namespace separation prevents identical random streams being reused inadvertently for different attributes. Patient partitions use a distinct namespace and fixed thresholds for an intended 80:10:10 train-validation-test allocation. Actual partition counts must be reported; hash thresholds do not force those proportions in a finite sample. The assignment remains stable when rows are reordered. Shared identities, templates, or reference pools may still create lexical overlap across patient partitions.

For a requested minimum interval delta and n encounter anchors in a window of length L, the revised sampler first requires L >= (n − 1) delta. It samples n sorted offsets from the remaining slack interval and adds i delta to offset i, with zero-based i. This construction preserves the observation bounds and minimum spacing for feasible inputs and rejects impossible requests. It replaces the beta timing distribution with uniform order statistics on the slack interval; it is a corrected timing option, not an equivalent stochastic reproduction of the archived generator. Sixty seconds is an illustrative software contract and carries no claim of clinically realistic visit spacing.

Sampling weights are retrieved by their dataframe column label rather than by an attribute on a renamed tuple. Missing, nonfinite, or nonpositive weights cause an explicit failure. A strict identifier parser accepts the cohort's date serialization and rejects malformed or impossible dates. Setting reconciliation returns one value that callers must reuse for both context construction and exported status, with an outpatient-only facility rule taking precedence. Application-level integration must verify that callers actually reuse this returned value.

Python None values and floating-point NaN values are omitted before prompt serialization; unsupported scalar types fail strict JSON serialization. The module does not replace missing values with invented clinical facts. After generation, a single enclosing Markdown fence can be removed. Remaining fences, standalone nan tokens or the angle-bracketed NA marker, and texts outside the declared length interval are rejected. Overlength notes are rejected rather than truncated, because truncation can remove required identifiers or invalidate clinical context. Text transformations precede offset construction, and a SHA-256 digest binds the label file to the final UTF-8 text. Literal missing-token filters can also reject legitimate words or names and require local review; they are not semantic missingness detectors.

The span compiler uses a ledger of entity identifiers, label classes, and explicitly declared surface forms or aliases. It recovers repeated exact mentions with nonword boundary checks, reports start and end offsets in Unicode code points, and uses an exclusive end offset. Ambiguous ownership and overlapping spans cause rejection for adjudication. A required city is credited only when its own declared surface is recovered; a broader administrative-area expression cannot satisfy it through a shared address family. Every exported span must reproduce the exact labeled substring when applied to the final text.

This procedure produces verifiable candidate labels, not a complete gold corpus. A language model may introduce an identifier that is absent from the ledger or alter one beyond its declared aliases. Those expressions require an independent detection and annotation pass. Clinical hard negatives also need

reviewed labels. Automated exact matching is therefore one layer in annotation construction, followed by independent review before training claims are made.

3.9 Software and computational environment

The implementation uses Python notebooks. pandas and NumPy support tabular construction, sampling, joins, and timestamp handling; Python standard-library modules provide hashing, serialization, and file operations. The narrative wrapper uses PyTorch CUDA execution and Unsloth FastLanguageModel to load the four-bit MedGemma-27B text instruction artifact. Model and sampling configuration are described in Section 3.6. The reference verification module is separate from the original notebook pipeline.

Generation used a virtual-machine environment utilizing dual-GPU cloud accelerators, as reported by the author. NVIDIA documents its L-series architecture as accelerators supporting AI inference and virtualization. [19] The notebook wrapper binds a model instance to one resolved CUDA device; the hardware description alone does not establish a distributed-inference strategy or a measured two-GPU speedup. The archived material does not provide GPU utilization, elapsed generation time, or attributable cost, and these are not inferred from the accelerator specification.

This implementation does not make a particular GPU, cloud provider, country-specific resource pack, or model checkpoint part of the abstract data contract. Replacing a model or execution platform requires checking context length, tokenizer behavior, supported precision, licensing, and the existing export and validation contracts. Reproducibility also requires exact package and artifact versions, rather than hardware names alone.

4 Evaluation design

4.1 Evaluation material

We reviewed archived results of a descriptive census performed on 19 October 2026 on structured exports and two distinct narrative sets from one configured implementation. These are not multi-setting or multilingual validation data. The primary narrative set was a 200-document CSV, and a separate spreadsheet contained 100 documents for 50 patients. Their patient and visit sets were disjoint. Duplicate CSV exports and spreadsheet subsets were identified by patient, visit, and document keys and excluded from pooled counts. The archived inventory records filenames, byte sizes, and SHA-256 checksums. The original raw export directory was not available for a second census during this revision; saved aggregate results and analysis code are retained by the author and may be requested as described in Data and code availability. These measurements are an archived export census, not a newly repeated generation or data census.

