# **Product Requirements Document: Internal Generative AI Knowledge Assistant (Codename: "Keystone")**

## **1. Product Vision and Strategic Context**

### **1.1. Introduction and Problem Statement**

Within the modern enterprise, knowledge is the most critical asset, yet it is often the most fragmented. Key information resides in disparate silos: technical documentation in Confluence, project plans on SharePoint, policy documents on network drives, and institutional knowledge scattered across internal wikis and slide decks. This fragmentation imposes a significant and continuous tax on productivity. Employees spend valuable time searching for information, often unsuccessfully, leading to duplicated work, inconsistent decision-making, and a reliance on outdated or incomplete data. The inability to efficiently access and synthesize proprietary knowledge represents a fundamental operational bottleneck, hindering agility and innovation.

"Keystone" is conceived to solve this problem. It is an enterprise-grade Generative AI application designed to serve as a secure, intelligent, and centralized conversational interface to the organization's entire body of internal knowledge.

### **1.2. Product Vision and Mission**

The vision for Keystone is to create a single, authoritative source of truth that is accessible to every employee through a natural, conversational interface. It will transform how information is discovered and consumed within the enterprise.

The mission of Keystone is to empower every employee to make faster, better-informed decisions by providing instant, accurate, and context-aware answers that are verifiably grounded in the company's own proprietary data. Keystone will reduce information retrieval friction, accelerate workflows, and unlock the collective intelligence of the organization.

### **1.3. Core Architectural Principle: Retrieval-Augmented Generation (RAG)**

The foundational architecture for Keystone is Retrieval-Augmented Generation (RAG). This is not merely a technical choice but a strategic commitment to factual accuracy, security, and relevance—qualities that are non-negotiable in an enterprise context.44 General-purpose Large Language Models (LLMs), while powerful, suffer from two critical flaws for enterprise use: their knowledge is limited to their last training date, rendering them ignorant of current, internal information, and they are prone to "hallucinations," where they generate plausible but factually incorrect information.1 These limitations make them unsuitable for applications where accuracy is paramount.

The RAG framework directly mitigates these risks by augmenting the LLM with an information retrieval system that operates on a trusted, external knowledge base—in this case, the company's internal document corpus.1 By grounding the LLM's responses in specific, verifiable documents, RAG transforms the model from a probabilistic text generator into a sophisticated knowledge synthesis engine. This approach ensures that all generated answers are based on up-to-date, authoritative internal data, building user trust and providing a level of auditability required for business-critical applications.5

The Keystone RAG process will adhere to three distinct steps 6:

1. **Retrieve:** When a user poses a query, the system first searches a specialized vector database (Redis) containing numerical representations (embeddings) of the company's documents. It retrieves the most semantically relevant text chunks related to the query.
2. **Augment:** The retrieved text chunks are then seamlessly incorporated into a new, augmented prompt that is sent to the LLM. This prompt includes the original user query along with the retrieved context.
3. **Generate:** The LLM is instructed to generate a response based *exclusively* on the provided context. This crucial step constrains the model, forcing it to synthesize an answer from the supplied facts rather than relying on its internal, generalized knowledge. The system will also provide citations linking the response back to the source documents, ensuring full transparency and verifiability.7

This strategic decision to use RAG frames Keystone not as a tool for creating new, unverified knowledge, but as a highly efficient system for retrieving, understanding, and synthesizing existing, authoritative enterprise knowledge. This distinction is fundamental to managing stakeholder expectations and delivering a reliable and trustworthy product.

## **2. User Personas and Journeys**

To ensure Keystone meets the diverse needs of the organization and incorporates security from the ground up, the system is designed around distinct user personas. These personas directly inform functional requirements, the user interface design, and, most critically, the Role-Based Access Control (RBAC) model.

### **2.1. User Personas**

The following table defines the primary user personas who will interact with the Keystone system. Differentiating these roles is essential for designing appropriate feature sets and, crucially, for implementing the granular access controls necessary to protect sensitive information. An engineer should not have access to sensitive HR documents, and a sales executive should not be able to query confidential financial projections. The persona-based design ensures these boundaries are enforced at an architectural level.

| Persona Title | Role Description | Primary Goals | Key User Stories |
| --- | --- | --- | --- |
| **Knowledge Consumer** | Any employee (e.g., Sales Executive, Project Manager, Software Engineer, HR Generalist) who needs to find information to perform their job. This is the largest user group. | - Find accurate answers to specific questions quickly.  - Understand complex documents without reading them in their entirety.  - Verify information by tracing it back to the source. | - "What is our company policy on remote work?"  - "Summarize the Q3 financial results from the latest earnings call transcript."  - "Provide a code example for authenticating to the internal 'Orion' API." |
| **Data Curator** | A subject matter expert or department head (e.g., Head of HR, Director of Engineering, Legal Counsel) responsible for the integrity and scope of the knowledge within their domain. | - Ensure their team's documents are ingested and up-to-date in Keystone.  - Manage the lifecycle of documents (add, update, remove).  - Define access permissions for sensitive documents within their domain. | - "Upload the new 2025 Employee Handbook PDF and make it searchable."  - "Restrict access to the 'Project Phoenix' M&A documents to only the Legal and Executive teams."  - "Remove the outdated Q1 marketing campaign brief." |
| **System Administrator** | A member of the IT or Operations team responsible for the overall health, security, and maintenance of the Keystone application. | - Manage user accounts and role assignments.  - Monitor system performance, latency, and costs.  - Configure system-level settings and security policies.  - Troubleshoot technical issues. | - "Onboard the new marketing team by creating user accounts and assigning them the 'Knowledge Consumer' role."  - "View the dashboard to check for spikes in query latency or API errors."  - "Configure the connection to a new document source repository." |

### **2.2. User Stories**

The following user stories translate the goals of each persona into specific, actionable requirements for the Keystone system.

#### **Knowledge Consumer Stories**

* As a **Sales Executive**, I want to ask, "What is our standard discount policy for enterprise clients with over 500 licenses?" so that I can provide accurate and approved pricing information during a client negotiation.
* As a **Software Engineer**, I want to ask, "What are the required environment variables for running the 'Payments' microservice in a local development environment?" so that I can set up my workstation without having to search through multiple README files in different repositories.
* As a **Project Manager**, I want to ask, "Summarize the key risks and mitigation strategies identified in the 'Project Titan' post-mortem document" and receive a bulleted list with direct citations to the specific page and section of the source PDF.
* As an **HR Generalist**, I want to ask, "What are the steps for processing a parental leave request for an employee in California?" to ensure I follow the correct, compliant procedure.

