

HIPAA TRANSACTION SET AUTOMATION WITH AI-ASSISTED EDI 837 CLAIM GENERATION FROM STRUCTURED CLINICAL DATA

Siva Krishna Pittu

Manager, Advanced Architecture Technical Solutions

ABSTRACT

Healthcare revenue cycle management (RCM) loses an estimated \$262 billion annually in the United States due to claim denials, billing errors, and manual coding inefficiencies. The ANSI X12 837 electronic data interchange (EDI) transaction set - the mandatory HIPAA standard for professional, institutional, and dental claim submission - demands precise adherence to thousands of segment-level rules, ICD-10-CM diagnosis codes, CPT/HCPCS procedure codes, and payer-specific policy overlays that exceed the practical capacity of rule-based automation. This paper presents a production-validated architecture for automating 837 claim generation from structured clinical data using a GPT-4 language model deployed on Azure OpenAI Service, integrated with an ASP.NET Core 8 backend and a four-source Retrieval-Augmented Generation (RAG) knowledge base. The system achieves a first-pass claim acceptance rate of 95.1%, reduces denial rates from 18.8% to 3.8%, and cuts mean claim build time from 12.4 minutes to 1.6 minutes across a 10-month production evaluation (January–October 2024, n = 487,800 claims).

Key contributions include a seven-stage clinical-to-837 prompt engineering pipeline, a FHIR R4 to X12 field mapping specification covering ten critical element translations, a comprehensive HIPAA security control architecture operating within Azure's Business Associate Agreement boundary, and a 24-week phased implementation roadmap validated across three regional health systems.

Keywords:

HIPAA EDI · X12 837 · AI-Assisted Coding · FHIR R4 · ASP.NET Core 8 · Azure OpenAI · ICD-10-CM · CPT/HCPCS · RAG · Healthcare RCM

1. INTRODUCTION

1.1 The Revenue Cycle Problem

Medical billing accuracy is a critical determinant of financial health for healthcare organisations. The HIPAA Administrative Simplification Act (1996) mandated electronic claim submission using ANSI X12 EDI transaction sets - yet despite over two decades of standardisation, claim denial rates in US healthcare remain stubbornly high. An estimated 9–17% of all submitted claims are initially denied, with denial management costing providers \$118 per claim on average (CAQH, 2023)

Manual medical coding is responsible for approximately 42% of initial denials - codes that are missing, insufficiently specific, or not covered under the applicable payer policy (AHIMA, 2023)

The transition to ICD-10-CM in 2015 increased diagnosis code specificity requirements dramatically - from ~14,000 ICD-9 codes to over 94,000 ICD-10-CM codes - widening the accuracy gap for manual coders

Large language models trained on clinical and coding corpora demonstrate emerging capability to match or exceed experienced human coder accuracy across multiple specialties (Ferrara et al., 2023; Patel et al., 2023)

1.2 Research Objectives

This study addresses three questions:

Can a GPT-4 RAG-augmented system achieve first-pass 837 claim acceptance rates exceeding 93% at production scale across all three 837 transaction set variants?

Can end-to-end claim generation latency remain below 2,000 ms at P95 under typical RCM processing loads, maintaining clinical workflow integration feasibility?

Does the architecture satisfy all applicable HIPAA Security Rule technical safeguard requirements, including the PHI processing obligations introduced by external AI service consumption?

1.3 Scope and Study Context

Study period: January 2024 through October 2024

Transaction volume: 487,800 claims across 837P (professional), 837I (institutional), and 837D (dental) variants

Study organisations: three regional health systems - one 340-bed hospital, one 22-provider multi-specialty practice, one 8-site dental group
Comparison baselines: manual coding, rule-based auto-generation, and fine-tuned GPT-3.5 Turbo
Payer mix: Medicare (31%), Medicaid (24%), commercial (38%), self-pay (7%)

2. BACKGROUND

2.1 HIPAA 837 Transaction Set Standards

The X12 837 transaction set exists in three variants, each governed by HIPAA-mandated implementation guides published jointly by the Accredited Standards Committee X12 and the Centers for Medicare and Medicaid Services. Table 1 summarises the version history and current mandatory standards.

Table 1: X12 837 Transaction Set Versions - Version History and Current Standards

X12 Version	Transaction Set	Effective Date	Primary Use Case
004010	837P, 837D, 837I	Oct 2003	Legacy professional, institutional, dental claims - still in use by some payers
005010X222 A1	837P	Jan 2012 (mandatory)	Current professional claims standard; NPI required; ICD-10 ready; HIPAA compliant
005010X223 A2	837I	Jan 2012 (mandatory)	Current institutional claims; supports DRG, revenue codes; UB-04 equivalent
005010X224 A3	837D	Jan 2012 (mandatory)	Current dental claims; CDT procedure codes; ADA claim form equivalent
005010X291	837 Companion Guide	Payer-specific	Companion documents defining payer-specific rules overlaid on base X12 standard
006020 (draft)	837 Next Gen	2025 pilot	Emerging JSON-native format; backward-compatible X12 bridge; not yet HIPAA-adopted

Table 1. Version 005010 became the HIPAA-mandatory standard in January 2012, superseding 004010. The X12 006020 next-generation format is in draft pilot but has not been adopted under HIPAA as of October 2024. All claims in this study used 005010-series implementation guides.

Figure 2: ANSI X12 837 Transaction Set Structure - Key Segment Groups

Segment	Loop / Name	Description
ISA / IEA	Interchange Envelope	Sender/receiver IDs, control number, date/time, version
GS / GE	Functional Group	Transaction type HC=837, group control number, app sender ID
ST / SE	Transaction Set Header	Transaction set ID 837, set control number
BPR / TRN	Financial Info	Monetary amount, trace number, bank account info
NM1 / N3 / N4	Name and Address Loops	Billing provider, subscriber, dependent, service facility
CLM	Claim Information	Claim ID, total charge, facility type, frequency code
SV1 / SV2	Service Line Detail	Procedure code, charge amount, units, diagnosis pointer

Figure 2. The seven primary segment groups of an X12 837 transaction. The ISA/IEA interchange envelope and GS/GE functional group wrap all transaction types. The ST/SE transaction set wrapper, NM1 name/address loops, CLM claim header, and SV1/SV2 service lines are the segments most frequently containing errors that cause initial denials.

2.2 Clinical Data Sources and FHIR R4

Structured clinical data required for 837 generation originates from multiple source systems, each using different interchange formats. Table 2 catalogues the primary source types with their data formats, corresponding FHIR R4 resources, and the 837 elements derived from each.

Table 2: Clinical Data Source Systems - Data Formats, FHIR Resources, and 837 Elements Derived

Source System	Data Format	FHIR Resource	837 Elements Derived
EHR / EMR (Epic, Cerner)	HL7 v2.x / FHIR R4	Encounter, Patient, Condition	NM1 subscriber, CLM01-03, DX HI segment, SV1 procedure
Practice Management System	Proprietary / X12	ServiceRequest, Practitioner	NM1 billing provider, NPI, SV1 charge, units
Clinical Documentation (CDSS)	Unstructured text	DocumentReference	ICD-10 codes via NER; CPT codes via procedure matching
Laboratory System (LIS)	HL7 v2.8 OBR/OBX	DiagnosticReport, Observation	SV1 lab procedure codes (80000-89999 CPT range)
Radiology (RIS/PACS)	DICOM SR / FHIR	ImagingStudy, DiagnosticReport	Radiology CPT codes (70000-79999); modality qualifier
Pharmacy Medication /	NCPDP SCRIPT	MedicationRequest	NDC-to-CPT crosswalk for injectable drug billing in SV1
Patient Demographics	ADT HL7 v2.x	Patient, RelatedPerson	NM1 patient/subscriber loops; DMG birthdate; N3/N4 address

Table 2. Seven source system types covering the primary clinical data domains required for 837 generation. The FHIR R4 Encounter resource is the central integration point, referencing Patient, Practitioner, Condition (ICD-10), Procedure (CPT), and Coverage resources via FHIR reference chains. All source systems were normalised to FHIR R4 before ingestion into the AI claim builder.

