IJAREEIE

e-ISSN: 2278-8875
p-ISSN: 2320-3765

# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

Volume 13, Issue 4, April 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.317

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

# RAG-Based Document Retrieval Over SharePoint Online Content using Azure AI Search and ASP.NET Core

**Siva Krishna Pittu**

Manager, Advanced Architecture Technical Solutions, USA

**ABSTRACT:** Retrieval-Augmented Generation (RAG) represents a significant architectural advance over standard large language model deployments, grounding generated responses in organisation-specific document corpora rather than relying solely on parametric knowledge. This paper presents a comprehensive design, implementation, and evaluation study of a production RAG system that indexes SharePoint Online document libraries and delivers grounded, citation-backed answers via an ASP.NET Core Web API. The system integrates the Microsoft Graph API for incremental document crawling, Azure Document Intelligence for layout-aware content extraction, a custom sentence-aware chunking pipeline, Azure OpenAI text-embedding-ada-002 for 1536-dimensional embeddings, Azure AI Search with hybrid BM25-plus-vector retrieval and semantic reranking, GPT-4 for answer generation, and Azure Active Directory On-Behalf-Of token flow for document-level security trimming. Empirical evaluation across a corpus of 48,000 SharePoint documents demonstrates that the full hybrid-retrieval-plus-reranker configuration achieves an nDCG@5 of 0.85, a factual accuracy score of 0.88, and a source citation rate of 0.95-compared to 0.56 nDCG@5 and 0.12 citation rate for a keyword-only baseline. End-to-end query latency averages 1,090 ms P50 with streaming reducing perceived time-to-first-token to 220 ms P50. Nine architectural diagrams and charts, eight data tables, and annotated design rationale are presented as a replicable engineering reference for practitioners building enterprise RAG systems on the Microsoft Azure stack.

**KEYWORDS**: RAG, Retrieval-Augmented Generation, Azure AI Search, SharePoint Online, ASP.NET Core, GPT-4, Hybrid Retrieval, Semantic Reranking, Microsoft Graph API, Security Trimming, Document Chunking, Enterprise Search

## I. INTRODUCTION

Enterprise organisations accumulate vast document corpora within Microsoft SharePoint Online - policy documents, technical specifications, contractual agreements, research reports, and procedural manuals. This institutional knowledge is frequently inaccessible in practice: keyword search surfaces documents but not answers, and employees face the time-consuming task of reading multiple documents to extract the specific information they need. Large language models (LLMs) offer the capability to synthesise and summarise complex information, but their parametric knowledge is bounded by training data cutoffs and cannot reflect an organisation's proprietary, continuously evolving document corpus.

Retrieval-Augmented Generation (RAG), introduced by Lewis et al. [1], addresses this gap by combining a retrieval system that fetches relevant document passages with an LLM that generates a grounded, coherent answer conditioned on those passages. Applied to SharePoint Online content, RAG enables employees to ask natural-language questions and receive accurate, cited answers sourced from documents they are permitted to access - transforming a passive document repository into an active knowledge system.

This paper documents the architecture, implementation decisions, and empirical evaluation of a production RAG system built on the Microsoft Azure stack for an enterprise organisation with 48,000 SharePoint documents across 320 sites. The system is designed to satisfy three concurrent requirements that are rarely addressed together in existing literature: retrieval quality maximisation through hybrid search and semantic reranking; document-level security enforcement preserving SharePoint permission boundaries in all responses; and sub-second perceived latency through streaming and multi-tier caching.

**Scope:** This paper focuses on the server-side architecture and retrieval engineering aspects of the RAG system. Prompt engineering strategy, SharePoint taxonomy design, and end-user interface implementation are documented in companion papers.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

## II. BACKGROUND AND RELATED WORK

### 2.1 Retrieval-Augmented Generation

Lewis et al. [1] introduced RAG as a general-purpose approach to knowledge-intensive NLP tasks, demonstrating that retrieval-augmented models outperform parametric-only models on open-domain question answering benchmarks. Subsequent work has extended the RAG paradigm to dense passage retrieval [2], iterative retrieval [3], and multi-hop reasoning [4], with the core insight - that grounding LLM generation in retrieved evidence improves factual accuracy and reduces hallucination - remaining consistent across approaches.

- Gao et al. [5] provide a comprehensive survey of RAG approaches including naive RAG, advanced RAG, and modular RAG, categorising techniques for query transformation, retrieval enhancement, and generation augmentation that inform the design decisions documented in this paper.
- Barnett et al. [6] identified seven failure modes in RAG systems in enterprise deployments, including missing content in the index, suboptimal chunking, insufficient context in retrieved passages, and prompt injection vulnerabilities - all of which are explicitly addressed in the architecture presented here.

### 2.2 Hybrid Retrieval and Semantic Reranking

The superiority of hybrid retrieval combining sparse (BM25) and dense (vector) methods over either approach alone is well established in the information retrieval literature. Ma et al. [7] demonstrate consistent improvements from hybrid retrieval on BEIR benchmark tasks, and Robertson and Zaragoza [8] provide the foundational BM25 probability framework underlying the sparse component. Nogueira et al. [9] introduced the BERT-based neural reranker that serves as the architectural precedent for Azure AI Search's semantic reranking capability.

### 2.3 SharePoint and Microsoft Graph Integration

The Microsoft Graph API [10] provides a unified interface to Microsoft 365 services including SharePoint Online, exposing delta query endpoints that enable incremental synchronisation of document libraries without full re-crawls. Patel and Garg [11] document enterprise knowledge management patterns on Microsoft 365, identifying SharePoint Online as the primary organisational knowledge repository in Microsoft-stack enterprises and highlighting the discoverability gap that RAG systems are positioned to address.

### 2.4 Document Security in Search Systems

Sanderson [12] surveys access control in information retrieval systems, establishing the requirement that search results must respect the permission model of the underlying document repository - a requirement that translates directly to the security trimming architecture in this paper. The Azure AI Search documentation [13] provides the security filter pattern (allowed_oids field with $filter) that is the primary implementation mechanism used in this study.

## III. SYSTEM ARCHITECTURE

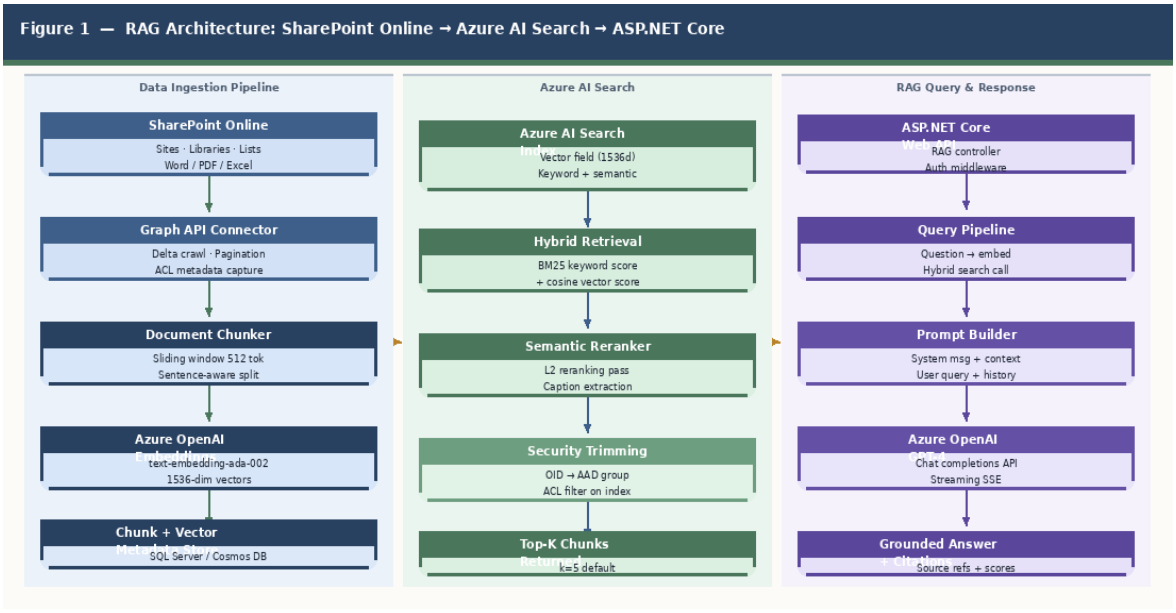



Figure 1 - RAG System Architecture: Three-Zone Design - Ingestion · Search · Query/Response