#### **Data Curator Stories**

* As an **HR Manager**, I want to upload the new, company-wide 2025 employee handbook PDF and ensure it is immediately indexed and available for all employee queries, replacing the previous version.
* As a **Legal Counsel**, I want to ingest a folder of sensitive M&A due diligence documents and tag them with the legal\_team\_only access role, so that only users with this role can retrieve information from them.
* As a **Director of Engineering**, I want to connect Keystone to our department's Confluence space so that all technical design documents and architectural decision records are automatically kept in sync and searchable.

#### **System Administrator Stories**

* As a **System Administrator**, I want to view a real-time dashboard showing the number of active users, total queries per hour, average end-to-end response latency, and total token consumption to monitor system health and operational costs.
* As a **System Administrator**, I want to receive an alert if the P99 query latency exceeds 3 seconds, so I can proactively investigate performance bottlenecks.
* As a **System Administrator**, I want to access a secure interface to create, modify, and deactivate user accounts and assign them to predefined roles (e.g., knowledge\_consumer, data\_curator\_hr, admin).

## **3. Functional Requirements and Features**

This section details the core functionalities of the Keystone system, from how data enters the system to how users interact with it.

### **3.1. Data Ingestion and Processing Pipeline**

The ingestion pipeline is the foundation of Keystone's knowledge base. It must be robust, automated, and capable of handling various enterprise data formats.

* **Document Source Connectors:** The system must provide out-of-the-box connectors for ingesting documents from primary enterprise repositories. Initial support will include:
  + Microsoft SharePoint Online
  + Atlassian Confluence Cloud
  + Local and Network File Systems (via a secure upload interface)
* **Document Parsing and Text Extraction:** Keystone will support a range of common enterprise file formats, including PDF, Microsoft Word (.docx), Microsoft PowerPoint (.pptx), Markdown (.md), and plain text (.txt). The system will employ robust parsing libraries (e.g., pypdf, python-docx) to accurately extract text content.7 The architecture must be designed to handle documents with complex layouts, such as PDFs containing tables, charts, and embedded images, which is a known challenge in RAG systems.10 Future iterations will explore advanced parsing techniques for these complex elements.
* **Text Chunking Strategy:** Raw text must be segmented into smaller, semantically coherent chunks before being converted into vector embeddings. This process is critical for retrieval accuracy. Keystone will implement a recursive character text splitting strategy that prioritizes semantic boundaries (e.g., paragraphs, sections, headings) to ensure that context is not arbitrarily broken mid-sentence. The chunk size will be a configurable parameter, with an initial recommended size of **200-300 tokens**. This range offers a strong balance between providing enough context for the LLM and maintaining the precision needed for accurate retrieval.11
* **Embedding and Storage Workflow:** The ingestion process will be managed by an asynchronous pipeline to ensure responsiveness and scalability. The workflow is as follows:
  1. A new or updated document is detected in a connected source or uploaded manually.
  2. The document is passed to the appropriate parser for text extraction.
  3. The extracted text is segmented into chunks based on the defined strategy.
  4. Each text chunk is sent to the selected embedding model to be converted into a vector embedding.
  5. The resulting vector, along with the original text content and critical metadata (e.g., source\_document\_id, chunk\_id, source\_uri, and allowed\_roles), is written as a single record to the Redis vector database.11
* **Document Lifecycle and Versioning Governance:** To ensure the knowledge base remains current and trustworthy, a robust document lifecycle management strategy is required.44
  + **Change Detection:** The system will implement mechanisms to detect document changes (creations, updates, deletions) in near real-time. This will be achieved through tailored approaches for each source, such as webhook triggers for Confluence, Change Data Capture (CDC) for databases, or file system watchers for network drives.44
  + **Document Versioning:** Each ingested document and its corresponding chunks will be associated with a version identifier. This can be implemented using content hashing (e.g., SHA-256) to detect meaningful changes, preventing unnecessary re-indexing for superficial updates like metadata modifications.46 This versioning aligns vector embeddings with their specific source data version, ensuring reproducibility and data lineage.45
  + **Deprecation and Archival Policy:** Data Curators will have the ability to define retention policies. A semi-automated process will flag documents that have not been updated or accessed past a certain threshold. Curators will be notified to review these documents for archival or deletion, preventing the knowledge base from becoming polluted with stale information.

### **3.2. Retrieval and Augmentation Engine**

This is the core "retrieval" component of the RAG architecture, responsible for finding the right information to answer a user's query.

* **Query Understanding and Intent Detection:** Before retrieval, the system must robustly process user queries, which are often noisy, ambiguous, or contain multiple intents.47 An LLM-assisted query understanding module will be implemented as a pre-retrieval step.47 This module will:
  + **Denoise and Normalize:** Correct spelling errors and normalize terminology.
  + **Detect Intent:** Classify the user's intent (e.g., factual lookup, summarization, comparison) to tailor the retrieval strategy.48 For example, a factual query ("What is our Q3 revenue?") requires high-precision retrieval, while a summarization query ("Summarize the new HR policy") may benefit from broader context.
  + **Decompose Complex Queries:** Break down multi-hop or ambiguous questions ("How did our healthcare costs change last year compared to the new policy's impact?") into structured sub-queries that can be executed independently.49
* **Semantic Search and Retrieval:** When a user submits a query, it is first converted into a vector embedding using the same model that processed the source documents. The system then executes a k-Nearest Neighbor (k-NN) vector similarity search against the Redis index to find the top-k most relevant document chunks.12
* **Metadata-Based Filtering for Access Control:** This is the cornerstone of Keystone's security model. Every user query will be executed within the context of that user's assigned roles. The vector search query sent to Redis will include a mandatory metadata filter. This filter ensures that the search operation *only* considers document chunks where the allowed\_roles tag contains one of the user's roles.14 This enforces access control at the earliest possible stage—the point of retrieval—preventing unauthorized data from ever reaching the LLM.
* **Reranking for Relevance (V1.1 Enhancement):** While the initial vector search provides a set of relevant candidates, their ranking can be further optimized. The architecture will be designed to support a subsequent reranking step. After the initial top-k chunks are retrieved, a more computationally intensive but more accurate reranker model (such as a cross-encoder) can be used to re-evaluate and re-order these chunks to ensure the most relevant information is placed at the top.15 This will be targeted for a V1.1 release, with considerations for caching reranked results for common queries to optimize latency.51
* **Prompt Construction:** The system will utilize a carefully engineered prompt template to structure the information sent to the LLM. This template will combine the user's original query with the retrieved and reranked context. Crucially, the prompt will contain explicit instructions for the LLM, such as: "You are a helpful AI assistant for [Company Name]. Answer the user's question based *only* on the provided context. Do not use any prior knowledge. If the context does not contain the answer, state that you cannot answer the question. For each piece of information you provide, you must cite the source document".3

### **3.3. Generative Chat Interface and User Experience (UX)**

The user-facing interface must be intuitive, responsive, and build trust through transparency.