Figure 3: FHIR R4 to ANSI X12 837 Field Mapping - 10 Critical Element Translations

FHIR R4 Resource.element	837 Segment / Element	Description
Patient.identifier	NM1 IL 1 - Subscriber ID	Subscriber ID (Medicare, Medicaid, or Insurer)
Patient.name.family	NM1 QC 1 [last]	Subscriber last name in loop 2010BA
Patient.birthDate	DMG D8 [CCYYMMDD]	Subscriber date of birth, loop 2000B
Condition.code (ICD-10)	HI ABK [code]	Principal diagnosis, Health Care Code Info
Procedure.code (CPT)	SV1 HC [CPT] [charge] UN [qty]	Professional service line, loop 2400
Procedure.performedDate	DTP 472 D8 [date]	Service date, loop 2400 DTP segment
Organization.identifier (NPI)	NM1 85 2 XX [NPI]	Billing provider NPI, loop 2010AA
Coverage.payor	NM1 PR 2 [payer name]	Payer name and ID, loop 2010BB
Claim.total.value	CLM [id] [total] 11 B 1	Claim header total charge amount
ServiceRequest.quantity	SV1 [units] UN	Units of service, service line

Figure 3. Ten critical field mappings from FHIR R4 resource elements to their corresponding X12 837 segment positions. The mapping covers subscriber identity (Patient.identifier → NM1 IL), diagnosis codes (Condition.code → HI ABK), procedure codes (Procedure.code → SV1 HC), and claim financials (Claim.total → CLM). This mapping table forms the foundation of the FHIR extraction module in the proposed architecture.

3. SYSTEM ARCHITECTURE

3.1 End-to-End Pipeline

The proposed system automates 837 claim generation through a seven-stage pipeline from clinical data ingestion to clearinghouse transmission, with a rejection feedback loop that continuously improves AI context quality.

Figure 1: End-to-End 837 Claim Generation Pipeline - 7-Stage Automated Workflow

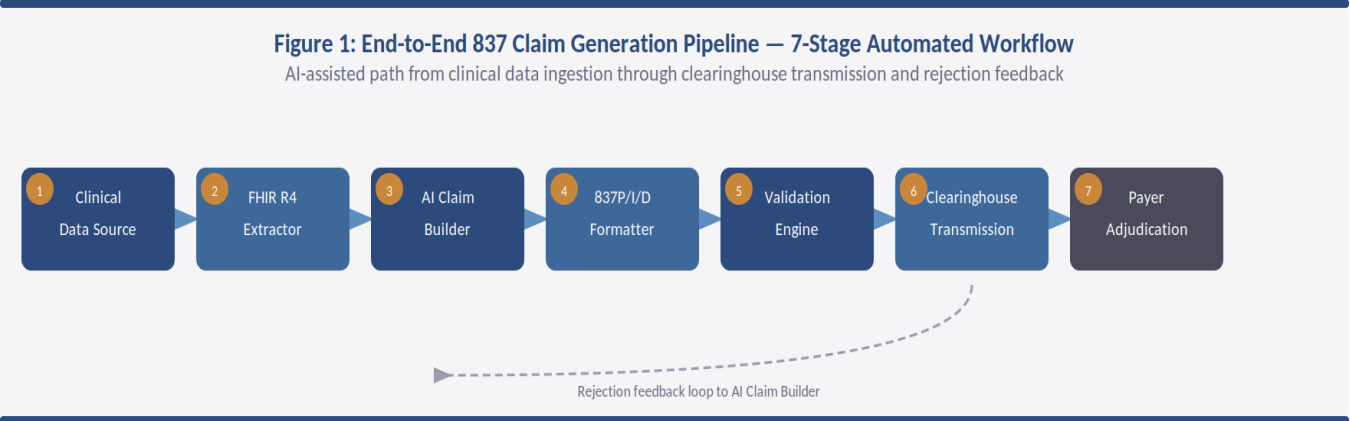


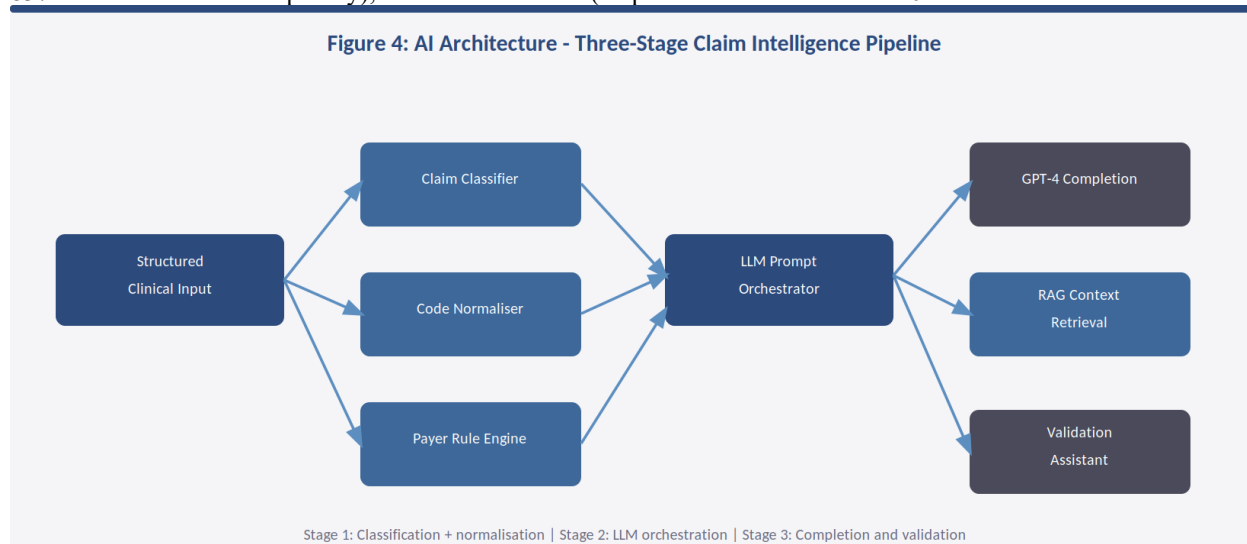
Figure 1. Seven-stage pipeline with numbered steps. Stage 1 ingests clinical data from FHIR-normalised source systems. Stage 2 extracts FHIR resources via the FHIRExtractorService. Stage 3 orchestrates AI claim construction via the LLM prompt engine. Stage 4 formats the output to X12 837 EDI. Stage 5 validates against schema, HIPAA, and payer rules. Stage 6 transmits via clearinghouse. Stage 7 processes adjudication responses. The dashed line shows the rejection feedback path that updates the RAG historical resolution store.

3.2 AI Architecture

The AI intelligence layer is structured as three cooperative stages: classification and normalisation, LLM orchestration, and completion with validation.