The reference implementation and output sets were treated as separate evidence sources because a source-to-output execution manifest was unavailable. The evaluation measures properties of existing

data rather than reconstructing original generation attempts. No new language-model inference or clinical review was performed.

4.2 Structural temporal and text endpoints

Structural checks measured key uniqueness, patient-set agreement, per-patient encounter counts, and patient–visit mappings. Operational coverage used the full encounter relation as denominator. Temporal checks compared encounters with registration dates recovered from patient keys, demographic birthdates, and exported observation endpoints. Operational event times were compared with service times.

Timestamps were parsed using mixed-format parsing and month-first interpretation for slash-formatted dates. Adjacent within-patient differences were measured in source order, retaining the final time across processing chunks. Sorting was not used to remove ordering violations before measurement.

Text endpoints included character length, exact patient- and visit-key occurrence, and full demographic-name occurrence after case folding and whitespace normalization. Literal artifact screens detected markdown code fences, standalone nan tokens, a generation-instruction phrase, and a fixed generic filler phrase. These endpoints describe representation and export behavior; they are not clinical-quality scores or independent entity annotations.

4.3 Boundary verification

Eight controlled fixtures exercised isolated function or class definitions using fictional inputs. A further fixture examined operational-setting consistency. Tests included changed interpreter hash seeds, fixed beta offsets, overlapping names, category-specific payloads, a constructed validation object, and dataframe field access. The original fixture report records Python 3.12.14, NumPy 2.3.5, and pandas 3.0.1. The portable fixture definitions were rerun on 25 October 2026, with the new environment and outputs retained separately. Appendix B records their scope and findings. Fixtures were kept separate from frequencies measured in the output evaluation.

4.4 Verification of the revised module

The reference implementation was exercised with 39 executable tests using fictional strings and constructed records. Tests covered cross-process seed consistency under two interpreter hash seeds, patient-partition consistency after row reordering, 1,000 feasible timing configurations, exact-capacity and impossible timing cases, dataframe weight semantics, identifier parsing, setting precedence, missing-field serialization, wrapper removal, length rejection, repeated and aliased spans, Unicode offsets, wrong-city noncredit, and ambiguous-label rejection. The test runner records interpreter version, source hashes, test count, and failures in a machine-readable result file. These are deterministic examples and bounded generated cases, not a formal proof or an estimate of clinical error rates.

We used no inference calls for this verification. The unit of analysis is a software behavior, and all pass counts are restricted to the tested module. The fixture inputs were chosen to expose known defects and

representative boundary conditions rather than sampled from a clinical population. Rerunning the tests after integration with actual generation, exports, and annotations is required before interpreting them as full-pipeline validation.

4.5 Statistical interpretation

The archived export analysis is a census of the supplied artifacts. Counts and proportions describe those artifacts, so we do not attach sampling confidence intervals or hypothesis-test p values to their deterministic integrity checks. The two narrative sets differ in documents per visit and remain separate for text endpoints. The 300 documents do not form 300 independent patients: the pooled count is 250 patients and visits. Future model experiments should calculate uncertainty using the patient as the resampling unit and preserve all notes from a patient in the same partition.

5 Results

5.1 Structured scale and linkage

The cohort and demographic tables each contained 50,000 unique patient keys, with exact patient-set agreement. Sixteen encounter partitions contained 3,225,625 rows and distinct visit keys. All patients occurred in the cohort, every cohort patient had enccounters, and observed encounter counts equaled the sampled count for every patient. The realized mean was 69.30275 encounters per patient, compared with the analytical expectation of 69.65.

Table 2
Observed outputs and structural results.

Layer	Observed scale	Structural result
Patient anchors	50,000 patients	Unique keys
Demographics	50,000 patient records	Exact agreement with cohort patient set
Encounters	3,225,625 visits	Unique visits and correct cohort membership
Operational events	375,000 rows for 250,000 visits	Unique event keys and correct patient–visit mappings
Narrative set A	200 documents for 200 patients and visits	All pairs agree with encounter relation
Narrative set B	100 documents for 50 patients and visits	All pairs agree with encounter relation

Operational rows covered 7.75% of encounters and 3,412 patients. There were 125,000 visits with one event and 125,000 with two. All operational visits matched the encounter relation with the same patient. The rows comprised 372,750 outpatient and 2,250 inpatient events; no inpatient-labelled row lacked a

room or inpatient-service identifier. These row counts use a different denominator from visit-level setting proportions.

The narrative sets together contained 300 unique document keys for 250 patients and visits. Set B contained two documents per visit. These are observed text artifacts, not a retrospectively inferred count of validator-accepted documents.