* **Query Input and Processing:** A clean, minimalist chat interface will allow users to easily type and submit their questions.7 The interface will support multi-line input and maintain a clear conversational history.
* **Response Generation and Streaming:** To provide a highly responsive user experience, the LLM's generated answer will be streamed back to the UI on a token-by-token basis. This allows the user to begin reading the response almost instantly, rather than waiting for the entire generation process to complete.
* **Source Citation and Traceability:** This feature is fundamental to user trust and the system's value proposition. Each key statement or piece of data in the generated response must be clearly linked back to the source document(s) and chunk(s) from which it was derived. The UI will display these citations (e.g., as numbered footnotes or interactive links) that, when clicked, reveal the source document name and a snippet of the original text, with an option to open the full source document.4
* **Conversation History and Session Management:** Keystone will support multi-turn conversations, allowing users to ask follow-up questions that build on the previous context. To facilitate this, Redis will be used not only as a vector database but also as a high-performance LLM session manager. It will store the history of each user's conversation, enabling the system to provide this context to the LLM for more natural and coherent interactions.6
* **User Feedback Mechanism:** Each generated response will be accompanied by a simple feedback mechanism, such as thumbs-up/thumbs-down icons and an optional text field for comments. This user feedback is an invaluable source of data for evaluating the system's performance and identifying areas for improvement in both the retrieval and generation components.3

### **3.4. Administration and Management Console**

A secure, web-based administration console will provide Data Curators and System Administrators with the tools needed to manage and maintain the Keystone system.

* **Document Corpus Management:** An interface for authorized Data Curators and Administrators to manage the knowledge base. This will include functionalities to:
  + Add new document sources (e.g., connect a new SharePoint site).
  + Manually upload, update, and delete individual documents or folders.
  + Trigger a re-indexing process for a specific source or the entire corpus.
  + View the indexing status and logs for all documents.
* **User Management and Role Assignment:** A dedicated section for System Administrators to manage the user base. This will include the ability to:
  + Create, view, edit, and deactivate user accounts.
  + Assign and un-assign users to the predefined roles (as defined in Section 2.1).
* **System Analytics Dashboard:** A dashboard providing a high-level overview of system health and usage. It will display key metrics in near real-time, including:
  + Query volume (total and per user).
  + Average and P99 response latency.
  + LLM and embedding model token consumption.
  + Aggregated user feedback scores (e.g., percentage of positive ratings).20

## **4. System Architecture and Technical Specifications**

This section outlines the technical blueprint for Keystone, detailing the interplay between its components and the specific technologies that will be employed.

### **4.1. High-Level Architecture**

The Keystone system is designed as a modern, scalable, and modular web application. The architecture consists of several distinct but interconnected components that work in concert to deliver the RAG functionality.

The end-to-end data flow is as follows: A user interacts with the **Frontend**, which sends requests to the **Backend API**. The API, built with **FastAPI**, authenticates the user and passes the query to the **Orchestrator**. The Orchestrator manages the RAG pipeline: it vectorizes the query, retrieves context from **Redis**, constructs the augmented prompt, calls the **LLM Service**, and streams the response back to the user. Redis serves as the central data hub for the entire application.

The primary components are:

1. **Frontend:** A single-page application (SPA), likely built with a modern JavaScript framework like React, providing the user-facing chat and administration interfaces.
2. **Backend API:** A high-performance web server built with **FastAPI**. FastAPI is chosen for its exceptional speed, native support for asynchronous operations (critical for handling I/O-bound tasks like API calls to LLMs), and automatic data validation through Pydantic models.7
3. **Orchestrator:** The logical core of the application. This layer, which can be implemented using a framework like **LangChain** or as a custom module, orchestrates the entire RAG workflow.11 It is responsible for the sequence of retrieving, augmenting, and generating.
4. **Redis:** The high-performance, in-memory data store that serves as the heart of the application's real-time operations.
5. **LLM Service:** The endpoint for the selected Large Language Model (e.g., a commercial API like OpenAI or a self-hosted model endpoint).
6. **Embedding Service:** The endpoint for the selected text embedding model.

A key architectural decision is to leverage Redis not merely as a vector database but as a unified **AI Application State Hub**. Production-grade conversational AI applications require three distinct real-time data functions: fast vector retrieval for context, a cache to reduce costs and latency on repeated queries, and a session store for managing conversational history. Instead of deploying and maintaining three separate systems (e.g., a dedicated vector DB, a Memcached/Redis cache, and a PostgreSQL DB for chat logs), this architecture consolidates these functions into a single, optimized Redis instance. This approach simplifies the technology stack, reduces operational overhead, minimizes points of failure, and leverages Redis's core strength—low-latency data access—across the entire application state, from retrieval to conversation management.6

### **4.2. Model Selection and Strategy**

The choice of AI models is the most significant factor influencing Keystone's performance, cost, and security posture. The selection process must be deliberate and data-driven.

#### **4.2.1. Embedding Model Selection**

The quality of the embedding model directly determines the quality of the retrieval step; a poor embedding model will fail to find relevant context, rendering the entire RAG pipeline ineffective.22 The selection will be based on a trade-off between retrieval accuracy, cost, and operational complexity.

The evaluation process will begin with a review of the **MTEB (Massive Text Embedding Benchmark) leaderboard**, with a specific focus on the "Retrieval" task scores.15 However, public benchmarks on general-purpose datasets may not perfectly predict performance on our specific, proprietary corporate documents. Therefore, the final decision will be validated by a "bake-off" evaluation using a representative sample of our own data.25

The following table compares potential candidates for the embedding model:

| Model Name | Provider/Type | MTEB Retrieval Score (Avg.) | Dimensionality | Max Context (Tokens) | Cost (per 1M tokens) / Hosting | Recommendation Rationale |
| --- | --- | --- | --- | --- | --- | --- |
| text-embedding-3-small | OpenAI (API) | ~61.0 | 1536 | 8192 | $0.02 | **Initial MVP Choice.** Excellent balance of performance, very low cost, and zero operational overhead. Ideal for rapid prototyping and initial deployment.22 |
| embed-english-v3.0 | Cohere (API) | ~64.5 | 1024 | 512 | ~$1.00 | High performance on MTEB, but a smaller context window and higher cost make it less ideal for our general use case compared to alternatives.22 |
| bge-large-en-v1.5 | BAAI (Open Source) | ~64.0 | 1024 | 512 | Self-hosted (GPU cost) | **Long-Term Strategy Candidate.** Top-tier open-source performance. Self-hosting provides maximum data security (no data leaves our network) and predictable costs at scale, but requires infrastructure management.24 |
| e5-large-v2 | Microsoft (Open Source) | ~63.5 | 1024 | 512 | Self-hosted (GPU cost) | Strong alternative to BGE. The choice between top open-source models will depend on the results of our internal bake-off evaluation.25 |

**Initial Recommendation:** The project will commence using the **text-embedding-3-small** model via its API. This approach minimizes initial setup complexity and allows the team to focus on building the core application logic. In parallel, an evaluation will be conducted to compare its performance and total cost of ownership against a self-hosted **bge-large-en-v1.5** model. The long-term goal is to migrate to a self-hosted model to maximize security and control costs.