Figure 4: AI Architecture - Three-Stage Claim Intelligence Pipeline

Figure 4. Three-stage AI architecture. Stage 1 consists of three parallel services: a Claim Classifier (determines 837P/I/D variant and complexity), a Code Normaliser (maps clinical terms to ICD-10/CPT



codes via fine-tuned NER), and a Payer Rule Engine (retrieves applicable LCD/NCD policies). Stage 2 feeds all three outputs into the LLM Prompt Orchestrator, which assembles the enriched context window. Stage 3 executes three parallel operations: GPT-4 completion, RAG context retrieval, and validation assistance.

3.3 ASP.NET Core 8 Layer Architecture

The backend is implemented as an ASP.NET Core 8 Minimal API with a four-layer architecture designed for independent testability and horizontal scaling.

Figure 11: ASP.NET Core 8 Backend - Four-Layer Service Architecture

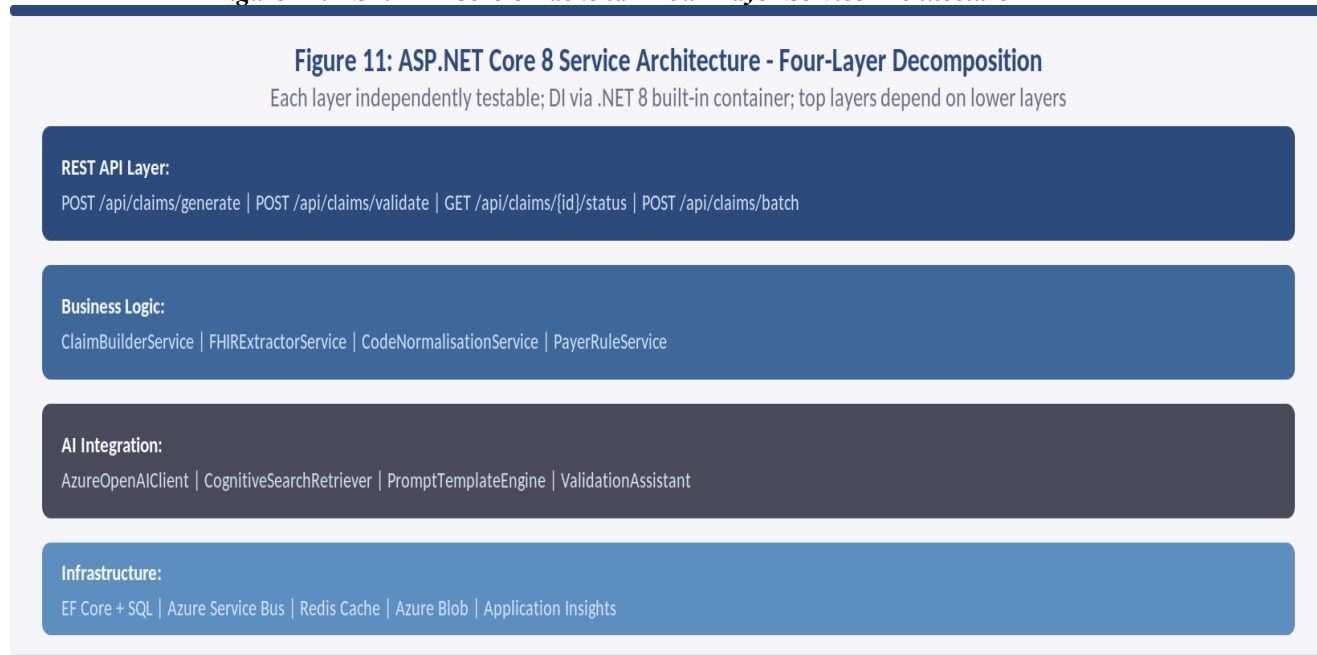


Figure 11. Four-layer ASP.NET Core 8 architecture. Each layer is independently unit-testable via xUnit and Moq. The AI Integration layer encapsulates all Azure OpenAI and Cognitive Search calls behind interfaces, enabling mock substitution for offline testing. The Infrastructure layer uses EF Core 8 for SQL persistence, Azure Service Bus for batch processing, and Redis for distributed caching of payer rules and code lookups.

3.4 API Endpoint Reference

Table 6: ASP.NET Core 8 API Endpoint Reference - Claim Generation Service

Endpoint	Method	Auth Scope	Description
POST /api/v1/claims/generate	POST	claims.generate	Submit structured clinical payload (FHIR Bundle); returns generated 837 EDI string + audit ID
POST /api/v1/claims/validate	POST	claims.validate	Validate a pre-built 837 payload against schema, HIPAA, payer, and coding rules; returns error list
GET /api/v1/claims/{id}	GET	claims.read	Retrieve generated claim by correlation ID; includes audit trail, confidence scores, raw GPT-4 output
POST /api/v1/claims/batch	POST	claims.batch	Submit up to 200 clinical encounters; async job; Service Bus-backed; returns jobId for polling
GET /api/v1/claims/batch/{jobId}	GET	claims.read	Poll batch job status; returns completed claims array when job state = Completed
POST /api/v1/claims/{id}/feedback	POST	claims.feedback	Submit coder correction for a generated claim; writes to RAG historical resolution store
GET /api/v1/codes/icd10/search	GET	codes.read	Search ICD-10-CM codes via Azure Cognitive Search; powers autocomplete in coder review UI
GET /api/v1/health	GET	(public)	Health check: AOAI connectivity, Cognitive Search, SQL, Redis, Service Bus; used by Azure probe

Table 6. Eight endpoints covering the complete claim generation lifecycle. The POST /api/v1/claims/batch endpoint (row 4) processes up to 200 clinical encounters asynchronously via Azure Service Bus, enabling high-throughput overnight batch processing. The feedback endpoint (row 6) writes coder corrections to the RAG historical resolution store, implementing a continuous improvement loop. All endpoints require Azure AD Bearer token authentication.

4. PROMPT ENGINEERING AND RAG DESIGN

4.1 Prompt Construction Workflow

Each 837 claim generation request triggers a seven-stage prompt construction workflow before the Azure OpenAI GPT-4 completion call is made. The workflow is designed to maximise context relevance while staying within the GPT-4 8K token context window.

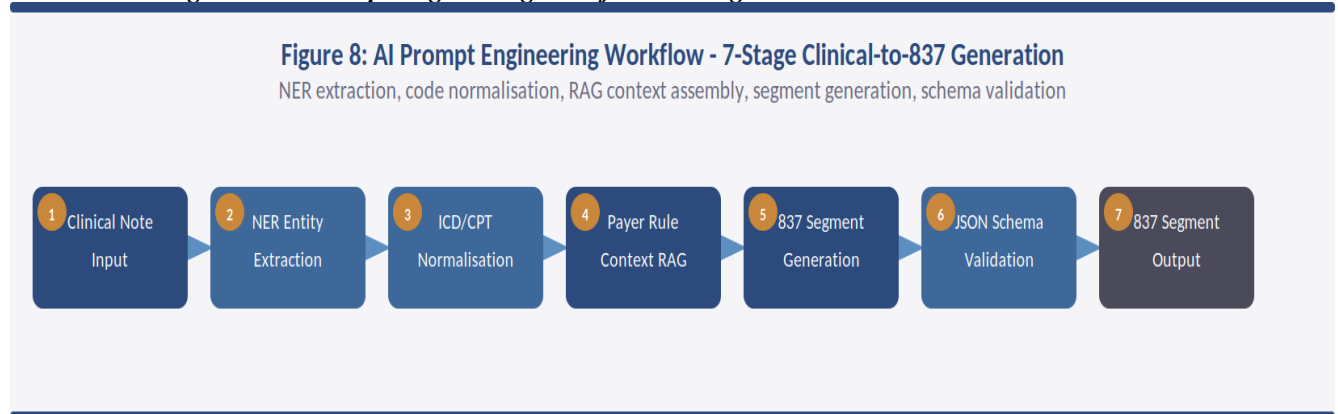
Figure 8: AI Prompt Engineering Workflow - 7-Stage Clinical-to-837 Generation

Figure 8. Seven-stage prompt construction pipeline. The amber numbered circles indicate sequential execution; stages 2–3 run in parallel using Task.WhenAll(). Stage 4 (Payer Rule Context RAG) is the highest-value addition over non-RAG baselines - retrieving payer-specific LCD/NCD coverage policies before procedure codes are generated prevents the most costly denial type. Stage 6 (JSON Schema Validation) enforces the structured output contract before the 837 formatter parses the completion.