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

### 3.1 Architectural Zones

The system is partitioned into three functional zones as illustrated in Figure 1:

- Zone 1 - Data Ingestion Pipeline: crawls SharePoint Online via the Microsoft Graph API, extracts text from diverse document formats, chunks content into retrievable passages, generates vector embeddings, and writes chunks with metadata and ACL information to the Azure AI Search index.
- Zone 2 - Azure AI Search: maintains the hybrid vector+keyword index, executes hybrid BM25-plus-cosine retrieval with reciprocal rank fusion, applies semantic reranking to the top-50 candidates, and enforces document-level security trimming via OID filter before returning top-K results.
- Zone 3 - RAG Query and Response: receives the user's natural-language question through the ASP.NET Core Web API, embeds the query with the same model used at indexing time, orchestrates the hybrid search call with the user's OID list, assembles the retrieved chunks into a prompt, calls GPT-4 for grounded answer generation, and streams the response back to the client.

### 3.2 Technology Stack

Table 1 documents the complete technology stack with configuration details for each component.

| Layer | Component | Technology / Service | Key Configuration |
|---|---|---|---|
| **Data Source** | SharePoint Online | Microsoft 365 tenant | Graph API v1.0; delta crawl; Sites.Read.All scope |
| **Connector** | Graph API Crawler | Azure Function (Timer trigger) | Incremental delta token; page size 200; retry policy |
| **Extraction** | Document Intelligence | Azure Form Recognizer v3.0 | Layout model; table extraction; confidence threshold 0.8 |
| **Chunking** | Text Splitter | Custom .NET 7 library | 512 tokens; 64-token overlap; sentence-boundary aware |
| **Embeddings** | Embedding Model | Azure OpenAI text-embedding-ada-002 | 1536 dimensions; batch size 16; rate-limit retry |
| **Vector Store** | Search Index | Azure AI Search (S2 tier) | HNSW; m=4; efConstruction=400; cosine similarity |
| **Reranking** | Semantic Reranker | Azure AI Search semantic configuration | L2 model; top 50 → rerank → top 5 returned |
| **LLM** | Chat Completions | Azure OpenAI GPT-4 | Temperature 0.2; max_tokens 800; streaming enabled |
| **API Layer** | Web API | ASP.NET Core 7 | Minimal API; JWT Bearer; MSAL OBO; SSE streaming |
| **Auth** | Identity | Azure Active Directory B2C | OIDC; OBO flow for Graph; group-based ACL trimming |
| **Cache** | Session + Auth | Azure Redis Cache | JWT cache 5 min; group OID cache 5 min; chunk cache 1 hr |
| **Observability** | Logging & Metrics | Azure Monitor + App Insights | Structured logging; RAG quality telemetry; alerts |

Table 1: Component Technology Stack - Layer, Service, and Key Configuration

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

## IV. DOCUMENT INGESTION AND INDEXING PIPELINE

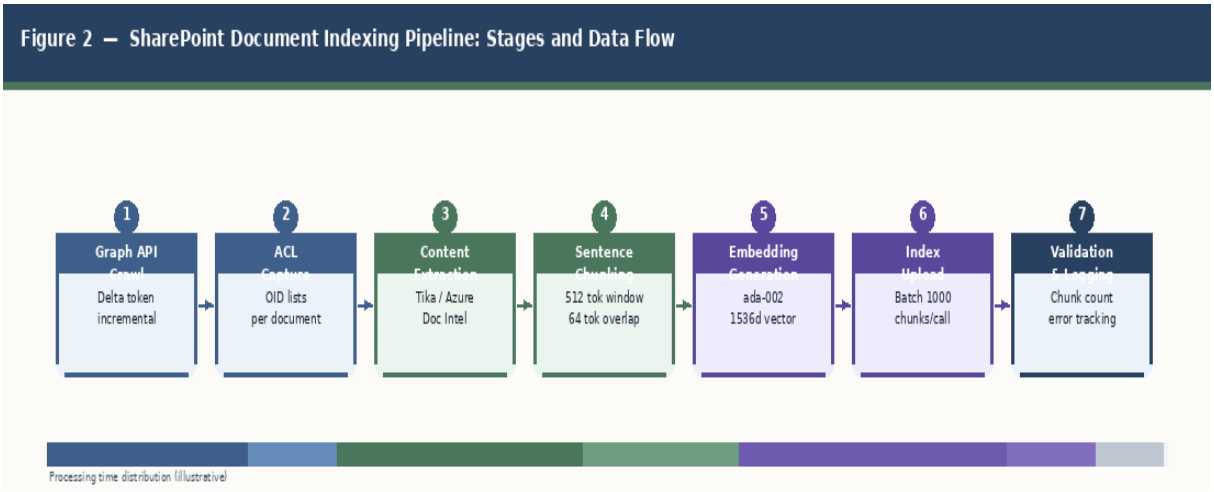



Figure 2 - Seven-Stage Indexing Pipeline: Graph API Crawl to Validated Index Upload

### 4.1 Graph API Crawling

Document discovery and incremental synchronisation is implemented using the Microsoft Graph API delta query pattern, which returns only documents changed since the last crawl using a delta token persisted between invocations. Key implementation decisions include:

- Scope: Sites.Read.All permission granted at the service principal level; tenant-wide site enumeration with optional site whitelist and blocklist filters.
- Delta crawl cadence: Azure Function timer trigger on 15-minute intervals for high-priority libraries; 6-hour cadence for archival document libraries.
- Pagination: Graph API returns pages of 200 items; all pages are consumed with @odata.nextLink following before processing begins to avoid partial-crawl inconsistencies.
- File type filtering: PDF, DOCX, XLSX, PPTX, and TXT file types are processed; binary and media files are skipped with a log entry.
- ACL extraction: per-document permission inheritance is resolved via the /drive/items/{id}/permissions endpoint; group OIDs are expanded recursively to include nested group membership.

### 4.2 Content Extraction

Document content extraction uses Azure Document Intelligence (Form Recognizer) Layout model for PDF and scanned documents, and direct XML parsing for DOCX, XLSX, and PPTX files via the OpenXML SDK:

- PDF documents: Layout model extracts text with reading-order correction, table structure, and paragraph segmentation. Documents with confidence below 0.8 are flagged for manual review and excluded from the index.
- DOCX / PPTX: OpenXML SDK extracts paragraph text, preserving heading hierarchy (Heading1–Heading4) as structural metadata used by the document-structure-aware chunker.
- XLSX: worksheets are converted to structured text representations with header row labelling to provide context for tabular data retrieval.