#### **4.2.2. Large Language Model (LLM) Selection**

The choice between a commercial, API-based LLM and a self-hosted, open-source LLM is a pivotal strategic decision. It involves fundamental trade-offs between cutting-edge performance, data security, long-term cost, and customization potential. For an enterprise application handling proprietary and potentially sensitive internal data, these trade-offs must be carefully weighed.

The table below analyzes this strategic choice:

| Factor | Commercial Models (e.g., GPT-4-Turbo, Claude 3 Opus) | Open-Source Models (e.g., Llama-3-70B, Mixtral 8x7B) | Analysis & Recommendation |
| --- | --- | --- | --- |
| **Performance & Capabilities** | Often lead in general reasoning, instruction following, and complex language nuances. Considered the state-of-the-art.28 | Performance is rapidly catching up and, for many tasks, is comparable to commercial counterparts. May require more prompt engineering to achieve similar results.29 | While commercial models may have a slight edge, the performance of top-tier open-source models is more than sufficient for a grounded RAG task, where the primary skill is synthesis, not unconstrained generation. |
| **Data Security & Privacy** | Data (prompts and context) is sent to a third-party provider. While providers offer security guarantees, this introduces risk of data leaks, unauthorized access, or data being used for model training. Log leaks are a known vulnerability.30 | When self-hosted on private infrastructure, **no data ever leaves the corporate network**. This provides the maximum possible level of data security and privacy, eliminating third-party risk.29 | **This is the deciding factor for an enterprise.** The ability to guarantee that proprietary data remains within the company's security perimeter makes self-hosted open-source the superior choice for mitigating risk. |
| **Cost** | Billed on a per-token basis (input and output). At scale, this can become a significant and unpredictable operational expense.31 | Involves a fixed, upfront cost for hosting infrastructure (GPUs). At scale, the total cost of ownership is significantly lower and more predictable than per-token API fees.31 | Open-source offers a far more sustainable and predictable cost model for a high-usage internal application. |
| **Customization & Control** | Limited customization options, typically restricted to what the provider's API allows. No access to the underlying model weights.32 | Full access to model weights allows for deep customization and fine-tuning on domain-specific data to improve performance on specialized tasks.31 | Open-source provides complete control over the model's behavior and future development, preventing vendor lock-in. |
| **Support & Reliability** | Come with enterprise-level support, SLAs, and high availability managed by the provider.28 | Reliability and uptime are the responsibility of the internal operations team. Requires in-house expertise to manage and maintain the hosting infrastructure. | This is the primary trade-off. The organization must be willing to invest in the operational expertise required to manage a self-hosted model, potentially using deployment baselines like Open WebUI or vLLM.52 |

**Recommendation:** For an internal enterprise tool like Keystone, where data security, cost control, and customization are paramount, the long-term strategy must be to use a **self-hosted, open-source LLM (e.g., Llama-3-70B)**. This approach aligns with a security-first posture and provides the best long-term economic value. For the initial MVP phase, a commercial API (e.g., GPT-4-Turbo) may be used to accelerate development, with a clear migration path to the self-hosted solution.

### **4.3. Redis Data and Indexing Architecture**

Redis will be configured to serve its multiple roles efficiently and securely. The **RedisVL** library will be used for all interactions, as it provides a high-level, developer-friendly abstraction for vector database operations.12

* **Redis Index Schema:** The index schema defines the structure of the data to be stored and searched. It will be defined declaratively in a YAML file to ensure it is version-controlled and easily auditable.12  
  YAML  
  index:  
   name: "keystone-idx"  
   prefix: "doc\_chunk"  
   storage\_type: "json"  
  fields:  
   - name: content  
   type: text  
   attrs:  
   sortable: false  
   - name: doc\_id  
   type: tag  
   - name: source\_uri  
   type: text  
   - name: allowed\_roles  
   type: tag # Critical field for enforcing Role-Based Access Control  
   - name: embedding  
   type: vector  
   attrs:  
   algorithm: hnsw  
   dims: 1536 # Must match the dimensionality of the chosen embedding model  
   distance\_metric: cosine  
   datatype: float32
* **Vector Indexing Strategy:** The **HNSW (Hierarchical Navigable Small World)** algorithm will be used for the vector index. HNSW is the industry standard for high-performance approximate nearest neighbor (ANN) search on large datasets, offering an excellent trade-off between search speed, accuracy (recall), and memory usage.11 Key HNSW parameters, such as  
  M (number of connections per node) and EF\_CONSTRUCTION (size of the dynamic list for candidate neighbors during indexing), will be tuned during performance testing to optimize for our specific dataset and latency requirements.11
* **Data Models in Redis:**
  + **Document Chunks:** Stored as **RedisJSON** documents, following the schema defined above. The key will follow a consistent pattern, e.g., doc\_chunk:{uuid}. Using JSON allows for flexible metadata and efficient filtering.33
  + **User Sessions:** Conversation history will be stored using **Redis Streams** or **Redis Lists**. The key will be session:{session\_id}. This allows for efficient appending of new messages and retrieval of recent conversation history to provide context for follow-up questions.
  + **Semantic Cache:** Implemented using **Redis Hashes**. The key will be a hash of a normalized user query, and the value will be the cached LLM response. A TTL (Time-To-Live) will be set on these keys to ensure the cache is periodically refreshed.

### **4.4. API Specification (FastAPI)**

The backend will expose a secure, well-documented RESTful API. All request and response bodies will be strictly validated using Pydantic models to ensure data integrity and prevent common errors.17

The following table outlines the key API endpoints for the Keystone system:

| Endpoint | HTTP Method | Request Body (Pydantic Model) | Response Body | Description |
| --- | --- | --- | --- | --- |
| /api/v1/chat | POST | QueryInput(question: str, session\_id: Optional[str]) | Streaming JSON with content and citations | Submits a user query. The core endpoint for the RAG functionality. Returns a streamed response with generated text and source citations. |
| /api/v1/documents | POST | UploadFile(...) (multipart/form-data) with metadata | DocumentInfo(id: str, filename: str, status: str) | Uploads a new document for ingestion and indexing. This is an asynchronous operation. |
| /api/v1/documents/{doc\_id} | DELETE | None | {"status": "deleted"} | Deletes a document and all its associated chunks from the Redis index. Restricted to Data Curators and Admins. |
| /api/v1/feedback | POST | FeedbackInput(response\_id: str, rating: bool, comment: Optional[str]) | {"status": "received"} | Allows users to submit feedback on the quality of a specific generated response. |
| /api/v1/admin/users | POST | UserCreate(email: str, roles: List[str]) | UserInfo(...) | (Admin only) Creates a new user and assigns roles. |
| /api/v1/admin/users/{user\_id} | GET | None | UserInfo(...) | (Admin only) Retrieves details for a specific user. |
| /api/v1/admin/analytics | GET | None | AnalyticsDashboardData(...) | (Admin only) Retrieves data for the system analytics dashboard. |

## **5. Non-Functional Requirements (NFRs)**

These requirements define the system's operational standards and quality attributes, which are as crucial as its features for enterprise adoption.