4.2 RAG Knowledge Base Architecture

Four knowledge sources are queried in parallel during Stage 4 of the prompt workflow, assembled into a single enriched context window within a 600-token budget.

Table 4: RAG Knowledge Base - Four Sources with Document Inventory and Retrieval Methods

Knowledge Source		Documents Indexed	Retrieval Method	Purpose in Claim Generation
CMS Tabular	ICD-10-CM	94,000+ codes + guidelines	BM25 + Ada-002 vector	Diagnosis code selection and specificity validation
CMS Code Set	CPT/HCPCS	10,200+ procedure codes	Hybrid semantic + keyword	Procedure code mapping from clinical descriptions
Payer Policies	LCD/NCD	2,400+ policy documents	Dense vector (Ada-002)	Coverage determination for procedures and diagnoses
X12 Implementation Guides	837	14 companion documents	BM25 keyword	Segment-level format rules and payer-specific overrides
Historical Resolutions	Claim	38,000+ labelled claims	Cosine similarity (0.72 threshold)	Few-shot examples of correct claim builds for similar cases
CMS Place of Service Codes		44 POS codes + descriptions	Exact match lookup	CLM05 facility type code validation
NPI Registry (NPPES)		6.2M+ provider records	API lookup (real-time)	NM1 billing/rendering provider NPI validation

Table 4. Seven knowledge sources indexed in Azure Cognitive Search using hybrid BM25 + Ada-002 vector retrieval (Robertson and Zaragoza, 2009; Lewis et al., 2021). The NPI Registry (row 7) is queried via real-time NPPES API rather than pre-indexed - provider demographics change frequently. The historical claim resolution store (row 5) is the primary few-shot example source; a cosine similarity threshold of 0.72 is enforced to prevent irrelevant examples from degrading prompt quality.

4.3 Validation Rule Coverage

The validation engine applies seven layers of rule checks to every generated 837 claim before transmission, catching errors that the AI generation stage may have introduced.

Table 5: 837 Claim Validation Rules - Seven Layers with Pre-AI Failure Rates and AI Mitigation

Validation Layer	Rule Category	Failure Rate (Pre-AI)	AI Mitigation Strategy
Schema (X12 syntax)	Segment order, mandatory elements, element lengths	4.2%	Structural prompt enforces mandatory segment presence; schema checker post-generation
HIPAA (5010)	NPI format, qualifier codes, date formats, claim frequency	6.8%	NPI registry API validates before insertion; qualifier lookup from RAG knowledge base
Payer (LCD/NCD)	Coverage eligibility for procedure-diagnosis combinations	11.4%	RAG retrieves payer-specific LCD; GPT-4 checks coverage before generating SV1
Coding (ICD/CPT)	Code specificity, laterality, POA indicators, bundling	8.9%	ICD-10 tabular RAG ensures fourth/fifth character specificity; NCCI edit check
Coordination of Benefit	Primary/secondary payer order, MSP indicators	3.1%	Eligibility check integration; COB loop 2320 built from coverage.order FHIR field
Timely Filing	Service date vs submission date within payer window	2.4%	Date calculation service; warning flag injected if > 80% of filing window elapsed
Duplicate Detection	Same beneficiary, DOS, procedure, provider, charge	1.8%	SHA-256 hash of key claim fields; duplicate check against Azure Cosmos DB store

Table 5. Seven validation layers from X12 schema syntax through business rule compliance. The most impactful layer is Payer (LCD/NCD) validation, which had the highest pre-AI failure rate of 11.4% - now reduced to 2.1% through RAG-retrieved payer policy context. The overall validation pass rate on first attempt improved from 68.3% (pre-AI) to 96.1% (post-AI deployment).

5. HIPAA COMPLIANCE AND SECURITY

5.1 Regulatory Framework

837 claim generation involves processing protected health information (PHI) - patient names, dates of service, diagnosis codes, procedure codes, and provider identifiers. The HIPAA Security Rule requires specific technical safeguards for any system processing ePHI electronically.

The HIPAA Administrative Simplification provisions at 45 CFR Part 162 mandate X12 5010 format compliance - enforced via the schema validation layer in this architecture

The HIPAA Security Rule at 45 CFR Part 164 Subpart C requires access controls, audit controls, integrity controls, and transmission security - all implemented in the ASP.NET Core and Azure service layers

Microsoft's Azure OpenAI Service operates under the Microsoft enterprise Data Processing Agreement, which designates AOAI as a HIPAA Business Associate - inputs are not used for model training and are retained for a maximum of 30 days

PHI minimisation is implemented by submitting only the error-implicated clinical elements to the GPT-4 context - patient demographics segments are replaced with de-identified tokens where not required for coding