5.2 Temporal behavior

All encounter timestamps parsed. None preceded identifier-derived registration or demographic birthdate, or exceeded the exported observation endpoint. No non-increasing adjacent encounter pairs occurred in source order. However, 341 adjacent intervals were positive and shorter than 60 seconds, with a minimum of approximately 0.057761 seconds. The archived outputs therefore satisfied strict ordering while demonstrating that the source repair rule does not impose a one-minute interval. The count represents 341 of 2,732,110 within-patient adjacent pairs, approximately 0.0125%, and is a contract finding rather than evidence of a clinical event-frequency distribution.

All operational event and service timestamps parsed, and no event preceded its service time. These findings concern the specified temporal assertions, rather than clinical progression across encounters.

5.3 Narrative representation

Set A contained 200 distinct exact texts, with minimum, median, and maximum lengths of 1,439, 3,276, and 6,919 characters. Every patient key appeared in its corresponding narrative; visit keys appeared in 188 texts and full demographic names in 181. Exact-name absence can reflect an omitted or abbreviated name and was not classified as an identity error.

Table 3
Text endpoints for the two narrative sets.

Screen	Set A 200 documents	Set B 100 documents
Correct patient key occurs	200	100
Correct visit key occurs	188	88
Full demographic name occurs	181	100
Markdown code fence occurs	175	62
Standalone nan token occurs	30	0
Generation-instruction phrase occurs	0	0
Fixed filler phrase occurs	0	0
Median character count	3,276	6,096.5

In set A, 179 documents matched at least one of the four artifact screens. The count is a literal screen-positive total. For the declared plain-text training export, enclosing fences are a representation defect; this screen alone does not measure medical plausibility or all export errors. Twelve texts were below the reference configuration's 2,200-character lower gate. Set B ranged from 3,375 to 12,961 characters, extending above the described reference configuration's upper gate. The number exceeding 9,000 characters was not retained as a separate aggregate endpoint and is not inferred from the maximum. Without original status reports, neither observation establishes what a particular generator run accepted.

The compared duplicate CSV contained the same 200 ordered document keys and exact narratives as set A. Additional sample files and spreadsheet subsets were not counted as separate generations. A small ancillary folder contained copied texts, instruction leakage, generic filler, and one visit key absent from the encounter relation. These examples motivated export checks but were not pooled with the two defined narrative sets.

5.4 Implementation boundaries

The fixtures identified differences between configured intent and effective behavior, including interpreter-sensitive seeding, overlapping candidate counts, broad geographic matching, targeted-only accounting, small-sample style allocation, dataframe weight access, date parsing, and the separately computed operational-setting flag. Full details are retained in Appendix B to support implementation and reproduction.

Two findings clarify interpretation of the main results. First, a broad administrative-area match could credit narrower city and postal targets through a shared address family, so ledger counts cannot establish category-specific realization. Second, the exported sub-minute gaps corroborated the spacing fixture, while the triggering operational-setting inconsistency was not observed among exported inpatient rows. A reproducible boundary condition and its frequency in a particular dataset are different results.

5.5 Verification of the revised module

All 39 reference-module tests passed, with zero recorded failures or errors. All 1,000 constructed feasible timing cases returned the requested number of anchors within their bounds and with at least 60 seconds between adjacent anchors. An exactly full timing window produced its expected boundary anchors, and an infeasible request was rejected. SHA-256-derived seeds agreed across the two tested interpreter hash configurations, while partition assignment remained stable under reversed patient order.

The weight test recovered the stored value 1.35 using column-label access, and missing or invalid weights failed explicitly. The cohort-format identifier was parsed successfully, malformed dates were rejected, and the outpatient-only facility rule returned outpatient status even for an explicit inpatient input. Text tests removed an enclosing fence before computing labels, rejected internal fences and missing-value tokens, and rejected notes outside the length interval without truncation. The exact-span tests recovered repeated mentions and declared aliases, preserved Unicode substring agreement,

rejected ambiguous spans, and did not credit a requested city when only a different geographic name appeared in the text.

These results validate the listed reference functions under the supplied test cases. They do not report a clean regenerated corpus, repaired prevalence estimates for the archived notes, or a trained de-identification model. Distinct source checksums distinguish the original generation snapshot from the revised module. The associated verification materials may be requested as described in Data and code availability.

6 Discussion

6.1 Contribution and interpretation

The framework connects patient-visit relationships, operational scope, configurable identity resources, requested narrative entities, and acceptance rules. The measured output scale demonstrates substantial structured construction, while pairwise checks establish specific cross-file properties. The operational subset illustrates why a valid relation can still cover only part of its upstream data. Reporting both quantities clarifies what the generated dataset can support.