### **5.1. Performance and Latency**

A responsive user experience is critical for user adoption. The system must meet the following latency targets under the specified load conditions:

* **P99 Vector Retrieval Latency:** The time taken for Redis to perform a top-10 vector similarity search must be **less than 50 milliseconds**.11 This ensures the retrieval step is not a bottleneck.
* **End-to-End Response Latency (First Token):** The time from when a user submits a query to when the first token of the response appears in the UI must be **less than 500 milliseconds**. This provides the perception of an instantaneous response.19
* **End-to-End Response Latency (Total):** The total time to generate a typical response (approx. 150 words) should be **less than 2 seconds**.11

### **5.2. Scalability and Reliability**

The system must be built to grow with the organization's needs and be resilient to failures.

* **Scalability:** The architecture must be designed to support an initial load of **500 concurrent users** and a knowledge base of at least **1 million document chunks**. The backend services will be containerized (e.g., using Docker) and deployed on a platform that supports horizontal scaling. The Redis deployment will be configured as a **Redis Cluster** to allow for horizontal sharding of data, ensuring that performance does not degrade as the dataset grows.11
* **Reliability:** The system will target a service uptime of **99.9%**. This will be achieved through redundant deployments of all components and automated health checks.

### **5.3. Security and Compliance**

Security is the most critical non-functional requirement for an enterprise system handling proprietary data. The architectural design is fundamentally shaped by the "RAG Security Paradox": the act of centralizing data to ground LLM responses also creates a new, high-value target for data exfiltration. Unlike source systems (like SharePoint or an HR platform) which have their own robust, built-in access controls, the RAG system's vector database does not inherit these permissions.30 Therefore, a security model cannot be an afterthought; it must be built into the core of the application. The guiding principle is to enforce deterministic authorization at the application layer, preventing the LLM from ever being in a position to make a security decision.36

The following security requirements are mandatory:

* **Role-Based Access Control (RBAC):** Access to information will be strictly enforced at the data retrieval layer. As specified in sections 3.2 and 4.3, every document chunk in Redis will be tagged with allowed\_roles metadata. Every query to Redis will include a filter that matches the user's roles against these tags. This ensures that a user can *never* retrieve a chunk of data they are not authorized to see, effectively implementing access control at the database level.14 This will be enforced not only at the application layer but also by leveraging Redis Access Control Lists (ACLs) where applicable to provide defense-in-depth.53
* **Data Encryption:** All data will be encrypted at all times.
  + **In-Transit:** All communication between system components (frontend, backend, Redis, LLMs) will be encrypted using TLS 1.3.37
  + **At-Rest:** All data stored in Redis and any persistent storage will be encrypted using industry-standard algorithms like AES-256.37
* **Input Sanitization and Prompt Security:** The API gateway and backend will rigorously sanitize all user-provided input to mitigate the risk of **Prompt Injection** attacks. This prevents malicious users from crafting inputs designed to override the system's instructions, jailbreak the LLM, or cause it to reveal sensitive information from its context.36
* **Secrets Management:** All sensitive credentials, such as API keys for LLM services, database passwords, and signing keys, will be stored in a dedicated secrets management solution (e.g., HashiCorp Vault, AWS Secrets Manager). They will never be hardcoded in the application source code or configuration files.11
* **Comprehensive Audit Logging:** The system will maintain immutable, detailed audit logs for all security-sensitive events. This includes every user query, the generated response, all administrative actions (e.g., user creation, role changes), and all data ingestion events. These logs are essential for security monitoring, incident response, and forensic analysis.35
* **Privacy-Preserving Logging and Data Handling:** While comprehensive logging is essential for audits, it must not compromise user privacy. The system will incorporate a privacy-preserving mode that can be enabled based on compliance requirements.54 This involves:
  + **PII Redaction:** Before logging, all user queries and LLM responses will be passed through a PII detection and redaction service. Sensitive information (names, emails, SSNs, etc.) will be automatically identified and either masked (e.g., AC43\*\*\*\*) or replaced with a non-reversible hash.54 This ensures that logs can be used for debugging and analysis without storing sensitive personal data.
  + **Configurable Anonymization:** System Administrators will have the ability to configure the level of redaction to comply with regulations like GDPR and CCPA.54
* **Data Minimization and Compliance:** The principle of least privilege will be applied to data. The system will only ingest and store data that is explicitly configured by Data Curators. This reduces the overall risk surface. The system's data handling practices will be designed to comply with relevant data protection regulations such as GDPR, by enabling data access, rectification, and deletion requests through the administrative console.30

### **5.4. Maintainability and Monitoring**

The system must be designed for ease of maintenance and proactive monitoring.

* **Structured Logging:** All application components will produce structured logs (e.g., in JSON format) containing clear, contextual information to facilitate debugging and analysis.
* **Monitoring and Alerting:** A comprehensive monitoring solution (e.g., using Prometheus and Grafana) and adhering to industry standards like OpenTelemetry for comprehensive observability will be implemented to track key system and business metrics.21 This includes:
  + **System Metrics:** CPU/memory usage, error rates, API latency.
  + **Business Metrics:** Query volume, token consumption, cache hit ratio.
  + **Alerting:** Automated alerts will be configured to notify the operations team of critical events, such as performance degradation (latency spikes), high error rates, or security anomalies detected in the audit logs.

### **5.5. Known Limitations and Mitigations (V1.0)**

While the V1.0 of Keystone is designed to be a robust and valuable tool, it is important to acknowledge its initial limitations. This transparency helps manage stakeholder expectations and guides the future roadmap.

* **Focus on Unstructured Text:** The initial version will excel at processing text-based documents (PDFs,.docx, etc.). It will not support querying structured data from SQL databases or other tabular sources. Mitigation: The roadmap includes Text-to-SQL capabilities for future releases.
* **Limited Handling of Complex Embedded Objects:** While text from tables and images will be extracted, the semantic relationship within complex charts or diagrams may be lost. Mitigation: The roadmap includes multi-modal RAG enhancements to better interpret these objects.
* **Basic User Feedback Analytics:** The UI will collect user feedback (thumbs up/down), but the admin console will only show aggregated scores. It will not provide deep analytics on which types of queries are failing. Mitigation: A more advanced feedback analysis module is planned for V1.1 to help curators identify knowledge gaps.
* **No Per-User History Summary:** Users can see their conversation history, but the system will not provide automated summaries or analysis of a user's query patterns.