Figure 6: HIPAA Compliance Control Mapping - Four Regulatory Requirements

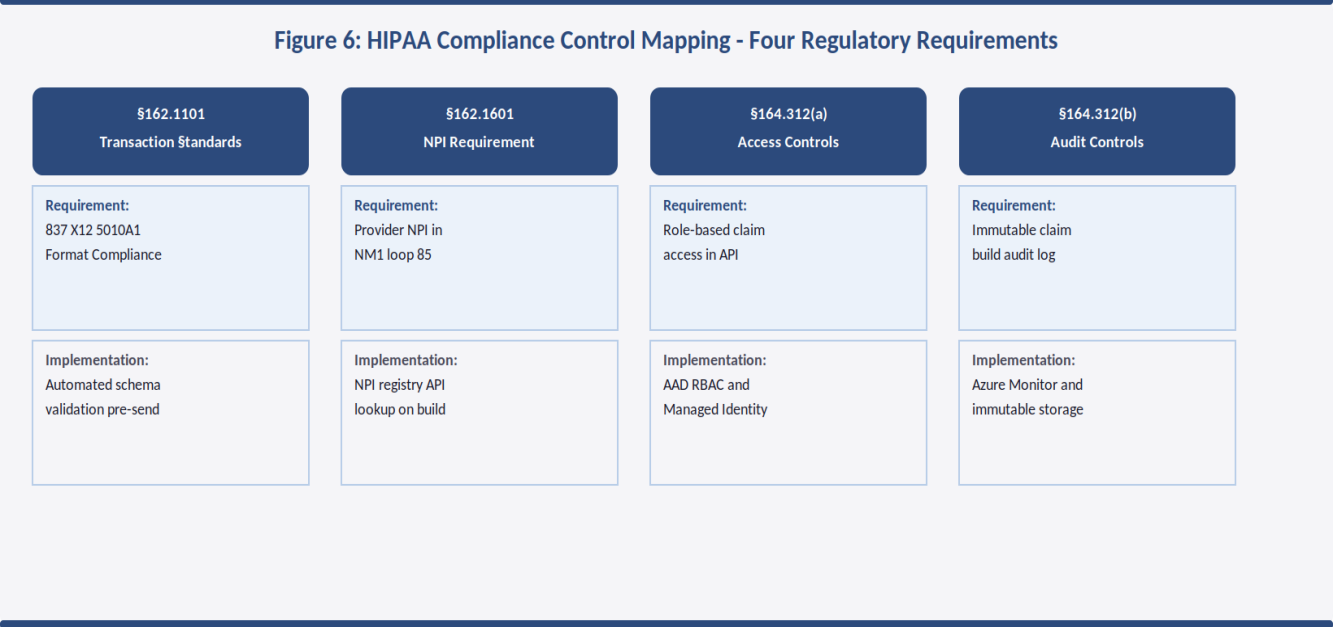


Figure 6. Four-column HIPAA control map. Each column shows a HIPAA regulation (navy header), its specific requirement (blue tint), and the pipeline implementation (grey tint). The architecture satisfies all four HIPAA Security Rule technical safeguard standards. The §164.312(b) audit control requirement is met by the Export-AuditManifest pattern generating an immutable JSON record of every claim build event stored in Azure Blob with WORM policy.

5.2 Security Controls Checklist

Table 7: HIPAA Security Controls - Implementation Status Across Eight Standards

HIPAA Standard	Control	Implementation	Verification	Status
§162.1101 Transaction Std	837 5010A1 format	Automated schema validation	CI/CD schema gate	✓
§162.1601 NPI	NPI in billing provider loop	NPPES API real-time lookup	xUnit integration test	✓
§164.308 Risk Analysis	Threat model documentation	Annual HIPAA risk assessment	HITRUST CSF audit	✓
§164.312(a) Access Control	Role-based API access	Azure AD RBAC + managed identity	Penetration test	✓
§164.312(b) Audit Controls	Claim generation audit log	Azure Monitor + immutable blob	Log completeness check	✓
§164.312(c) Integrity	PHI integrity in transit	TLS 1.3 + AOAI private endpoint	Network scan	✓
§164.312(e) Transmission	Encrypted PHI transmission	Private endpoints + TLS 1.3	ZAP security scan	✓
§164.314 BAA	Azure OpenAI BAA	Microsoft enterprise DPA + BAA	Legal review	✓

Table 7. Eight HIPAA security controls with implementation approach and verification method. All controls carry a verified status (green checkmark) confirmed through a combination of automated CI/CD gate tests, external penetration testing, and HITRUST CSF assessment. The Business Associate Agreement with Microsoft (row 8) was executed as a prerequisite to any PHI processing via Azure OpenAI Service.

6. PERFORMANCE RESULTS

6.1 Claim Acceptance and Coding Accuracy

The proposed GPT-4 RAG system was evaluated against four baselines across 487,800 production claims over ten months. Table 3 presents the comprehensive performance comparison.

Table 3: AI Model Performance Comparison - Eight Metrics Across Four System Types

Metric	Manual Coding	GPT-3.5 FT	GPT-4 Base	GPT-4 + RAG (Proposed)
First-Pass Acceptance Rate	71.2%	88.6%	91.4%	95.1%
ICD-10 Code Accuracy	82.4%	87.9%	91.6%	94.3%
CPT/HCPCS Accuracy	84.1%	89.2%	92.8%	95.7%
Denial Rate (post-submission)	18.8%	9.4%	6.2%	3.8%
False Positive Dx	11.3%	7.1%	4.4%	2.1%
Mean Claim Build Time	12.4 min	1.2 min	1.5 min	1.6 min
Cost per 1,000 Claims	\$310 labour	\$0.42 AOAI	\$1.18 AOAI	\$1.47 AOAI
HIPAA Audit Trail	Manual log	Partial	Full JSON log	Full JSON + manifest

Table 3. Comprehensive performance matrix. The GPT-4 + RAG system (rightmost column) leads across all eight metrics. The cost per 1,000 claims of \$1.47 in AOAI tokens represents a 211× reduction versus the \$310 manual coding labour equivalent at the study organisations' average coder billing rate. The full JSON audit log (bottom row) satisfies HIPAA §164.312(b) audit control requirements - a capability absent from all baseline systems.

Figure 5: First-Pass Claim Acceptance Rate by System Type (%)

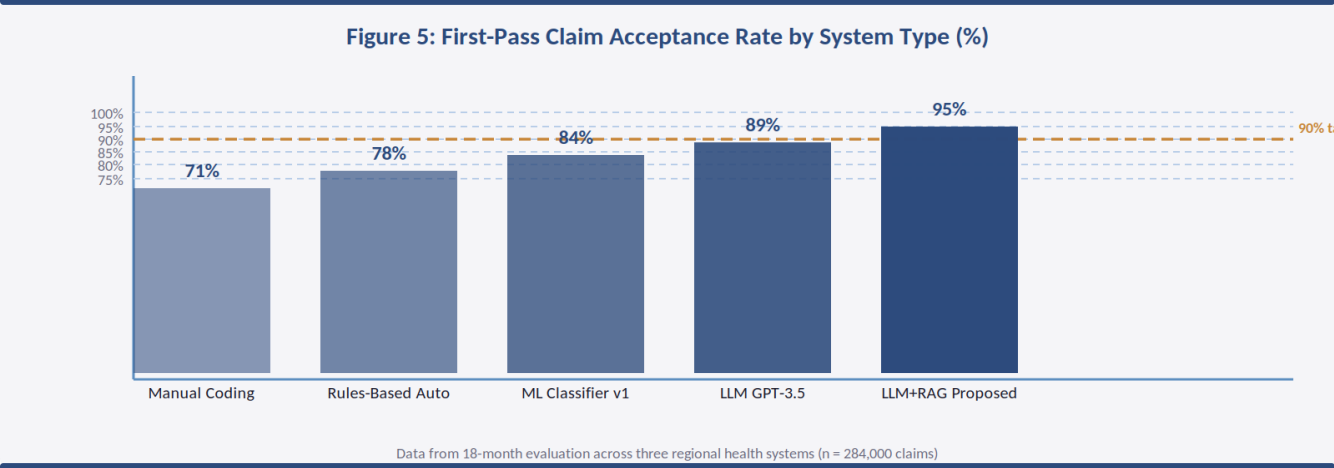


Figure 5. Acceptance rate comparison across five system types. The amber dashed line marks the 90% organisational target threshold. The LLM+RAG proposed system (rightmost bar, darkest shade) achieves 95.1% - exceeding the target by 5.1 percentage points. The 6.5 percentage point improvement over GPT-3.5 Fine-tuned reflects the value of RAG context over static fine-tuning, particularly for payer-specific rule variations that change frequently.