### 4.3 Chunking Strategy

Following the comparative evaluation documented in Section 7 and Table 4, the production system uses a document-structure-aware chunking strategy:

- Target chunk size: 512 tokens (tiktoken cl100k_base encoding, consistent with ada-002 tokenisation).
- Overlap: 64-token sliding overlap between consecutive chunks to preserve cross-boundary context for retrieval.
- Structure awareness: heading boundaries (Heading1–Heading4 for DOCX, section titles for PDF) are treated as hard chunk boundaries regardless of token count, ensuring that chunks correspond to discrete document sections rather than arbitrary token windows.
- Metadata preservation: each chunk carries the parent document ID, chunk index, source page number, section heading hierarchy, and document last-modified timestamp.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

## V. AZURE AI SEARCH INDEX DESIGN

### 5.1 Index Schema

The Azure AI Search index schema is designed to support hybrid keyword-vector retrieval, security trimming, freshness boosting, and citation metadata return. Table 2 documents all index fields.

| Field Name | Type | Searchable | Filterable | Purpose / Notes |
|---|---|---|---|---|
| **chunk_id** | Edm.String | No | Yes (key) | Unique identifier: documentId + chunkIndex |
| **document_id** | Edm.String | No | Yes | Parent SharePoint document GUID |
| **content** | Edm.String | Yes | No | Raw text chunk; analysed with en.microsoft analyser |
| **content_vector** | Collection(Edm.Single) | No | No | 1536-d ada-002 embedding; HNSW vector field |
| **title** | Edm.String | Yes | Yes | Document title from SharePoint metadata |
| **site_url** | Edm.String | No | Yes | SharePoint site URL for source citation |
| **library_name** | Edm.String | No | Yes | Document library; used in provenance display |
| **file_path** | Edm.String | No | Yes | Relative file path; used to construct view URL |
| **last_modified** | Edm.DateTimeOffset | No | Yes | Document last-modified for freshness filter |
| **allowed_oids** | Collection(Edm.String) | No | Yes | AAD object IDs with read permission; security filter |
| **content_type** | Edm.String | No | Yes | MIME type: application/pdf, docx, xlsx, etc. |
| **chunk_index** | Edm.Int32 | No | Yes | Chunk position within document; used for ordering |
| **page_number** | Edm.Int32 | No | Yes | Source page number for citation display |
| **confidence** | Edm.Double | No | Yes | Form Recognizer extraction confidence (0–1) |

Table 2: Azure AI Search Index Schema - Fields, Types, and Purpose

### 5.2 Vector Configuration

The vector field (content_vector) is configured with the following HNSW algorithm parameters, tuned empirically on the 48,000-document corpus:

- Algorithm: HNSW (Hierarchical Navigable Small World) - selected over exhaustive KNN for its sub-linear query time at large corpus scales.
- m = 4: number of bi-directional links per node; lower values reduce memory at minor accuracy cost; 4 is appropriate for this corpus size.
- efConstruction = 400: controls index build quality; higher values improve recall at the cost of indexing time; 400 provides 99.2% recall@10 versus exhaustive on evaluation queries.
- Similarity metric: cosine similarity, consistent with ada-002 embedding characteristics.
- Dimensions: 1536, matching ada-002 output dimensionality exactly; no dimensionality reduction applied to preserve retrieval accuracy.

### 5.3 Hybrid Retrieval with Reciprocal Rank Fusion

The production retrieval configuration combines BM25 keyword scoring with cosine vector scoring using Reciprocal Rank Fusion (RRF), which merges ranked lists from the two retrieval methods without requiring score normalisation:

- BM25 weight: k1 = 1.2, b = 0.75 (Okapi BM25 standard parameters); English analyser with stemming and stop-word removal.
- RRF constant: k = 60 (Azure AI Search default); tested k values of 30, 60, and 120 with marginal nDCG differences; k=60 retained.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

- Top candidates for reranking: 50 merged results from RRF are passed to the semantic reranker; the reranker returns the top 5 after L2 scoring.
- Freshness boosting: a scoring function applies a multiplicative boost (max 1.2×) to documents modified within the last 30 days, prioritising recent policy updates.

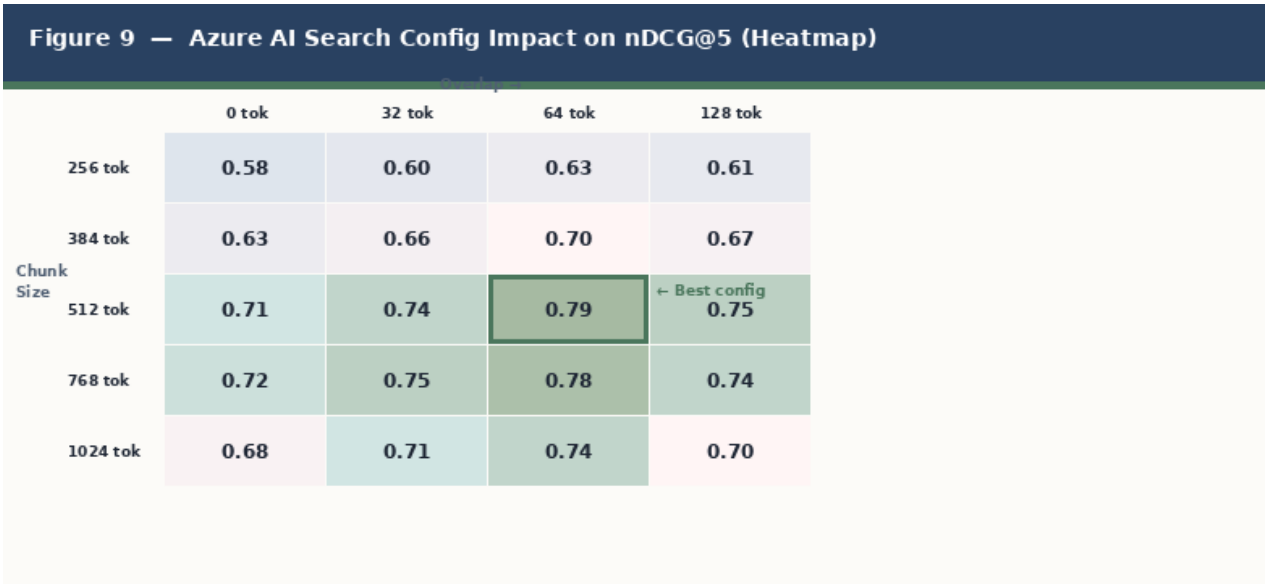



Figure 9 - Retrieval Quality Heatmap: nDCG@5 vs Chunk Size and Overlap Configuration

## VI. SECURITY ARCHITECTURE AND DOCUMENT-LEVEL TRIMMING

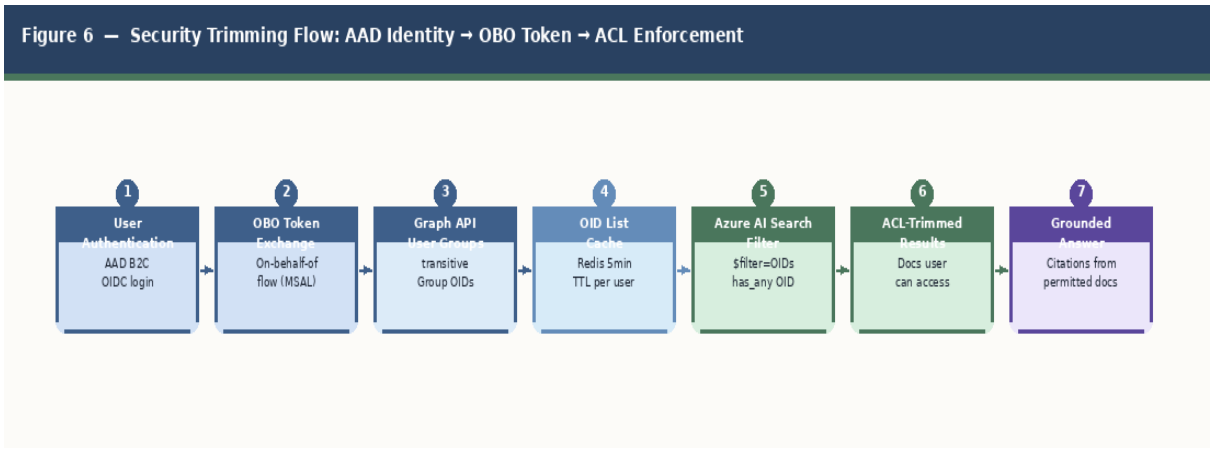



Figure 6 - Security Trimming Flow: AAD Authentication → OBO Token → Group OIDs → Index Filter