The distinction between a payload, a validator result, and a verified span is equally consequential. A payload describes generation intent. A validator tests implemented matching rules. Independent annotation describes realized text. The broad-family matching fixture shows how treating these as one quantity can inflate apparent specificity. The framework therefore supports independent evaluation rather than using its own acceptance score as the only measure of success.

The planning-capacity bound adds a practical design check before expensive inference. With the configured thresholds, fewer than 57 accepted documents cannot fill the ledger even under ideal allocation. The actual requirement can be higher, particularly under the sequential style schedule and repeated values. This makes feasibility a property to establish from targets and capacity, rather than infer from whether a small test run completes.

Relative to prior work, the proposed contribution is the coupling of explicit record contracts, configurable entity semantics, deficit-based coverage planning, and layered evaluation. Clinical journey generation, tabular statistical fidelity, and identifier-focused testing address different endpoints. A controlled comparison must hold the intended task and resource budget constant. The choice of an L4 accelerator or a particular language model is an implementation decision, not a methodological novelty claim.

6.2 Applications and configuration portability

The framework can support synthetic clinical NLP development, annotation exercises, and information-system testing by providing linked structured context and unstructured narratives. The evaluated identifier-focused configuration targets de-identification development and stress testing. The reference module can create candidate span files tied to final text, while the relational layers supply patient-

grouping keys. Training use remains contingent on label completeness, realistic negative examples, and independent evaluation. A virtual EHR requires additional integration, described in Section 6.6.

Portability is a configuration and validation problem. Each deployment needs reviewed field definitions, resource provenance, identifier and contact conventions, document types, and the relevant data-handling policy. Administrative hierarchies and social attributes should be included only when required by the intended task. Languages, code switching, specialty vocabulary, and institutional shorthand need separate evaluation strata. No evaluated configuration can be assumed to represent all health systems merely because its generation code is reusable.

Privacy-preserving research describes the intended use of fictional records to reduce routine reliance on identifiable patient files. It is not a claim that every generated string is anonymous, that a pretrained model cannot reproduce sensitive material, or that a legal de-identification standard has been met. Published de-identification guidance illustrates the importance of explicit identifier policies and risk assessment. [20] Applicable requirements depend on the actual jurisdiction and use case. The present seed mechanism supplies computational reproducibility, not a formal privacy guarantee.

Relational continuity also differs from clinical continuity. Multiple encounters under one patient key do not establish diagnosis persistence, treatment response, or coherent disease progression. The archived reference sampler selects one visit per patient, and Set B supplies two documents for each selected visit. These observations do not establish longitudinal clinical trajectories. A trajectory model would require additional clinical state and consistency checks.

6.3 Evaluating clinical fidelity and downstream utility

High fidelity must be defined for the intended task. Structural fidelity concerns valid records and relationships; semantic fidelity concerns agreement between a note and its supplied context; clinical fidelity concerns plausible facts and trajectories; distributional fidelity requires an appropriate reference population. Downstream utility is a separate empirical outcome. For the identifier-focused use case, the next question is whether an annotated synthetic corpus improves identification of protected expressions on independent clinical text. Endpoints should include strict entity-level recall, precision, F1, category-level errors, and the proportion of documents with a missed direct identifier. Over-redaction should be assessed on reviewed negative expressions such as clinical measurements, medications, and namesake medical terms. Synthetic-only testing cannot establish transfer to real documentation.

A useful minimum design compares three training conditions using the same downstream tagger and training budget: an available licensed real-data baseline, synthetic training alone, and mixed training data. When real training data are unavailable, the comparison can start with an externally trained tagger and a synthetic adaptation condition; that design must disclose its different question. A held-out local test set requires lawful access and independently reviewed spans. Public corpora can provide an external reference, but a public English-language benchmark cannot substitute for local validation.

The coverage-planning ablation must isolate feedback. One arm samples targets from a fixed allocation and another uses deficits, with the same contexts, target capacity, style schedule, payload generator, inference settings, retry limit, and output-token budget. A third unconstrained prompting arm can assess explicit targeting, but it answers a separate question. At least three seed-controlled runs would expose run variation; the ultimate number of runs and annotated examples should follow a pilot analysis of rare categories and reviewer workload. No arm or seed count is presented here as an already completed experiment.

The annotation sample should include random accepted documents, rare identifiers, rejected candidates, and cases near decision thresholds. Two annotators should mark spans independently and resolve disagreements with a prespecified policy. Clinical plausibility should receive a separate blinded review of a pilot sample, for example 30 notes assessed by two clinicians for context agreement, internal clinical coherence, and documentation authenticity. Thirty notes is a feasibility proposal rather than a powered estimate of corpus-wide clinical validity. Clinical review and de-identification utility answer different questions and are both needed for claims spanning those domains.