## **6. Evaluation Framework and Success Metrics**

The success of Keystone cannot be determined by a single metric. A system that is factually accurate but too slow will fail. A system that is fast but hallucinates will fail. A system that is accurate and fast but prohibitively expensive will also fail. Therefore, a comprehensive evaluation framework is required, built upon what can be termed the "Evaluation Triad": a balanced scorecard that tracks Retrieval Quality, Response Quality, and System Performance as co-equal pillars of success. A decline in any one of these areas indicates a problem that must be addressed, ensuring that optimizations in one dimension do not inadvertently degrade another.16

### **6.1. Methodology**

Our evaluation will be rigorous, automated, and repeatable.

* **Golden Dataset:** A "golden" test set will be curated and maintained. This dataset will consist of a representative sample of questions covering various topics and complexities from our internal documents. Each question will be paired with a hand-verified "golden" answer and a list of the specific source document chunks required to answer it correctly.20 This dataset will be supplemented by synthetic user queries generated from anonymized production logs to expand test coverage.48
* **Evaluation Framework:** We will leverage an open-source evaluation framework like **RAGAs** (Retrieval-Augmented Generation Assessment). RAGAs provides a suite of metrics to programmatically measure key aspects of the RAG pipeline, such as faithfulness and context relevance, allowing for rapid, objective, and consistent evaluation.20

### **6.2. Key Success Metrics and KPIs**

The following table defines the specific metrics that will be tracked for each dimension of the Evaluation Triad, along with their target values for the V1 release. This provides a clear, quantitative definition of success for the project.

| Dimension | Metric | Description | Target (V1) | Source |
| --- | --- | --- | --- | --- |
| **Retrieval Quality** | Context Precision@K | Of the top K chunks retrieved, what percentage are relevant to the user's query? | > 0.90 | 20 |
|  | Context Recall@K | Of all the relevant chunks that *should* have been retrieved, what percentage were actually in the top K results? | > 0.85 | 41 |
|  | NDCG@10 | Normalized Discounted Cumulative Gain. A measure of ranking quality. It rewards retrieving relevant documents and penalizes placing them lower in the results. | > 0.80 | 20 |
| **Response Quality** | Faithfulness / Groundedness | What percentage of the generated answer is directly supported by the provided context? Measures the absence of hallucinations. | > 95% | 16 |
|  | Answer Relevance | How relevant is the generated answer to the user's original question? | > 90% | 16 |
|  | Answer Correctness | When compared to the "golden" answer, is the generated answer factually correct? | > 90% | 41 |
| **System Performance & Cost** | P99 End-to-End Latency | The 99th percentile latency for a query from submission to full response generation. | < 2.5 seconds | 11 |
|  | Average Cost Per Query | The average cost of LLM and embedding API calls per user query. | < $0.05 | 22 |
|  | Semantic Cache Hit Ratio | The percentage of queries that are served directly from the semantic cache, avoiding a full LLM call. | > 30% | 19 |

## **7. Future Roadmap**

While the V1 release will deliver a powerful and secure RAG system, Keystone is envisioned as a platform that will evolve to meet more advanced enterprise needs.

### **7.1. V1.1 Enhancements (Short-Term)**

* **Advanced Reranking:** Integrate a sophisticated cross-encoder reranking model into the pipeline to further improve the precision of retrieved context, especially for nuanced or ambiguous queries.
* **Semantic Caching:** Fully implement and optimize the semantic cache layer in Redis. This will significantly reduce latency and LLM API costs for frequently asked or semantically similar questions.6
* **Expanded Data Sources (Text-to-SQL):** Develop connectors for structured data sources, such as internal PostgreSQL or MySQL databases. This will involve integrating a Text-to-SQL model that can translate a user's natural language query into a SQL query, execute it, and use the results as context for the LLM, dramatically expanding the scope of knowledge Keystone can access.38

### **7.2. V2.0 Vision (Long-Term)**

* **Agentic Workflows:** Evolve Keystone from a question-answering system into an "agent" that can perform actions on behalf of the user. This could include tasks like, "Find the latest project status report for 'Project Apollo' and email a summary to the project stakeholders," or "File a new IT support ticket based on this error description".35 This requires careful implementation of security guardrails and permissions.
* **Multi-Modal RAG:** Extend the system's capabilities to understand and retrieve information from non-textual data. This includes parsing and searching for information within images, diagrams, charts, and tables embedded in documents, providing a much richer understanding of the source material.1
* **Proactive Information Delivery:** Develop features that can anticipate user needs and proactively push relevant information. For example, based on a user's calendar, Keystone could proactively surface the meeting agenda and relevant background documents just before a meeting starts.
* **Prompt Management and Registry System:** As the system's use cases expand, a centralized registry for managing, versioning, and testing prompt templates will become essential.57 This system will allow authorized administrators to create, edit, and A/B test different prompt strategies to optimize for specific tasks (e.g., summarization vs. factual extraction) without requiring code changes.57
* **Integration with Collaboration Tools (Extensibility):** To embed Keystone deeper into existing workflows, the system will provide APIs and webhooks for integration with primary collaboration platforms.59 This includes:
  + **Slack/Microsoft Teams Bot:** Develop a conversational bot that allows users to query Keystone directly from their chat client, bringing knowledge to where conversations happen.60
  + **Webhooks for Automation:** Provide outbound webhooks that can trigger actions in other systems based on AI-generated output, enabling more complex, automated workflows.

## **8. Adoption and Change Management Strategy**

A technically excellent tool can still fail if users do not adopt it. Therefore, a people-first adoption strategy is as critical as the technical architecture itself.62 The goal is to ensure employees understand the value of Keystone, feel confident using it, and successfully integrate it into their daily workflows. We will use the ADKAR model (Awareness, Desire, Knowledge, Ability, Reinforcement) as a framework for this strategy.62

### **8.1. Awareness: Communicating the "Why"**

* **Executive Sponsorship:** Secure active endorsement from leadership, who will communicate how Keystone aligns with the company's strategic goals for productivity and innovation.63
* **Clear Messaging:** Develop a clear and consistent internal communication plan that explains the problem of knowledge fragmentation and how Keystone solves it. Use multiple channels (emails, team meetings, internal newsletters) to build awareness before launch.63

### **8.2. Desire: Creating a Pull for the Tool**

* **Start with the Problem, Not the Tool:** Conduct workshops with different teams (e.g., Sales, HR, Engineering) to identify their specific pain points in information retrieval. Frame Keystone as a solution to *their* problems, not just another mandated tool.65
* **Identify Internal Champions:** Recruit a group of "power users" and "early adopters" from various departments. These individuals will pilot the tool, provide early feedback, and share success stories with their peers, creating social proof and reducing skepticism.64

### **8.3. Knowledge: Providing the "How"**

* **Role-Based Onboarding:** Develop structured, hands-on training materials tailored to different user personas. A sales executive's training will focus on different use cases than an engineer's.64
* **In-App Guidance:** Embed tutorials, tooltips, and guided workflows directly into the Keystone application using a digital adoption platform. This allows employees to learn by doing within their workflow, without disrupting productivity.64

### **8.4. Ability: Turning Knowledge into Practice**

* **Progressive Rollout:** Introduce Keystone gradually, starting with the internal champion teams before a company-wide launch. This allows the project team to gather feedback, fix issues, and build momentum.64
* **Accessible Support:** Establish clear support channels, including a dedicated help channel (e.g., in Slack/Teams) and regular "office hours" where users can ask questions and get help from the project team.