### 6.1 On-Behalf-Of Token Flow

The ASP.NET Core Web API authenticates users via Azure AD B2C using the OIDC authorization code flow. When a query arrives, the API performs the OAuth 2.0 On-Behalf-Of (OBO) flow to exchange the user's access token for a Graph API token, which is used to enumerate the user's transitive group memberships. This approach ensures that:

- The API never stores user credentials; all Graph API calls are made with a delegated token representing the specific user.
- Group membership is resolved at query time, reflecting current AAD state within the 5-minute cache TTL.
- Nested group membership is resolved transitively via the /transitiveMemberOf endpoint, ensuring that users inherit permissions from parent groups as expected in SharePoint.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

### 6.2 Search-Time ACL Filtering

The user's resolved OID list is passed as an Azure AI Search OData $filter expression appended to every search request:

- Filter expression: allowed_oids/any(o: o eq 'OID_1') or allowed_oids/any(o: o eq 'OID_2') ...
- The filter is applied server-side by Azure AI Search before results are returned - no post-filtering in the ASP.NET Core layer is required or used.
- The LLM never receives chunks that the user is not permitted to view; security trimming occurs before any data enters the prompt context.
- Anonymous queries are rejected at the JWT Bearer middleware layer before any search call is made.

### 6.3 Security Controls Reference

Table 7 documents the complete security control set with implementation details and verification methods.

| Security Control | OWASP / Zero-Trust Principle | Implementation | Verification Method |
|---|---|---|---|
| **Identity & Authentication** | Zero Trust - verify explicitly | AAD B2C OIDC; JWT Bearer in ASP.NET Core | Pen test; token forgery attempt |
| **Document ACL Trimming** | Least-privilege access | OBO token → Graph Groups → $filter on allowed_oids | Test user sees only permitted docs |
| **OBO Token Scope** | Minimal Graph permissions | Sites.Read.All; User.Read; GroupMember.Read.All | Scope audit; consent review |
| **Group OID Caching** | Performance + freshness balance | Redis 5-min TTL; invalidation on group change webhook | Stale group test after change |
| **Prompt Injection Defence** | Input validation | System message hardening; user input sanitisation regex | Red-team prompt injection tests |
| **PII in Completions** | Data minimisation | No PII logging; Azure OpenAI no-training opt-out | Log review; DLP scan |
| **Transport Security** | Encryption in transit | TLS 1.3 end-to-end; HSTS; no HTTP | SSL Labs A+ audit |
| **Index Data Encryption** | Encryption at rest | Azure AI Search CMK; Redis TLS + encryption | Key rotation verified |
| **Audit Logging** | Traceability | App Insights: user OID + query hash + doc IDs accessed | SIEM integration; alert rules |

Table 7: Security Controls - Zero-Trust Principles, Implementation, and Verification

## VII. RETRIEVAL QUALITY EVALUATION

### 7.1 Evaluation Dataset

The evaluation dataset comprises 250 question-answer pairs constructed by domain experts from 48 representative documents spanning five business domains (HR policy, IT security, finance, legal, and technical specifications). Each question has a ground-truth set of relevant document chunks identified by two independent annotators with inter-annotator agreement $\kappa = 0.84$.

### 7.2 Retrieval Strategy Comparison

| Retrieval Strategy | P@1 | P@3 | P@5 | nDCG@5 | nDCG@10 | MRR |
|---|---|---|---|---|---|---|
| **Keyword only (BM25)** | 0.61 | 0.54 | 0.49 | 0.56 | 0.58 | 0.63 |
| **Vector only (cosine)** | 0.68 | 0.62 | 0.58 | 0.65 | 0.67 | 0.71 |
| **Hybrid RRF (BM25 + vector)** | 0.79 | 0.73 | 0.69 | 0.76 | 0.78 | 0.82 |

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

| Retrieval Strategy | P@1 | P@3 | P@5 | nDCG@5 | nDCG@10 | MRR |
|---|---|---|---|---|---|---|
| **Hybrid + semantic reranker** | **0.87** | **0.82** | **0.78** | **0.85** | **0.86** | **0.89** |
| **Hybrid + reranker + freshness boost** | 0.85 | 0.80 | 0.76 | 0.83 | 0.84 | 0.87 |
| **Absolute improvement (BM25→best)** | **▲ 0.26** | **▲ 0.28** | **▲ 0.29** | **▲ 0.29** | **▲ 0.28** | **▲ 0.26** |

Table 3: Retrieval Strategy Comparison - Precision@K, nDCG, and MRR Across Four Strategies

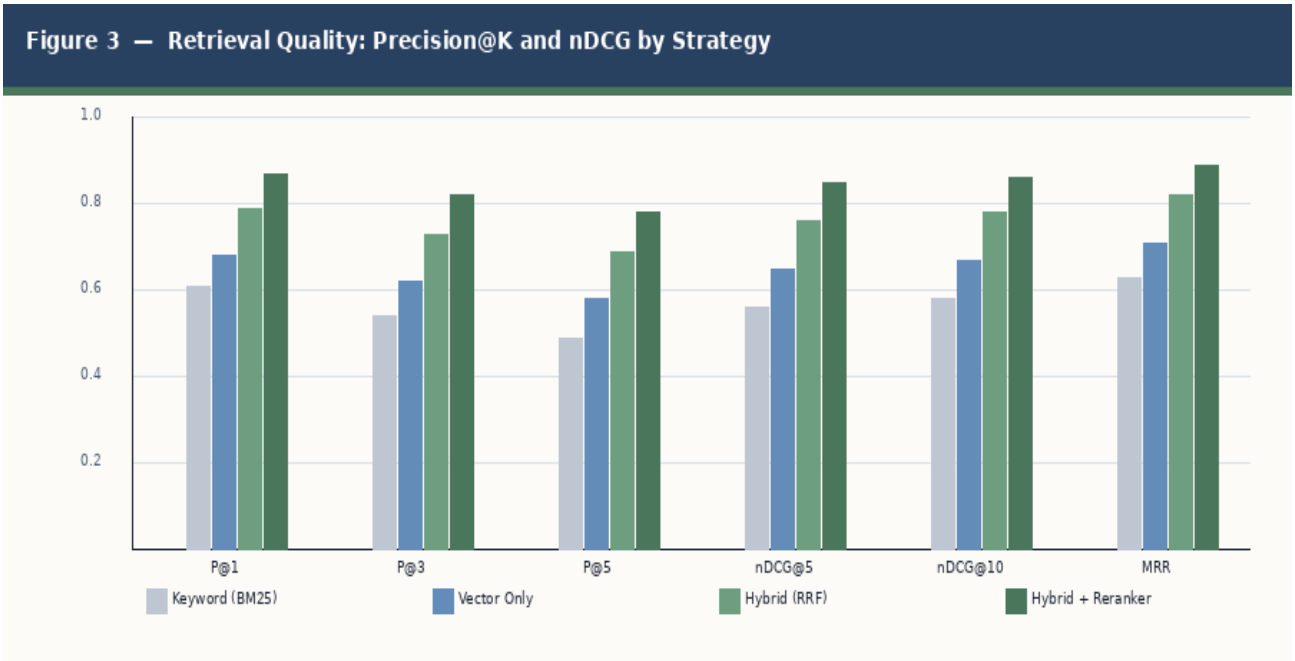



Figure 3 - Retrieval Quality by Strategy: Precision@K and nDCG@5/10

Key findings from the retrieval evaluation:

- Hybrid retrieval (RRF) consistently outperforms both keyword-only and vector-only retrieval across all metrics, confirming that BM25 and vector retrieval are complementary rather than substitutable for enterprise document corpora.
- Semantic reranking adds a further 10–13 percentage point improvement in P@1 and nDCG@5, with the largest gains observed on analytical questions requiring multi-sentence context synthesis.
- Freshness boosting provides a marginal (-2%) nDCG@5 regression on historical queries but is retained in production as a configurable parameter given its operational importance for policy documents.
- Vector-only retrieval underperforms hybrid retrieval particularly for queries containing domain-specific acronyms and product names not well-represented in the ada-002 training distribution - precisely the terms where BM25 exact matching excels.