Table 4 specifies the experiments and their interpretation. Patient-level partitions should be frozen before generating or annotating notes. Uncertainty for downstream comparisons should be estimated with patient-cluster bootstrap intervals, and repeated model training should report variation across seeds. Model selection must use development data; testing should occur once the policy and checkpoints are frozen.

Table 4
Evaluation plan for claims beyond software integrity.

Comparison	Primary endpoint	Interpretation
Fixed allocation versus deficit planning	Verified category coverage per fixed generation budget	Incremental effect of feedback
Synthetic versus licensed real or mixed training	Strict entity recall and F1 on an independent test corpus	Utility and transfer under one label policy
Identifier-positive versus hard-negative expressions	Missed identifiers and over-redaction	Privacy-utility balance
Blinded clinical and local review	Anchored coherence and documentation ratings	Plausibility for the reviewed tasks and sample
Country or language adaptation	Per-stratum errors and coverage	Transfer to explicitly evaluated settings

6.4 Reproducible generation and resource reporting

A future end-to-end run must link its executable source and configuration to every exported artifact. The manifest should record code and prompt hashes, reference-resource versions, model and tokenizer revision identifiers, software versions, hardware, random states, selected patient-visit keys, shard configuration, and output hashes. Each attempt needs its document identity, attempt number, input and

output token counts, elapsed time, termination reason, validation outcomes, and accepted or rejected status. Failed candidates belong in a diagnostic partition and must not enter a training export through a fallback path.

The author-reported two-L4-GPU environment establishes the stated deployment hardware, not a throughput benchmark. The 300 archived narratives cannot be attributed to GPU capacity, cost, or a planned sampling design without an execution record. A new sample size should follow category coverage and annotation capacity, with total computation and failed attempts disclosed. Millions of structured encounters provide available contexts, not millions of independently validated unstructured training examples.

Resource efficiency should be reported as total generation tokens, attempts, elapsed time, and attributable cost per independently accepted annotated document. Price, electricity, or hardware amortization assumptions must be stated. Model size and four-bit quantization alone establish neither affordability nor superiority. The archive lacks the attempt histories needed to calculate these endpoints retrospectively.

6.5 Implications of the reference corrections

The new functions turn several identified failure conditions into explicit and testable decisions. They also change parts of the method: the bounded timing sampler uses a different distribution, span matching is conservative, and ambiguous labels require review. Such changes can lower apparent coverage or increase rejection while improving interpretability. The effect on note quality, throughput, and training utility remains an empirical question.

Software tests are necessary but narrower than complete validation. Passing a missing-token fixture does not show that all generated documents are clean; passing an exact-span fixture does not show that unplanned identifiers have been labeled. The study therefore reports archived output findings, legacy fixture behavior, and revised function tests without combining them into a single overall pass rate. This separation lets other researchers reproduce the available claims and identify the remaining integration work.

6.6 Extension toward a virtual EHR

A virtual electronic health record is a research environment that stores, retrieves, and presents fictional patient histories together with their linked encounters and documents. The current pipeline supplies a relational data foundation for such an environment. It is not itself a production EHR: the study did not implement a persistent clinical service, user interface, access-control layer, or a validated interchange adapter. Its immediate role would be to populate a research sandbox for workflow, annotation, retrieval, and software tests.

An integration layer could map patient and demographic records to Patient resources, encounters to Encounter resources, and clinical-note metadata to DocumentReference resources while retaining links

to note content. These are proposed mappings to the published HL7 FHIR Release 4 resource model, not a claim of conformance. [21] Table 5 identifies the main integration work. Schema validation, terminology bindings, reference resolution, and round-trip preservation would need to be tested against a declared implementation profile.

Table 5
Proposed integration of generated artifacts with a virtual EHR.

Generated component	Proposed virtual EHR representation	Additional work required
Patient anchors and demographics	Patient identity and demographic record	Authoritative fields, identifier namespaces, profile validation
Encounter records	Encounter history linked to the patient	Reference integrity, encounter semantics, lifecycle rules
Facility and operational context	Organization and location references	Controlled mappings; preserve the meaning of event and setting fields
Clinical narratives and metadata	DocumentReference with linked note content	Content storage, encounter linkage, versioning, synthetic labels
Run records and validation results	Linked provenance and research audit records	Persistent service, access controls, and integration tests

Synthetic-EHR integration has established precedents. SyntHIR combines synthetic-data generation, data conversion, and an adapter to a FHIR-based EHR environment. [22] Its contribution concerns interoperability and tool transportability. The proposed extension here would connect a contract-checked record-to-narrative pipeline to that class of environment, with explicit generation status and coverage records. It is not a claim to be the first synthetic EHR, and no comparable deployment result is reported.