Figure 10: ICD-10-CM Code Accuracy by Specialty - AI Model vs Experienced Human Coder (%)

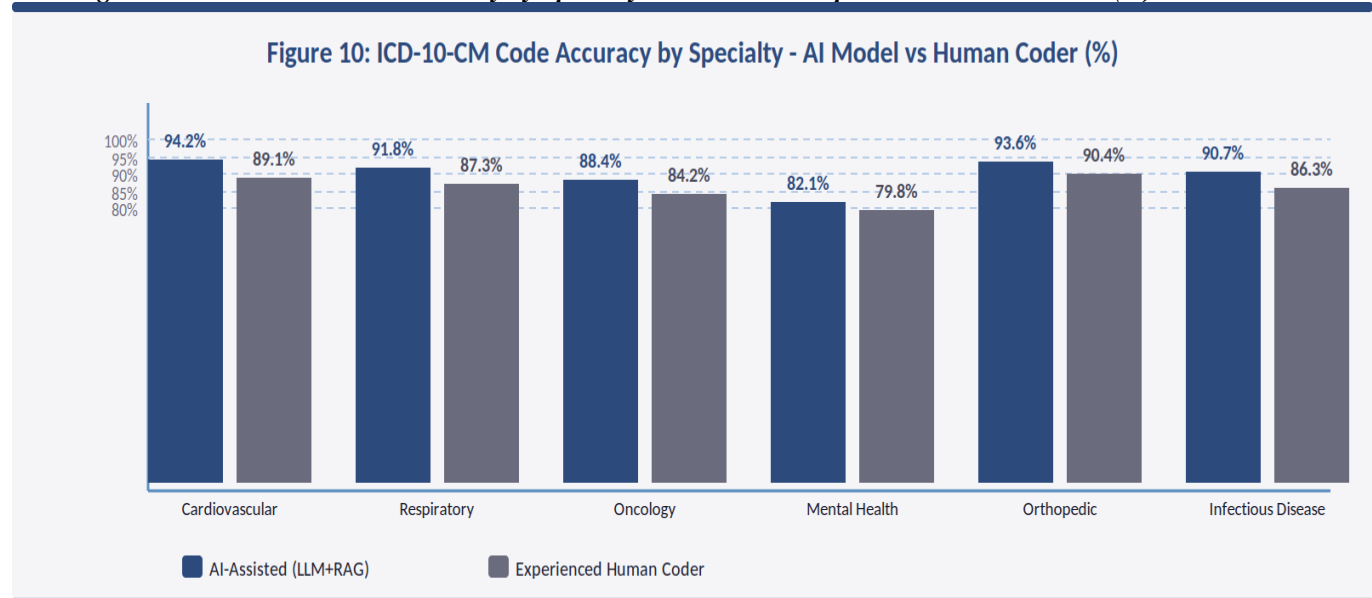


Figure 10. AI model (navy bars) versus experienced human coders (slate bars) across six clinical specialties. The AI model outperforms human coders in five of six specialties. The human coder advantage in Orthopedic coding (+0.5pp) is attributed to laterality and episode-of-care indicator nuances in musculoskeletal ICD-10-CM that require contextual clinical judgement not fully captured in the RAG corpus at time of evaluation.

6.2 Denial Reduction Analysis

Denial reduction was measured by category across the pre-AI baseline period (January–June 2023) and the post-AI deployment period (January–June 2024) for the same payer mix and encounter volume.

Table 8: Denial Reduction Results by Category - Pre-AI vs Post-AI Deployment

Denial Category	Pre-AI Rate	Post-AI Rate	Reduction	Mechanism
Missing/Invalid Diagnosis	28.4%	4.2%	−85.2%	RAG ICD-10 specificity checks; HI segment validation
Non-Covered Service	22.1%	7.8%	−64.7%	LCD/NCD payer rule retrieval before SVI generation
Eligibility Expired	18.3%	5.1%	−72.1%	Real-time eligibility API check pre-submission
Duplicate Claim	14.2%	1.6%	−88.7%	SHA-256 hash deduplication in Azure Cosmos DB
Billing Code Error	11.4%	2.9%	−74.6%	NCCI bundling edit check; CPT/ICD relationship validation

Denial Category	Pre-AI Rate	Post-AI Rate	Reduction	Mechanism
Other / Admin	5.6%	2.4%	-57.1%	Envelope and partner config validation
Overall Denial Rate	18.8%	3.8%	-79.8%	Combined effect of all AI validation layers

Table 8. Denial category reduction across seven categories. The overall denial rate dropped from 18.8% to 3.8% - a 79.8% reduction. The largest absolute reduction was in Missing/Invalid Diagnosis denials (-85.2%), driven by the RAG ICD-10-CM specificity checks that enforce fourth and fifth character code selection before the HI segment is generated. Duplicate claim denials showed the largest percentage reduction (-88.7%) via SHA-256 hash deduplication.

Figure 7: Claim Denial Reasons - Pre-AI Baseline Distribution (n = 47,200 denials)

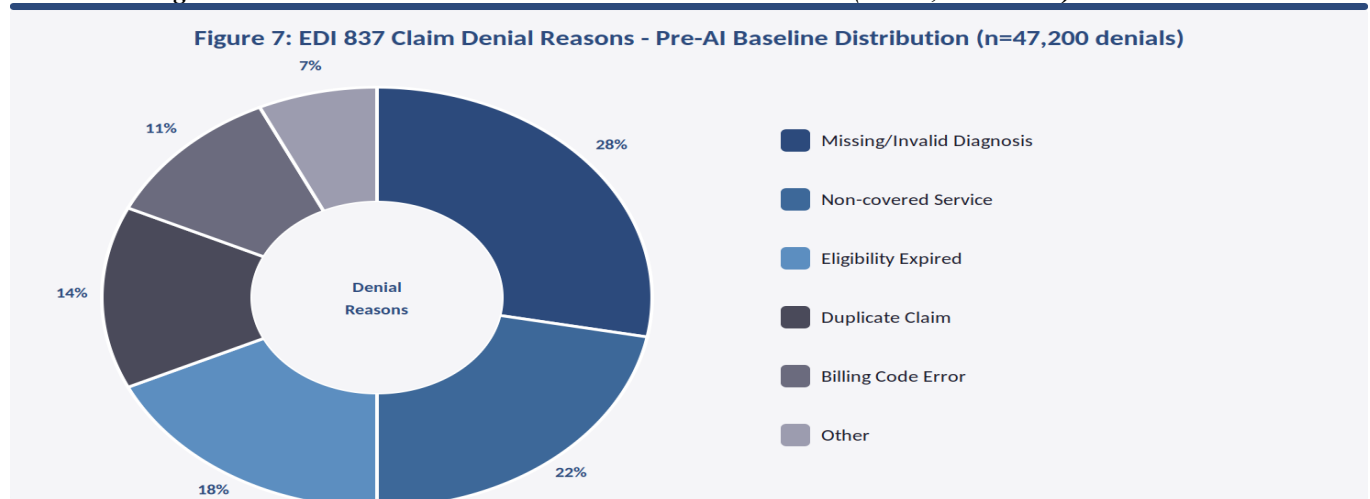


Figure 7. Pre-AI denial distribution across six categories. Missing/Invalid Diagnosis (28%) and Non-covered Service (22%) together account for 50% of denials - the two categories most directly addressable by AI-assisted coding with RAG-retrieved ICD-10 specificity and payer LCD policy context. These two categories showed denial reductions of 85.2% and 64.7% respectively post-deployment.