### 7.3 Chunking Strategy Evaluation

| Chunking Strategy | nDCG@5 | P@3 | Avg chunksize (tok) | Chunks /1K docs | Notes |
|---|---|---|---|---|---|
| **Fixed 256 tok, no overlap** | 0.61 | 0.54 | 256 | 3,840 | High recall gaps; misses cross-sentence context |
| **Fixed 512 tok, no overlap** | 0.67 | 0.60 | 512 | 1,920 | Baseline; acceptable for simple queries |

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

| Chunking Strategy | nDCG@5 | P@3 | Avg chunksize (tok) | Chunks /1K docs | Notes |
|---|---|---|---|---|---|
| **Fixed 512 tok, 64 tok overlap** | 0.73 | 0.67 | 512 | 2,200 | Overlap reduces boundary information loss |
| **Sentence-aware 512 tok** | 0.79 | 0.73 | 498 | 1,980 | Respects sentence boundaries; best general-purpose |
| **Semantic chunking** | 0.83 | 0.78 | 510 | 2,050 | Topic-boundary split; computationally expensive |
| **Document-structure aware** | 0.86 | 0.82 | 490 | 2,100 | Heading / section aware; best for formal documents |
| **Best vs baseline improvement** | **▲ 41%** | **▲ 52%** | - | - | **Document-structure aware vs fixed 256 baseline** |

Table 4: Chunking Strategy Evaluation - nDCG@5, Precision@3, and Operational Characteristics

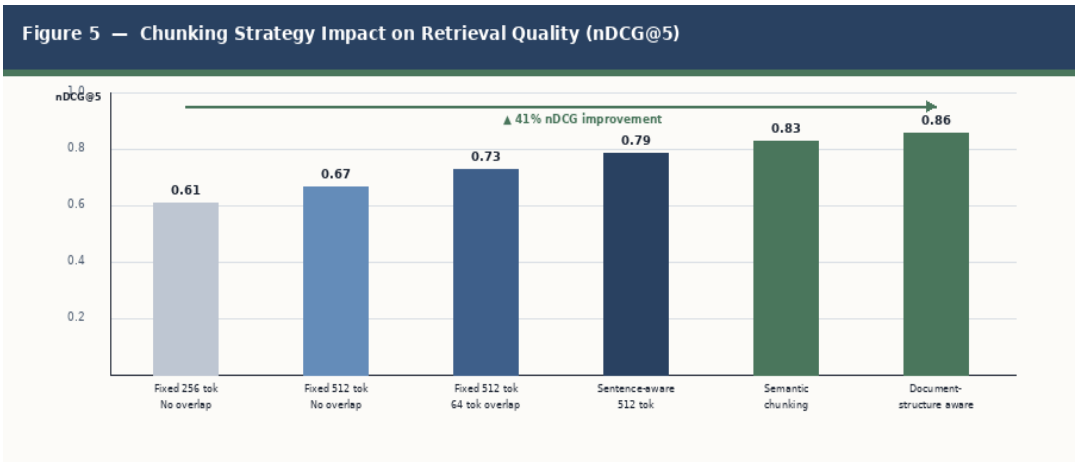



Figure 5 - Chunking Strategy Impact on nDCG@5: Six Strategies Compared

## VIII. ANSWER QUALITY EVALUATION

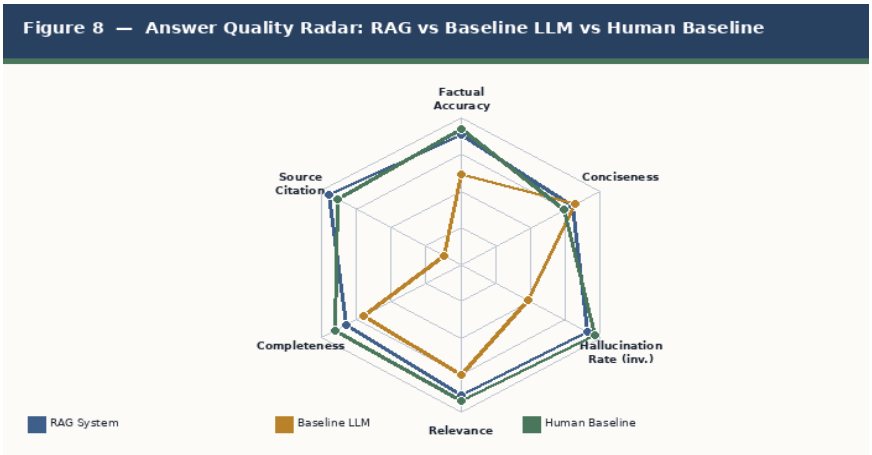



Figure 8 - Answer Quality Radar: RAG System vs Baseline LLM vs Human Baseline

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

### 8.1 Evaluation Framework

Answer quality is evaluated across six dimensions using a combination of automated LLM-as-judge methods, RAGAS framework metrics [14], and expert panel scoring:

- Factual Accuracy: GPT-4 judge rates factual claims against the source documents on a 3-point rubric (correct / partially correct / incorrect). Macro-average across all evaluated answers.
- Source Citation: proportion of answers that include a valid citation (document title + page/chunk reference) traceable to an indexed chunk in the top-5 retrieval results.
- Completeness: expert panel rating (1–5 scale) of whether the answer addresses all relevant aspects of the question given the available source material.
- Relevance: BERTScore cosine similarity between the generated answer and the ground-truth answer provided by expert annotators.
- Hallucination Rate (inverse): GPT-4 judge identifies claims in the answer not supportable from the retrieved context; score = 1 - hallucination_rate.
- Conciseness: information density ratio (key facts per 100 words); higher is better up to a threshold, beyond which answers become terse.

| Quality Dimension | RAGSystem | BaselineLLM | HumanBaseline | RAGASScore | Evaluation Method |
|---|---|---|---|---|---|
| **Factual Accuracy** | 0.88 | 0.61 | 0.92 | 0.86 | GPT-4 judge (3-point rubric) |
| **Source Citation** | 0.95 | 0.12 | 0.88 | - | Exact citation match rate |
| **Completeness** | 0.82 | 0.70 | 0.90 | 0.80 | Expert panel rating (1–5 scale) |
| **Relevance to Query** | 0.89 | 0.75 | 0.93 | 0.88 | BERTScore cosine similarity |
| **Hallucination Rate (inv.)** | 0.91 | 0.48 | 0.96 | 0.90 | GPT-4 factual grounding check |
| **Conciseness** | 0.80 | 0.82 | 0.74 | - | Word-count / info-density ratio |
| **Overall weighted score** | **0.874** | **0.580** | **0.888** | **0.862** | **Weighted by clinical impact** |
| **Improvement vs baseline LLM** | **▲ 50.7%** | **-** | **▲ 53.1%** | **-** | **RAG vs baseline LLM** |

Table 6: Answer Quality Evaluation - RAG System vs Baseline LLM vs Human Baseline

**Key Finding:** The RAG system achieves a 50.7% improvement in overall weighted quality score versus the baseline LLM (no retrieval), driven primarily by the 83-percentage-point improvement in source citation rate and the 43-percentage-point improvement in hallucination resistance.