### **8.5. Reinforcement: Making Adoption Stick**

* **Feedback Loops:** Actively solicit and act upon user feedback. Create simple ways for users to report issues or suggest improvements. When employees see their feedback is valued and implemented, they become more invested in the tool's success.64
* **Monitor and Share Success:** Use the analytics dashboard to track adoption metrics (e.g., daily/monthly active users per department). Share success stories and metrics widely to demonstrate the tool's impact and reinforce its value.64

#### Works cited

1. What is Retrieval-Augmented Generation (RAG)? | Google Cloud, accessed July 11, 2025, <https://cloud.google.com/use-cases/retrieval-augmented-generation>
2. Retrieval Augmented Generation, accessed July 11, 2025, <https://genai.byu.edu/rag>
3. How to Prevent AI Hallucinations with Retrieval Augmented ..., accessed July 11, 2025, <https://www.itconvergence.com/blog/how-to-overcome-ai-hallucinations-using-retrieval-augmented-generation/>
4. Retrieval Augmented Generation (RAG) in Azure AI Search - Learn Microsoft, accessed July 11, 2025, <https://learn.microsoft.com/en-us/azure/search/retrieval-augmented-generation-overview>
5. What is RAG? - Retrieval-Augmented Generation AI Explained - AWS, accessed July 11, 2025, <https://aws.amazon.com/what-is/retrieval-augmented-generation/>
6. RAG with Redis | Docs, accessed July 11, 2025, <https://redis.io/docs/latest/develop/get-started/rag/>
7. Building a Retrieval-Augmented Generation (RAG) API and ..., accessed July 11, 2025, <https://dev.to/vivekyadav200988/building-a-retrieval-augmented-generation-rag-api-and-frontend-with-fastapi-and-react-native-2n7k>
8. RAG application with Azure OpenAI and Azure AI Search (FastAPI) - Azure App Service, accessed July 11, 2025, <https://learn.microsoft.com/en-us/azure/app-service/tutorial-ai-openai-search-python>
9. RAG from scratch with the Redis Vector Library - Colab, accessed July 11, 2025, <https://colab.research.google.com/github/redis-developer/redis-ai-resources/blob/main/python-recipes/RAG/01_redisvl.ipynb>
10. Top 7 Challenges with Retrieval-Augmented Generation - Valprovia, accessed July 11, 2025, <https://www.valprovia.com/en/blog/top-7-challenges-with-retrieval-augmented-generation>
11. Scalable RAG System with LangChain and Redis Vector Search - WeblineIndia, accessed July 11, 2025, <https://www.weblineindia.com/blog/build-rag-with-langchain-redis-vector-search/>
12. redis/redis-vl-python: Redis Vector Library (RedisVL) -- the ... - GitHub, accessed July 11, 2025, <https://github.com/redis/redis-vl-python>
13. Vector search concepts | Docs - Redis, accessed July 11, 2025, <https://redis.io/docs/latest/develop/ai/search-and-query/vectors/>
14. redis-ai-resources/python-recipes/RAG/07\_user\_role\_based\_rag.ipynb at main - GitHub, accessed July 11, 2025, <https://github.com/redis-developer/redis-ai-resources/blob/main/python-recipes/RAG/07_user_role_based_rag.ipynb>
15. Choosing the right embedding model for your RAG application: a comprehensive guide, accessed July 11, 2025, <https://unstructured.io/blog/understanding-embedding-models-make-an-informed-choice-for-your-rag>
16. RAG Evaluation Metrics Starter Kit - Arize AI, accessed July 11, 2025, <https://arize.com/blog-course/rag-evaluation/>
17. Create a RAG Chatbot with FastAPI & LangChain - FutureSmart AI Blog, accessed July 11, 2025, <https://blog.futuresmart.ai/building-a-production-ready-rag-chatbot-with-fastapi-and-langchain>
18. Building a RAG Pipeline from Scratch with RedisVL | Step-by-Step Tutorial - YouTube, accessed July 11, 2025, <https://www.youtube.com/watch?v=cCTKmmGO4CY>
19. Using Redis for real-time RAG goes beyond a Vector Database, accessed July 11, 2025, <https://redis.io/blog/using-redis-for-real-time-rag-goes-beyond-a-vector-database/>
20. Best Practices in RAG Evaluation: A Comprehensive Guide - Qdrant, accessed July 11, 2025, <https://qdrant.tech/blog/rag-evaluation-guide/>
21. This repository contains the code for building a Retrieval-Augmented Generation (RAG) system using LangChain and FastAPI. It includes document loading, text splitting, vector embedding, and API deployment for a scalable and efficient RAG-based application. - GitHub, accessed July 11, 2025, <https://github.com/anarojoecheburua/RAG-with-Langchain-and-FastAPI>
22. How to Choose the Right Embedding for Your RAG Model? - Analytics Vidhya, accessed July 11, 2025, <https://www.analyticsvidhya.com/blog/2025/03/embedding-for-rag-models/>
23. Step-by-Step Guide to Choosing the Best Embedding Model for Your Application | Weaviate, accessed July 11, 2025, <https://weaviate.io/blog/how-to-choose-an-embedding-model>
24. Top embedding models on the MTEB leaderboard | Modal Blog, accessed July 11, 2025, <https://modal.com/blog/mteb-leaderboard-article>
25. Choosing an Embedding Model - Pinecone, accessed July 11, 2025, <https://www.pinecone.io/learn/series/rag/embedding-models-rundown/>
26. How to Choose the Best Embedding Model for Your LLM Application | MongoDB, accessed July 11, 2025, <https://www.mongodb.com/developer/products/atlas/choose-embedding-model-rag/>
27. Use embedding models with Vertex AI RAG Engine - Google Cloud, accessed July 11, 2025, <https://cloud.google.com/vertex-ai/generative-ai/docs/rag-engine/use-embedding-models>
28. Open Source Vs. Proprietary LLMs: When to Use - Deepchecks, accessed July 11, 2025, <https://www.deepchecks.com/open-source-vs-proprietary-llms-when-to-use/>
29. [2407.13511] Can Open-Source LLMs Compete with Commercial Models? Exploring the Few-Shot Performance of Current GPT Models in Biomedical Tasks - arXiv, accessed July 11, 2025, <https://arxiv.org/abs/2407.13511>
30. Security Risks with RAG Architectures | IronCore Labs, accessed July 11, 2025, <https://ironcorelabs.com/security-risks-rag/>