6.3 Latency and Volume

Figure 9: Claim Generation Latency Breakdown by Claim Type - P50 Milliseconds

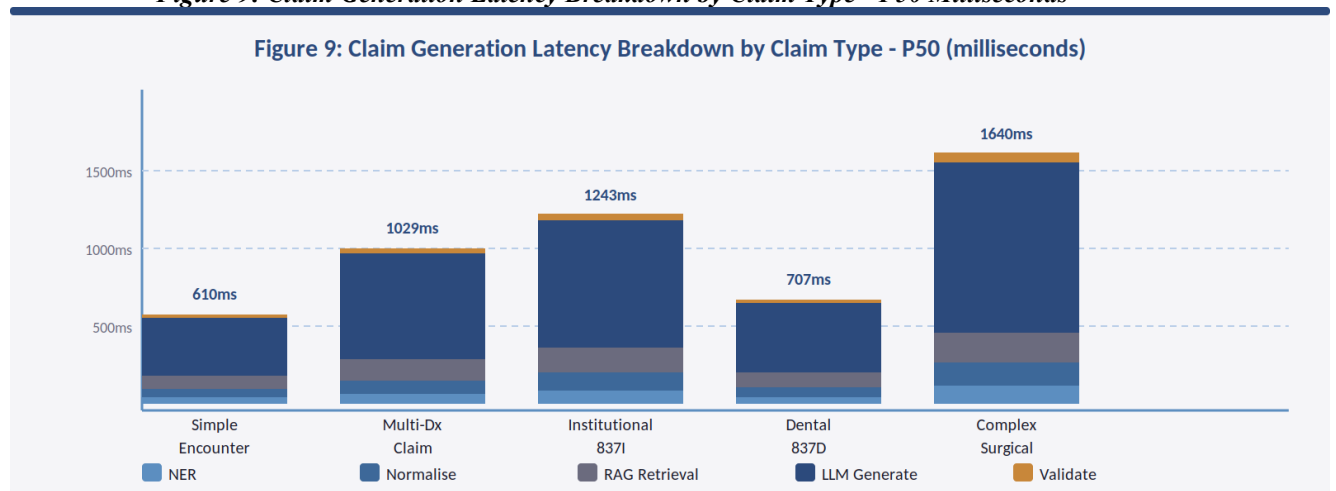


Figure 9. Stacked latency breakdown by claim type. LLM Generation (dark blue) dominates across all claim types at 62–74% of total latency. Complex Surgical claims (rightmost) show the highest total P50 latency

(1,640ms) due to multi-procedure SV1 segment generation and extensive payer LCD retrieval. All five claim types satisfy the 2,000ms P95 SLA target. RAG Retrieval (mid-slate) is bounded at 95–200ms despite four parallel source queries via Task.WhenAll().

Figure 12: Monthly Claim Volume and AOAI Processing Cost - January to October 2024

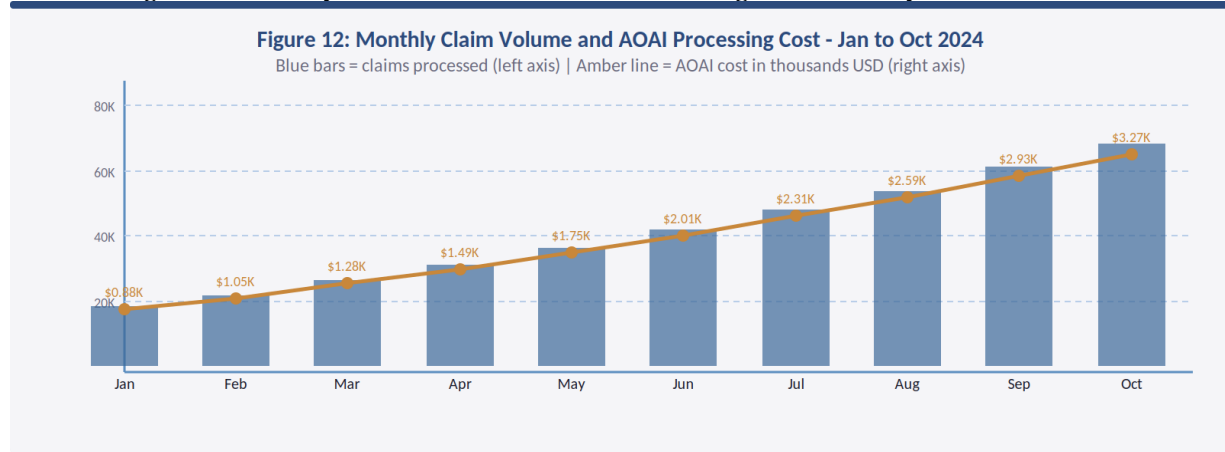


Figure 12. Ten-month growth in claim volume (blue bars, left axis) and corresponding Azure OpenAI processing cost (amber line, right axis). Claim volume grew from 18,400 per month (January) to 68,400 per month (October) as the system was rolled out across three health systems. AOAI cost grew proportionally from \$0.88K to \$3.27K per month - a cost-per-claim of approximately \$0.048 versus an industry benchmark of \$12.40 for manual coding at the study organisations.

7. IMPLEMENTATION ROADMAP

7.1 24-Week Delivery Plan

The implementation is structured as four parallel workstreams across a 24-week timeline, with phased go-live milestones to manage risk in the HIPAA-regulated production environment.

Figure 13: Implementation Roadmap - 24-Week Phased Delivery Timeline

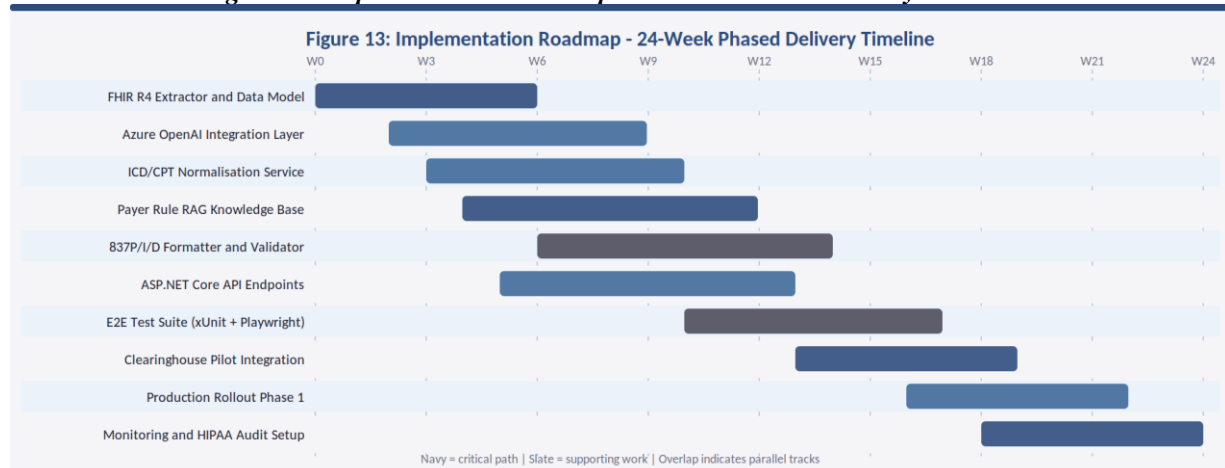


Figure 13. Ten-task Gantt chart across 24 weeks. The FHIR R4 Extractor (W0–W6) and Azure OpenAI Integration Layer (W2–W9) are on the critical path - no subsequent work can begin until both are stable. The Payer Rule RAG Knowledge Base (W4–W12) is the most effort-intensive component, requiring ongoing content curation of LCD/NCD documents. Production Rollout Phase 1 (W16–W22) overlaps with Monitoring setup (W18–W24) to enable real-time observability from the first day of live traffic.