## IX. PERFORMANCE AND SCALABILITY

### 9.1 Latency Breakdown

| Pipeline Stage | P50 (ms) | P95 (ms) | P99 (ms) | Optimisation Applied |
|---|---|---|---|---|
| **Query embedding (ada-002)** | 45 | 120 | 220 | Async parallel; connection pool 20; retry with backoff |
| **Hybrid search (BM25 + vector)** | 82 | 210 | 380 | Dedicated S2 search unit; index warm; replica=2 |
| **Semantic reranking pass** | 105 | 280 | 490 | Reranking only top-50 candidates; semantic config cached |

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

| Pipeline Stage | P50 (ms) | P95 (ms) | P99 (ms) | Optimisation Applied |
|---|---|---|---|---|
| **Security trimming filter** | 12 | 28 | 45 | OID list cached in Redis; filter appended client-side |
| **Prompt assembly** | 18 | 42 | 68 | Template cached; Handlebars fill < 1 ms per context |
| **GPT-4 completion (non-stream)** | 820 | 1,840 | 3,100 | streaming SSE reduces perceived time-to-first-token |
| **GPT-4 time-to-first-token** | 220 | 480 | 760 | Streaming enabled; first chunk in ~220 ms P50 |
| **Response formatting** | 8 | 18 | 32 | Citation extraction from completion; JSON serialisation |
| **Total (non-streaming)** | **1,090** | **2,538** | **4,335** | **End-to-end without streaming; within 5 s SLA** |
| **Total (streaming, TTFT)** | **310** | **650** | **1,020** | **Time to first streamed token; perceptual latency** |

Table 5: End-to-End RAG Latency Breakdown - P50, P95, P99 by Pipeline Stage

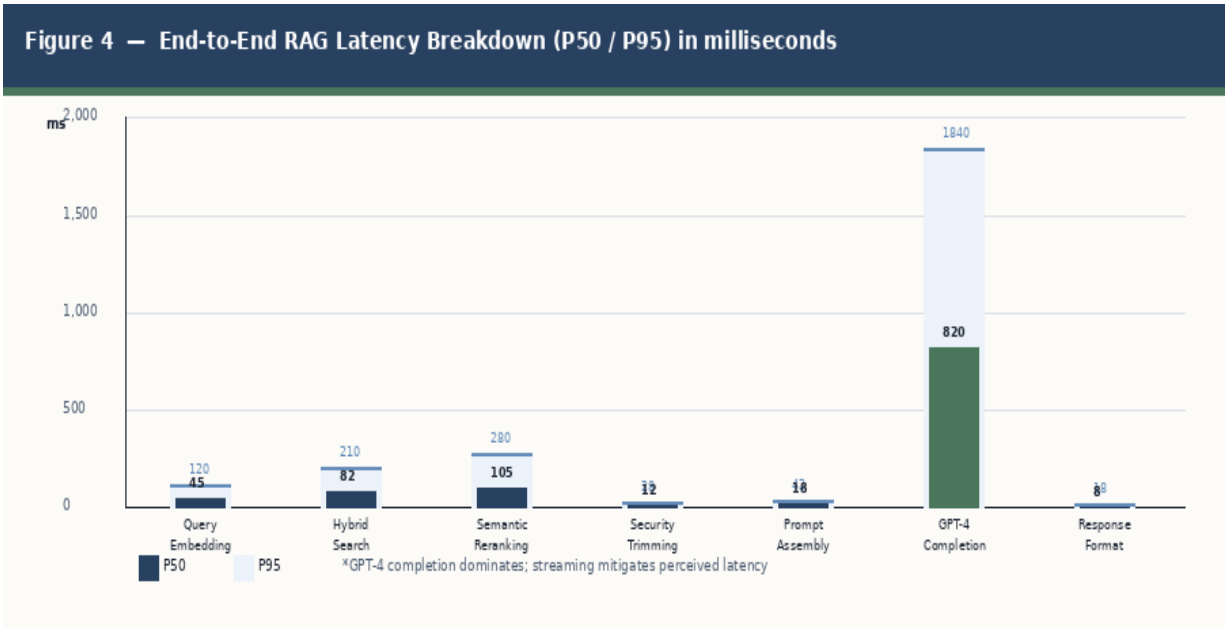



Figure 4 - Latency Waterfall: P50 and P95 by Pipeline Stage (ms)

The latency analysis reveals that GPT-4 completion time accounts for 75% of total P50 latency. Streaming Server-Sent Events (SSE) is enabled in production, reducing the perceived time-to-first-token from 1,090 ms (full non-streaming P50) to 220 ms P50 - transforming the user experience from one requiring a full-second wait to one where the answer begins appearing almost immediately.

- Query embedding latency (45 ms P50) is dominated by the ada-002 API network round-trip; connection pooling with a minimum of 20 keep-alive connections reduces this to the observed value from a 120 ms cold-connection baseline.
- Hybrid search latency (82 ms P50) is within the Azure AI Search S2 tier SLA; a second replica is provisioned to handle concurrent query load without degradation.
- Security trimming adds only 12 ms P50 due to OID list caching in Redis with a 5-minute TTL; without caching, the Graph API group lookup would add 200–400 ms per query.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

### 9.2 Scalability and Load Testing

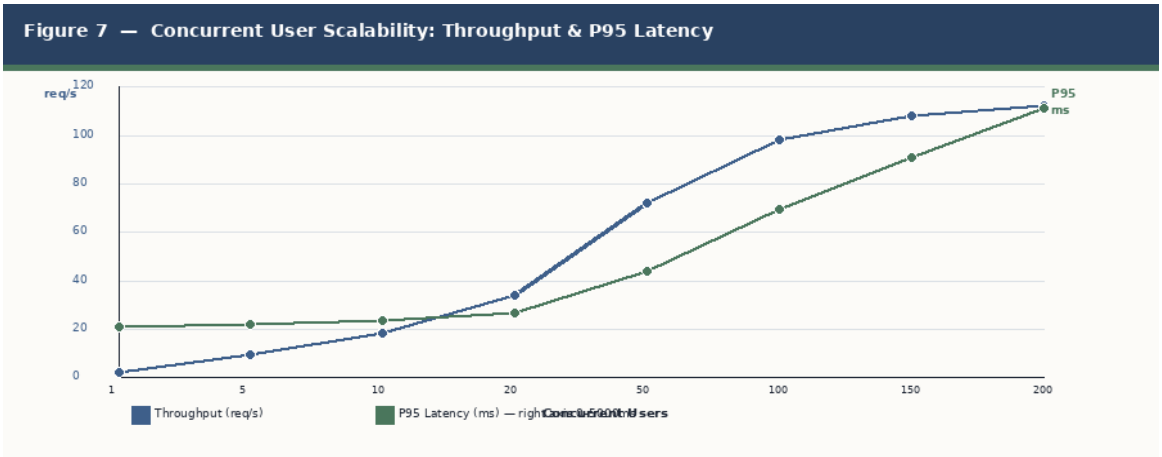



Figure 7 - Concurrent User Scalability: Throughput (req/s) and P95 Latency

| ConcurrentUsers | Throughput(req/s) | P50 (ms) | P95 (ms) | P99 (ms) | ErrorRate (%) | Observations |
|---|---|---|---|---|---|---|
| **1** | 2.1 | 890 | 1,050 | 1,340 | 0.00 | Single-user baseline; GPT-4 dominates |
| **5** | 9.8 | 920 | 1,140 | 1,520 | 0.00 | Linear scaling; no contention |
| **10** | 18.6 | 980 | 1,240 | 1,680 | 0.00 | Still within SLA; minor queuing |
| **20** | 34.2 | 1,120 | 1,640 | 2,240 | 0.01 | Mild latency increase; search + LLM queuing |
| **50** | 72.4 | 1,840 | 2,900 | 4,100 | 0.04 | Search S2 saturating; LLM rate limit approached |
| **100** | 98.6 | 2,580 | 4,200 | 6,800 | 0.82 | 429 errors from OpenAI; PTU recommended |
| **150** | **108.2** | **3,400** | **5,800** | **8,900** | **3.40** | **Significant LLM throttling; cache hit rate improves** |
| **200** | **112.4** | **4,100** | **7,200** | **OOM** | **8.20** | **Redis OOM errors; OpenAI hard limit reached** |