A clinically coherent virtual history would also need an explicit state model for diagnoses, medications, observations, and care transitions. Reusing a patient identifier across many encounters does not create that state model. Generated text must be reconciled with structured clinical facts, and later notes must respect earlier events where longitudinal coherence is required. Fictional records should remain clearly labeled and segregated from systems used for real clinical care.

6.7 Continual generation and quality control

The framework can be extended from isolated batches to repeated, on-demand corpus production. A proposed service cycle would allocate a versioned run, construct or select patient contexts, request categories from a persisted coverage ledger, execute bounded generation, validate final text and links, and publish accepted artifacts to a versioned store. Failed and review-pending candidates would remain in separate diagnostic partitions. New batches could then target remaining deficits or new scenarios without silently modifying an earlier research release.

Continual operation requires persistent identifiers, restart checkpoints, idempotent writes, and reconciliation of coverage across workers. Distinct-value coverage must use set unions, not sums of worker totals. Patient-level partition assignments should remain fixed across releases, and duplicate or near-duplicate narratives require checks across the entire release history. Appending encounters to an existing patient additionally requires a validated clinical-state model. These service-level controls are proposed requirements; their implementation and sustained operation were not evaluated here.

A renewable source of narratives is not an assurance of unlimited clinically faithful data. Repeated use of a fixed model and prompt library can produce correlated text and exhaust scenario diversity. Quality control should measure semantic agreement with source context, clinician-rated coherence, category and subgroup coverage, duplication, privacy risk, and task utility on independent data. A sustained-generation study should also report throughput, restart recovery, failure rates, resource use per independently accepted document, and quality across successive batches. The current two-GPU description and archived narrative counts do not establish those outcomes.

7 Limitations

The archived export census was performed on supplied files without an execution manifest linking them to a frozen source configuration. The current revision preserves its analysis code, inventory, and aggregate findings but did not repeat the raw-file census. Exact model revisions, sampling provenance, generation acceptance rates, token use, costs, and coverage trajectories cannot be recovered from those aggregates. Narrative Sets A and B differ in their structure and are not experimental arms.

The source-level fixtures and revised-module tests establish bounded software behaviors. They do not validate a corrected full pipeline or retrospectively repair archived outputs. The complete generator still needs integration with the reference functions, a clean run, independent export validation, and versioned resources. Some decisions, including style rounding and the intentional exclusion of incidental categories from a planning ledger, are design choices whose task impact requires measurement rather than automatic designation as defects.

No de-identification model was trained for this study. There was no independent gold-span corpus, blinded clinical assessment, adaptive-planning ablation, or synthetic-to-real transfer experiment. The archived text screens do not measure clinical accuracy, complete PHI coverage, or generalization. Span matching in the revised module misses undeclared variants and unplanned identifiers, while conservative overlap rejection can increase annotation workload.

The evaluated resource configuration encodes scenario assumptions and has not been calibrated to a population distribution. No multi-setting, multilingual, subgroup-fairness, or model-memorization evaluation was conducted. Independently generated contact fields may disagree, generated strings can coincide with real values, and pretrained model behavior adds uncertainty. The study did not deploy a virtual EHR, validate a FHIR adapter, or measure sustained generation. Claims remain restricted to the described method, archived integrity findings, and tested software contracts.

8 Conclusion

We describe a configurable framework for linked synthetic clinical records and unstructured narratives for privacy-preserving research. Its methodological contribution couples record contracts, deficit-driven coverage planning, and separate checks of generated representations. An archived census of 50,000 patients, 3,225,625 encounters, 375,000 operational events, and 300 narrative documents establishes specific linkage properties and exposes temporal and export limitations. A reference module passed 39 software tests. The Python and four-bit language-model implementation was reported to run in a virtual-machine environment utilizing dual-GPU cloud accelerators. The architecture provides a data foundation for a proposed virtual EHR and continual, on-demand narrative generation. Those extensions require implementation and validation; high clinical fidelity, sustained operation, privacy protection, and transfer across settings are not established by the current integrity results.

Declarations

Ethics approval and consent to participate

This computational study used fictional records, source-code fixtures, and archived analyses of synthetic exports. It did not recruit human participants or analyze individual real patient records. No institutional approval or exemption identifier is claimed. Any subsequent evaluation involving real clinical text will require the applicable institutional review and data-access arrangements before that evaluation begins.

Consent for publication

No identifiable human participant material is presented. Consent for publication of participant data is not applicable to the material reported here.

Funding

The author received no external funding for this work.

Competing interests

The author declares no competing interests.

Data and code availability

A public GitHub repository is planned for the project code and documentation; no public repository URL has been assigned for this version. In the meantime, the portable legacy fixtures and their provenance,

archived export-analysis code, aggregate results, source checksums, revised contract module, executable tests, and software-verification records may be requested from the corresponding author at allanzablon@gmail.com . Materials will be shared on reasonable request, subject to applicable permissions and third-party licences.