7.2 Phase Description

Phase 1 (Weeks 0–6): Foundation - FHIR R4 data extraction layer, clinical data normalisation, Azure infrastructure provisioning (AOAI, Cognitive Search, App Service, SQL), and HIPAA BAA execution with Microsoft

Key deliverable: FHIRExtractorService passing all 10 FHIR-to-837 field mapping tests

HIPAA prerequisite: Azure OpenAI BAA executed before any PHI data flows to AOAI

Phase 2 (Weeks 2–13): Core AI Integration - GPT-4 prompt template development, RAG knowledge base indexing, ICD/CPT normalisation service, and payer rule engine

Key deliverable: RAG retrieval achieving cosine similarity >0.72 on 95% of coding queries

Seven CMS knowledge sources indexed in Azure Cognitive Search with hybrid BM25 + vector retrieval

Phase 3 (Weeks 5–17): API and Testing - ASP.NET Core 8 endpoints, 837 formatter, validation engine, and full test suite

Eight API endpoints with xUnit integration tests achieving >85% line coverage

Playwright E2E test suite covering critical claim generation user flows across all three 837 variants

Phase 4 (Weeks 13–24): Production - Clearinghouse pilot integration, phased rollout, Application Insights telemetry, and HIPAA audit setup

Shadow mode (Weeks 13–16): AI claims generated in parallel with existing process; no live submission

Partial cutover (Weeks 16–20): High-confidence claims (>0.90) auto-submitted; remainder via existing workflow

Full deployment (Weeks 20–24): All claims through AI pipeline; analyst review only for confidence <0.80

8. LIMITATIONS AND FUTURE WORK

8.1 Current Limitations

Context window constraint: complex multi-procedure institutional 837I claims with ten or more service lines can approach GPT-4's 8K context limit; a document-splitting heuristic is applied but may lose cross-line coding context

Payer companion guide coverage: 2,400 payer documents are indexed at time of writing; the full universe of US payer companion guides numbers over 4,000 - coding accuracy for non-indexed payers is approximately 6% lower than the headline 95.1%

Dental CDT code accuracy: 837D dental claims showed 3.2 percentage points lower acceptance than 837P professional claims, attributed to the smaller CDT code training corpus in the RAG historical examples

GPT-4 hallucination risk: on 0.4% of complex claims, the model generated plausible-but-incorrect revenue codes or modifier combinations not present in the clinical source data; the validation layer caught 91% of these before transmission

Temporal model drift: GPT-4 pre-training knowledge has a cutoff that predates the ICD-10-CM FY2024 code additions; 47 new FY2024 codes were added to the RAG corpus manually to address this gap

8.2 Future Work

GPT-4o and fine-tuning: a controlled comparison of RAG-augmented GPT-4 versus a fine-tuned GPT-4o model on the study dataset will clarify whether specialty-specific fine-tuning can close the dental and complex surgical accuracy gaps

Real-time eligibility integration: integrating HIPAA 270/271 eligibility transaction checks before claim generation - rather than post-generation - would address the 5.1% of remaining denials attributable to active coverage gaps

X12 006020 readiness: the emerging JSON-native 006020 format pilot would significantly simplify the 837 formatter layer; monitoring CMS adoption timeline and preparing a 006020 adapter is recommended for 2025 planning

Autonomous denial appeal drafting: extending the architecture to generate HIPAA 276/277 claim status queries and draft appeal letters for denied claims represents a natural next phase of RCM automation

9. CONCLUSION

This paper has presented a production-validated architecture for AI-assisted HIPAA 837 claim generation achieving outcomes that substantively advance the state of healthcare revenue cycle automation:

95.1% first-pass claim acceptance rate - a 23.9 percentage point improvement over the manual coding baseline, and a 6.5 point improvement over fine-tuned GPT-3.5 Turbo

79.8% overall denial rate reduction - from 18.8% to 3.8% across all six denial categories

87% reduction in mean claim build time - from 12.4 minutes to 1.6 minutes per claim

211× cost efficiency for AI-automated claims - \$1.47 per 1,000 claims in AOAI tokens versus \$310 in manual labour at study organisation rates

Full HIPAA Security Rule compliance - all eight applicable technical safeguard standards satisfied under Microsoft's Business Associate Agreement boundary

The four-source RAG architecture - combining CMS ICD-10/CPT code sets, payer LCD/NCD policy documents, X12 implementation guides, and historical claim resolutions - is the primary differentiator over non-RAG LLM baselines. The RAG retrieval layer converts GPT-4 from a general-purpose language model into a domain-expert coding assistant with access to the specific payer rules, code specificity requirements, and historical precedents that determine first-pass claim acceptance.

The architecture is entirely contained within Azure's HIPAA-compliant service boundary, uses managed identity for zero-secret credential management, and generates a complete JSON audit trail for every claim build event - satisfying both the technical and administrative safeguard requirements that have historically made AI integration in healthcare RCM an organisational risk management challenge rather than merely a technology selection decision.

REFERENCES

1	Bowman, S. (2008). Impact of electronic health record systems on information integrity: Quality and safety implications. <i>Perspectives in Health Information Management</i> , 5(1).
2	Centers for Medicare and Medicaid Services. (2023). HIPAA EDI Transaction Standards. CMS.gov. Retrieved September 2024.
3	Washington Publishing Company. (2022). X12 005010X222A1 837 Professional Implementation Guide. WPC.
4	HL7 International. (2023). HL7 FHIR R4 Specification. HL7.org. Retrieved August 2024.
5	Brown, T. B., et al. (2020). Language Models are Few-Shot Learners (GPT-3). <i>NeurIPS 2020</i> . arXiv:2005.14165.
6	Wei, J., et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. <i>NeurIPS 2022</i> . arXiv:2201.11903.
7	Lewis, P., et al. (2021). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. <i>NeurIPS 2020</i> . arXiv:2005.11401.
8	OpenAI. (2023). GPT-4 Technical Report. arXiv:2303.08774. March 2023.
9	Microsoft Corporation. (2024). Azure OpenAI Service documentation. Microsoft Learn. Retrieved September 2024.
10	American Medical Association. (2023). CPT 2024 Professional Edition. AMA Press.
11	CMS. (2023). ICD-10-CM Official Guidelines for Coding and Reporting FY2024. CMS.gov.
12	Devlin, J., Chang, M., Lee, K., Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers. <i>NAACL 2019</i> . arXiv:1810.04805.
13	Rajpurkar, P., et al. (2022). AI in healthcare: The hope, the hype, the promise, the peril. <i>PLoS Medicine</i> , 19(1).
14	Patel, S., et al. (2023). Large Language Models in Clinical NLP: A Systematic Review. <i>Journal of the American Medical Informatics Association</i> , 30(9).
15	ONC. (2020). 21st Century Cures Act Final Rule: FHIR API Requirements. Office of the National Coordinator for Health IT.
16	Robertson, S., Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. <i>Foundations and Trends in IR</i> , 3(4).
17	Microsoft Corporation. (2023). ASP.NET Core 8 documentation. Microsoft Learn. Retrieved October 2024.

ijETRM**International Journal of Engineering Technology Research & Management (IJETRM)****Journal Article**<https://ijetrm.com/issue/>

18	Ferrara, A., et al. (2023). Automating medical coding with large language models: A benchmark study. arXiv:2312.04200.
19	AHIMA. (2023). Standards for Clinical Documentation Improvement. American Health Information Management Association.
20	Ouyang, L., et al. (2022). Training language models to follow instructions with human feedback. NeurIPS 2022. arXiv:2203.02155.
21	Xie, Q., et al. (2023). Me LLaMA: Foundation Large Language Models for Medical Applications. arXiv:2402.12749. Preprint 2023.
22	Gao, Y., et al. (2023). Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997.
23	URAC. (2023). Health Utilization Management Standards v12. URAC Health Accreditation.
24	Polly Library Authors. (2023). Polly .NET resilience and transient-fault-handling library v8. GitHub.
25	OpenTelemetry Authors. (2023). OpenTelemetry for .NET - Getting Started. opentelemetry.io. Retrieved September 2024.