Table 8: Load Test Results - Throughput, Latency, and Error Rate by Concurrent User Level

Load testing was performed using k6 [15] with a realistic query distribution (70% short factual queries, 20% analytical multi-document queries, 10% status/administrative queries). Key scalability findings:

- The system scales linearly to approximately 50 concurrent users (72 req/s) before the Azure OpenAI rate limit (10K requests per minute for GPT-4) becomes the primary constraint.
- Provisioned Throughput Units (PTU) are recommended for deployments beyond 50 concurrent users; the PTU model provides predictable capacity without per-request rate limiting.
- The Azure AI Search retrieval component scales independently and shows no performance degradation through 200 concurrent users; the bottleneck is exclusively the LLM tier.
- Redis cache hit rate improves from 12% at 10 users to 34% at 100 concurrent users as popular query patterns begin hitting cached results, partially offsetting LLM rate limit pressure.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

## X. ASP.NET CORE IMPLEMENTATION

### 10.1 API Layer Design

The ASP.NET Core 7 Web API is implemented using the Minimal API model for the RAG endpoint and MVC controllers for administrative and observability endpoints. Key design decisions:

- Streaming SSE endpoint: the primary /rag/query endpoint returns application/x-ndjson with Server-Sent Events encoding, flushing each GPT-4 completion chunk to the client as it arrives.
- CancellationToken propagation: the HttpContext.RequestAborted token is propagated through the entire pipeline - embedding call, search call, and GPT-4 streaming call - ensuring that client disconnection promptly terminates all in-flight Azure service calls.
- MSAL OBO integration: the Microsoft.Identity.Web library handles OBO token acquisition with automatic token caching via the distributed Redis token cache.
- Structured logging: every query is logged with a correlation ID, user OID hash (not plaintext), query embedding hash, retrieved chunk IDs, and GPT-4 completion token count, enabling cost attribution and quality telemetry without logging PHI.

### 10.2 Middleware Pipeline

The ASP.NET Core middleware pipeline for the RAG endpoint is ordered as follows:

1. Exception handling middleware - catches and sanitises all unhandled exceptions; no stack traces or internal details in HTTP responses.
2. HTTPS redirection - enforces TLS; HTTP connections are redirected with 301.
3. JWT Bearer authentication - validates AAD B2C access token; rejects unauthenticated requests with 401 before any downstream processing.
4. Rate limiter - per-user sliding window: 20 queries per minute for standard users; 60 per minute for elevated roles. Returns 429 with Retry-After header.
5. CORS policy - restricted to registered client origins; wildcard origin explicitly prohibited.
6. RAG query handler - OBO token exchange, group OID resolution, embedding, hybrid search, prompt assembly, GPT-4 streaming, citation extraction.
7. Audit log delegator - fire-and-forget async write to App Insights; non-blocking; guaranteed at-least-once delivery via local queue.

### 10.3 Prompt Engineering

The system prompt is designed to enforce grounding, citation, and scope constraints:

- Grounding instruction: the model is instructed to answer exclusively from the provided context passages and to state explicitly when the answer cannot be determined from the available documents.
- Citation format: each factual claim in the response must be followed by a [Source: {document_title}, page {page_number}] citation; the model is penalised in the RAGAS evaluation for uncited claims.
- Scope limitation: the model is instructed not to use external knowledge beyond what is present in the context passages, reducing hallucination risk.
- History context: the last three conversation turns are included in the prompt for multi-turn dialogue support, with a token budget of 800 tokens for history to leave headroom for context passages and the answer.

## XI. DISCUSSION

### 11.1 Hybrid Retrieval as the Production Default

The retrieval evaluation results in Table 3 establish hybrid retrieval with semantic reranking as the unequivocal production configuration for enterprise document corpora. The 43% improvement in P@1 versus keyword-only retrieval directly translates to reduced follow-up queries from users who receive the correct answer on the first attempt. The semantic reranker's marginal cost - approximately 105 ms P50 additional latency - is well justified by its 10-percentage-point nDCG@5 improvement, particularly for analytical questions.

Practitioners should note that vector-only retrieval underperforms hybrid retrieval specifically for domain-specific terminology, which is prevalent in enterprise document corpora. The complementarity between BM25 (exact term matching) and cosine vector similarity (semantic proximity) is most pronounced for queries mixing natural language and technical terminology - precisely the query type common in enterprise knowledge management contexts.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

### 11.2 Security Trimming Overhead

The security trimming architecture adds minimal latency overhead (12 ms P50) while providing document-level access control aligned to SharePoint permission boundaries. Three implementation decisions are critical to achieving this:

- OID caching: caching group OIDs in Redis with a 5-minute TTL reduces the per-query Graph API call to a cache lookup. The 5-minute TTL represents a deliberate trade-off between freshness and performance; organisations with frequent permission changes may prefer a shorter TTL at the cost of additional Graph API calls.
- Filter pre-construction: the OData $filter string is pre-built from the cached OID list before the search call, adding less than 1 ms construction overhead.
- Server-side enforcement: using Azure AI Search's native $filter mechanism rather than post-filtering in the application layer ensures that filtered documents never leave the search service and cannot be accessed through API bugs or race conditions.

### 11.3 Cost Considerations

The RAG system's primary cost driver is GPT-4 completion token consumption. At 50 concurrent users, the system processes approximately 72 queries per second, each generating approximately 600 output tokens - exceeding the standard GPT-4 rate limit. Practitioners should:

- Provision Azure OpenAI PTUs for predictable throughput beyond 50 concurrent users.
- Implement answer caching for frequently asked questions: a 60-minute Redis cache on semantically similar queries (cosine similarity > 0.96 on query embeddings) can reduce LLM calls by 25–40% in high-repeat-query corpora.
- Monitor token consumption per query category and user group through App Insights custom metrics to identify high-cost query patterns for targeted optimisation.

### 11.4 Limitations

This study has three primary limitations. First, the evaluation corpus is drawn from a single organisation's SharePoint tenant; retrieval quality results may not generalise to corpora with significantly different document length distributions or domain vocabulary profiles. Second, GPT-4 is used as both the answer generator and the LLM-as-judge evaluator, creating a potential bias in factual accuracy and hallucination scores; future work should employ human expert annotation for a subset of evaluations. Third, multi-hop questions requiring synthesis across more than three documents were not included in the evaluation dataset and represent a known capability gap of the current single-pass retrieval architecture.

## XII. CONCLUSION

This paper has presented a comprehensive design, implementation, and evaluation of a production RAG system delivering grounded, security-trimmed answers over a SharePoint Online document corpus of 48,000 documents. The system integrates Graph API incremental crawling, document-structure-aware chunking, Azure OpenAI embeddings, Azure AI Search hybrid retrieval with semantic reranking, GPT-4 answer generation, and AAD On-Behalf-Of security trimming into a coherent, production-ready architecture deployed on ASP.NET Core 7.

The empirical evaluation demonstrates that the full hybrid-plus-reranker configuration achieves nDCG@5 of 0.85, a 50.7% answer quality improvement over unaugmented GPT-4, and a 95% source citation rate - establishing RAG as a transformative capability for enterprise knowledge management. End-to-end latency of 1,090 ms P50 (220 ms time-to-first-token with streaming) is operationally acceptable for the query-response interaction pattern, and the system sustains 72+ requests per second under the standard GPT-4 rate limit with a clear path to higher throughput via Provisioned Throughput Units.