The legacy fixtures can be run without the original generation environment. Repeating the archived export census requires the original source files and directory layout; those raw files are not included with this article. No complete narrative corpus, pretrained model weights, or newly generated dataset is released with this version. Access to or redistribution of third-party resources and model artifacts remains subject to applicable permissions and licences.

References

1. Walonoski J, Kramer M, Nichols J, Quina A, Moesel C, Hall D, et al. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association*. 2018;25(3):230–238. doi:10.1093/jamia/ocx079. Publication record.
2. Poulett W, Waterhouse A, Wallace B, Kynoch S, Imaz Blanco A, Spence M, Pearson J. A Pipeline for Generating Longitudinal Synthetic Clinical Notes Using Large Language Models. *arXiv* [preprint]. 2026. arXiv:2606.26879v2. doi:10.48550/arXiv.2606.26879. Public manuscript.
3. Yan C, Yan Y, Wan Z, Zhang Z, Omberg L, Guinney J, et al. A Multifaceted benchmarking of synthetic electronic health record generation models. *Nature Communications*. 2022;13:7609. doi:10.1038/s41467-022-35295-1. Publication.
4. Kumichev G, Blinov P, Kuzkina Y, Goncharov V, Zubkova G, Zenovkin N, et al. MedSyn: LLM-Based Synthetic Medical Text Generation Framework. In: *Machine Learning and Knowledge Discovery in Databases Applied Data Science Track*. 2024:215–230. doi:10.1007/978-3-031-70381-2_14. Public manuscript.
5. Stubbs A, Uzuner Ö. Annotating longitudinal clinical narratives for de-identification: The 2014 i2b2/UTHealth corpus. *Journal of Biomedical Informatics*. 2015;58 Suppl:S20–S29. doi:10.1016/j.jbi.2015.07.020. Publication record.
6. Sarkar AR, Chuang YS, Mohammed N, Jiang X. De-identification is not enough: a comparison between de-identified and synthetic clinical notes. *Scientific Reports*. 2024;14:29669. doi:10.1038/s41598-024-81170-y. Public author manuscript and journal metadata.
7. Dwork C, Roth A. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*. 2014;9(3–4):211–407. doi:10.1561/0400000042. Public monograph.
8. Li J, Zhou Y, Jiang X, Natarajan K, Pakhomov SVS, Liu H, Xu H. Are synthetic clinical notes useful for real natural language processing tasks: A case study on clinical entity recognition. *Journal of the American Medical Informatics Association*. 2021;28(10):2193-2201. doi:10.1093/jamia/ocab112. https://doi.org/10.1093/jamia/ocab112.

9. Dernoncourt F, Lee JY, Uzuner O, Szolovits P. De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association*. 2017;24(3):596-606. doi:10.1093/jamia/ocw156. https://doi.org/10.1093/jamia/ocw156.
10. Weatherhead J, Golovko G, McCaffrey P. ASQ-PHI: An adversarial synthetic data benchmark for clinical de-identification and search utility. *Data in Brief*. 2026;65:112586. doi:10.1016/j.dib.2026.112586. https://doi.org/10.1016/j.dib.2026.112586.
11. Shimizu S, Baroud I, Raithel L, Yada S, Wakamiya S, Aramaki E. RecordTwin: Towards Creating Safe Synthetic Clinical Corpora. In: *Findings of the Association for Computational Linguistics ACL 2025*. 2025:14714-14726. doi:10.18653/v1/2025.findings-acl.759. https://aclanthology.org/2025.findings-acl.759/.
12. Cornelius J, Rinaldi F. Measuring the gap: correlating synthetic-to-real drift with PHI de-identification performance. *Genomics & Informatics*. 2026;24:10. doi:10.1186/s44342-026-00072-9. https://doi.org/10.1186/s44342-026-00072-9.
13. Sellergren A, Kazemzadeh S, Jaroensri T, Kiraly A, Traverse M, Kohlberger T, et al. MedGemma Technical Report. *arXiv* [preprint]. 2025; revised 2026. arXiv:2507.05201v4. doi:10.48550/arXiv.2507.05201. Public manuscript.
14. Google. MedGemma-27B text instruction model [model card]. Hugging Face. Accessed October 19, 2026. Model documentation.
15. Unsloth. MedGemma-27B text instruction model, bitsandbytes four-bit quantization [model artifact]. Hugging Face. Accessed October 19, 2026. Model artifact.
16. Python Software Foundation. Data model: object.__hash__ [software documentation]. Accessed October 19, 2026. Hash behavior documentation.
17. Mitchell M, Wu S, Zaldivar A, Barnes P, Vasserman L, Hutchinson B, et al. Model Cards for Model Reporting. In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 2019:220-229. doi:10.1145/3287560.3287596. Public manuscript and publication metadata.
18. Gebru T, Morgenstern J, Vecchione B, Vaughan JW, Wallach H, Daumé H III, Crawford K. Datasheets for Datasets. *Communications of the ACM*. 2021;64(12):86–92. doi:10.1145/3458723. Author publication record and public manuscript.
19. NVIDIA. NVIDIA L-Series Tensor Core GPUs [product documentation].. Accessed October 26, 2026. https://www.nvidia.com/en-us/data-center/l4/.
20. US Department of Health and Human Services. Guidance Regarding Methods for De-identification of Protected Health Information in Accordance with the HIPAA Privacy Rule. 2012. Accessed October 25, 2026. https://www.hhs.gov/hipaa/for-professionals/special-topics/de-identification/.
21. HL7 International. FHIR Release 4 version 4.0.1. Overview and resource definitions [technical specification]. Accessed October 26, 2026. https://hl7.org/fhir/R4/overview.html.
22. Chauhan P, Askar MGS, Svendsen K, Fjukstad B, Elvevåg B, Bongo LA, Pedersen E. From research to clinic: Accelerating the translation of clinical decision support systems by making synthetic data