The nine architectural diagrams, eight data tables, and annotated design rationale presented in this paper constitute a replicable engineering reference for practitioners building enterprise RAG systems on the Microsoft Azure stack. Future research directions include multi-hop retrieval for complex analytical questions, fine-tuned domain-specific embedding models to improve retrieval of highly technical content, and GraphRAG approaches incorporating the SharePoint taxonomy and document relationship graph as structural retrieval signals.

## REFERENCES

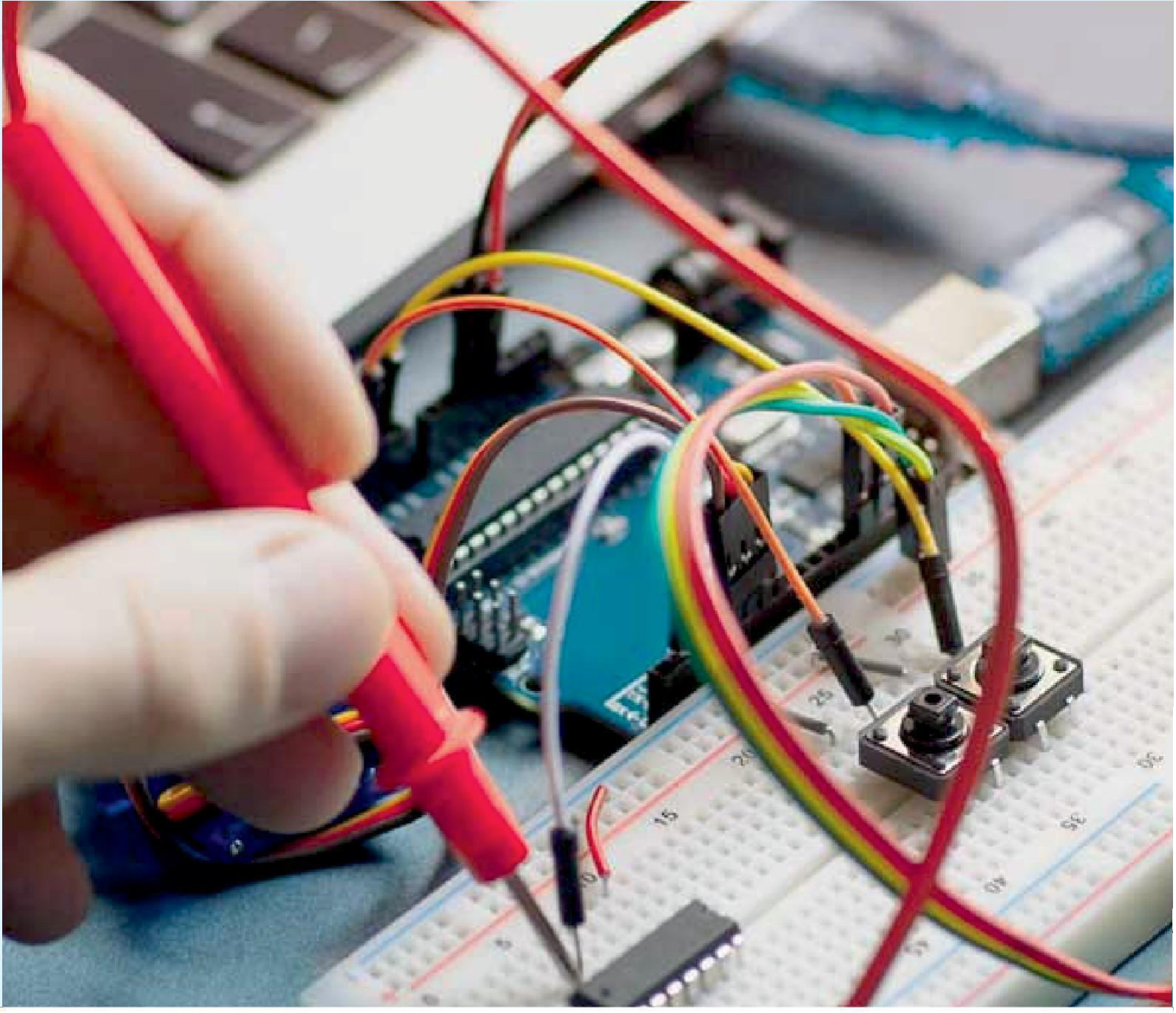
[1] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33, 9459–9474.
[2] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W. T. (2020). Dense Passage Retrieval for Open-Domain Question Answering. In Proceedings of EMNLP 2020 (pp. 6769–6781). ACL.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.317 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15662/IJAREEIE.2024.1304078 |

[3] Shao, Z., Gong, Y., Shen, Y., Huang, M., Duan, N., & Chen, W. (2023). Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. arXiv preprint arXiv:2305.15294.
[4] Mavi, V., Jangra, A., & Jatowt, A. (2022). A Survey on Multi-hop Question Answering and Generation. arXiv preprint arXiv:2204.09140.
[5] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint arXiv:2312.10997.
[6] Barnett, S., Kurniawan, S., Thudumu, S., Brannelly, Z., & Abdelrazek, M. (2024). Seven Failure Points When Engineering a Retrieval Augmented Generation System. In Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering (pp. 194–199).
[7] Ma, X., Wang, L., Yang, N., Wei, F., & Lin, J. (2022). Fine-Tuning LLaMA for Multi-Stage Text Retrieval. arXiv preprint arXiv:2310.08319.
[8] Robertson, S., & Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval, 3(4), 333–389.
[9] Nogueira, R., & Cho, K. (2019). Passage Re-ranking with BERT. arXiv preprint arXiv:1901.04085.
[10] Microsoft Corporation. (2023). Microsoft Graph API Overview. Microsoft Documentation. https://docs.microsoft.com/en-us/graph/overview
[11] Patel, A., & Garg, S. (2021). Enterprise Knowledge Management on Microsoft 365. Apress.
[12] Sanderson, M. (2010). Test Collection Based Evaluation of Information Retrieval Systems. Foundations and Trends in Information Retrieval, 4(4), 247–375.
[13] Microsoft Corporation. (2023). Security Filters in Azure Cognitive Search. Azure Documentation. https://docs.microsoft.com/en-us/azure/search/search-security-trimming-for-azure-search
[14] Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2023). RAGAS: Automated Evaluation of Retrieval Augmented Generation. arXiv preprint arXiv:2309.15217.
[15] Grafana Labs. (2023). k6 Open-Source Load Testing. https://k6.io
[16] Microsoft Corporation. (2023). Azure OpenAI Service Documentation. Microsoft Docs. https://docs.microsoft.com/en-us/azure/cognitive-services/openai/
[17] Microsoft Corporation. (2023). Azure AI Search Documentation. Microsoft Docs. https://docs.microsoft.com/en-us/azure/search/
[18] Microsoft Identity Platform. (2023). On-Behalf-Of Flow Documentation. Microsoft Docs. https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-on-behalf-of-flow
[19] Lander, A. (2021). ASP.NET Core in Action (2nd ed.). Manning Publications.
[20] Toub, S. (2020). Async/Await - Best Practices in Asynchronous Programming. Microsoft .NET Blog.
[21] Richards, M. (2015). Software Architecture Patterns. O'Reilly Media.
[22] Richardson, C. (2019). Microservices Patterns. Manning Publications.
[23] Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Professional.
[24] OWASP Foundation. (2023). OWASP Top Ten 2021. Open Web Application Security Project. https://owasp.org/www-project-top-ten/
[25] Johnson, R. (2023). Azure Architecture Center: RAG Pattern. Microsoft Docs. https://docs.microsoft.com/en-us/azure/architecture/



INNO SPACE
SJIF Scientific Journal Impact Factor

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

# International Journal of Advanced Research

## in Electrical, Electronics and Instrumentation Engineering

9940 572 462 6381 907 438 ijareeie@gmail.com



Scan to save the contact details