interoperable. arXiv [preprint]. 2023; revised 2025. arXiv:2308.02613v2. doi:10.48550/arXiv.2308.02613. https://arxiv.org/abs/2308.02613v2.

Figures

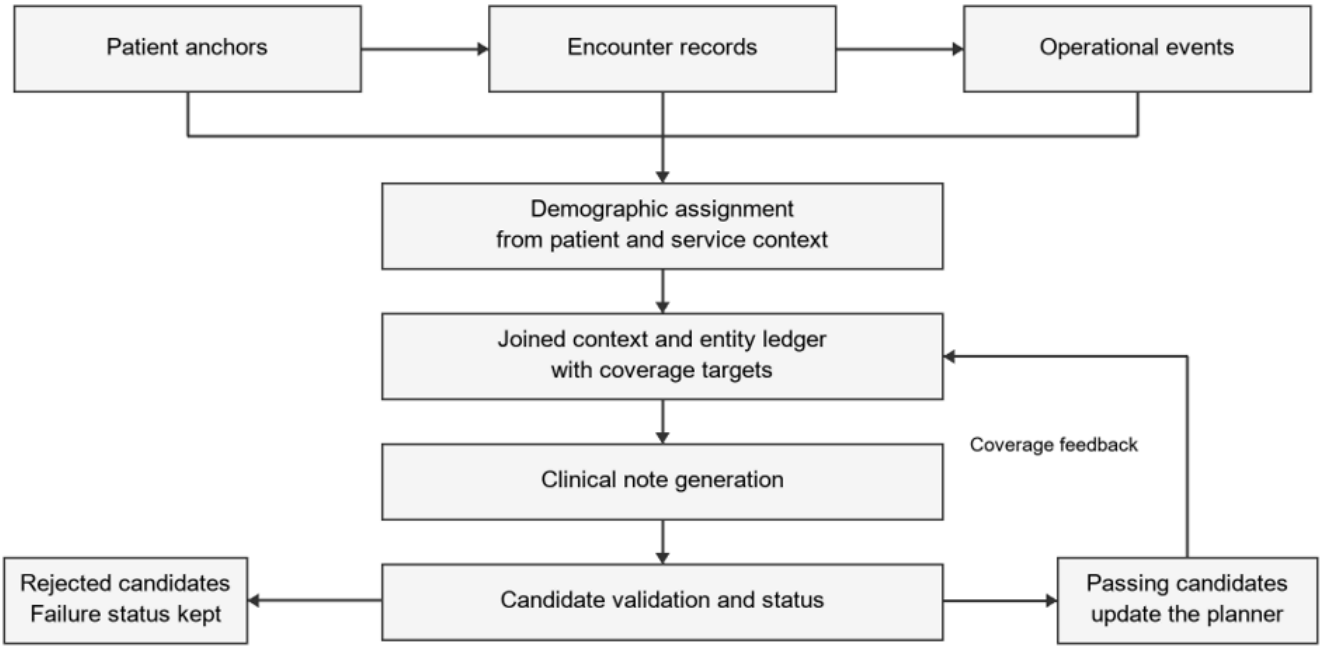


Figure 1

Construction dependencies and narrative feedback. Patient and encounter records supply operational and demographic context. The note stage combines that context with target categories and a payload. Accepted outcomes update the coverage state used for subsequent documents.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Appendix.docx