31. Open-Source LLMs vs Closed: Unbiased Guide for Innovative Companies [2025], accessed July 11, 2025, <https://hatchworks.com/blog/gen-ai/open-source-vs-closed-llms-guide/>
32. Open Source vs. Commercial LLMs: Which for Your Enterprise ..., accessed July 11, 2025, <https://context-clue.com/blog/open-source-vs-commercial-llms/>
33. Redis as a vector database quick start guide | Docs, accessed July 11, 2025, <https://redis.io/docs/latest/develop/get-started/vector-database/>
34. Redis Vector Library (RedisVL) — RedisVL, accessed July 11, 2025, <https://docs.redisvl.com/>
35. ​​RAG is dead: why enterprises are shifting to agent-based AI architectures | TechRadar, accessed July 11, 2025, <https://www.techradar.com/pro/rag-is-dead-why-enterprises-are-shifting-to-agent-based-ai-architectures>
36. Is RAG a security risk? - Reddit, accessed July 11, 2025, <https://www.reddit.com/r/Rag/comments/1iunqcz/is_rag_a_security_risk/>
37. Securing your RAG application: A comprehensive guide - Pluralsight, accessed July 11, 2025, <https://www.pluralsight.com/resources/blog/ai-and-data/how-to-secure-rag-applications-AI>
38. RAG Security 101 - Protect AI, accessed July 11, 2025, <https://protectai.com/blog/rag-security-101>
39. Agentic RAG Data Security Risks and Mitigations - Piiano, accessed July 11, 2025, <https://www.piiano.com/blog/agentic-rag-data-security-risks-and-mitigations>
40. RAG Security: Risks and Mitigation Strategies, accessed July 11, 2025, <https://www.lasso.security/blog/rag-security>
41. Assess performance: Metrics that matter - Azure Databricks ..., accessed July 11, 2025, <https://learn.microsoft.com/en-us/azure/databricks/generative-ai/tutorials/ai-cookbook/evaluate-assess-performance>
42. RAG systems: Best practices to master evaluation for accurate and reliable AI. | Google Cloud Blog, accessed July 11, 2025, <https://cloud.google.com/blog/products/ai-machine-learning/optimizing-rag-retrieval>
43. RAG Evaluation: Don't let customers tell you first - Pinecone, accessed July 11, 2025, <https://www.pinecone.io/learn/series/vector-databases-in-production-for-busy-engineers/rag-evaluation/>
44. RAGOps: Operating and Managing Retrieval-Augmented Generation Pipelines - arXiv, accessed July 11, 2025, <https://arxiv.org/html/2506.03401v1>
45. Data Governance for Retrieval-Augmented Generation (RAG) - Enterprise Knowledge, accessed July 11, 2025, <https://enterprise-knowledge.com/data-governance-for-retrieval-augmented-generation-rag/>
46. Why Your Enterprise RAG System Needs Real-Time Vector Database Updates (And How to Build Them), accessed July 11, 2025, <https://ragaboutit.com/why-your-enterprise-rag-system-needs-real-time-vector-database-updates-and-how-to-build-them/>
47. [2506.21384] Leveraging LLM-Assisted Query Understanding for Live Retrieval-Augmented Generation - arXiv, accessed July 11, 2025, <https://arxiv.org/abs/2506.21384>
48. Advanced RAG for Search and Recommendations with personalization - GoPenAI, accessed July 11, 2025, <https://blog.gopenai.com/advanced-rag-for-search-and-recommendations-with-personalization-9b0b5e337ffc>
49. 5 RAG Query Patterns Every Engineering Leader Should Know - Nirant Kasliwal, accessed July 11, 2025, <https://nirantk.com/writing/rag-query-types/>
50. Query Expansion in Enhancing Retrieval-Augmented Generation (RAG) - Medium, accessed July 11, 2025, <https://medium.com/@sahin.samia/query-expansion-in-enhancing-retrieval-augmented-generation-rag-d41153317383>
51. Reranking Explained: Why It Matters for RAG Systems - Chatbase, accessed July 11, 2025, <https://www.chatbase.co/blog/reranking>
52. open-webui/open-webui: User-friendly AI Interface (Supports Ollama, OpenAI API, ...) - GitHub, accessed July 11, 2025, <https://github.com/open-webui/open-webui>
53. ACL | Docs - Redis, accessed July 11, 2025, <https://redis.io/docs/latest/operate/oss_and_stack/management/security/acl/>
54. Anonymize your PII data before sending it to an LLM - mstack, accessed July 11, 2025, <https://mstack.nl/blogs/anonymize-pii-llm/>
55. Building an LLM governance solution - PII redaction, audit logs, model blocking - looking for feedback : r/LLMDevs - Reddit, accessed July 11, 2025, <https://www.reddit.com/r/LLMDevs/comments/1ll6kcn/building_an_llm_governance_solution_pii_redaction/>
56. How to redact sensitive / PII data in your logs - OpenObserve, accessed July 11, 2025, <https://openobserve.ai/blog/redact-sensitive-data-in-logs/>
57. RAG prompt engineering makes LLMs super smart - K2view, accessed July 11, 2025, <https://www.k2view.com/blog/rag-prompt-engineering/>
58. Prompt Tuning For Building Enterprise Grade RAG Systems | by Abel Bekele | Medium, accessed July 11, 2025, <https://medium.com/@abelbekele.addise/prompt-tuning-for-building-enterprise-grade-rag-systems-f0437281c26b>
59. How do I integrate LangChain with messaging platforms like Slack or Teams? - Zilliz, accessed July 11, 2025, <https://zilliz.com/ai-faq/how-do-i-integrate-langchain-with-messaging-platforms-like-slack-or-teams>
60. Create a Teams AI Bot with RAG - Learn Microsoft, accessed July 11, 2025, <https://learn.microsoft.com/en-us/microsoftteams/platform/toolkit/build-a-rag-bot-in-teams>
61. How to Transform Your Slack Support Channel in a Weekend with a RAG-Powered, Talking Chatbot, accessed July 11, 2025, <https://ragaboutit.com/how-to-transform-your-slack-support-channel-in-a-weekend-with-a-rag-powered-talking-chatbot/>
62. AI Adoption: Driving Change With a People-First Approach - Prosci, accessed July 11, 2025, <https://www.prosci.com/blog/ai-adoption>
63. Product adoption: The missing piece in your digital strategy | Think Company, accessed July 11, 2025, <https://www.thinkcompany.com/blog/internal-tool-digital-product-adoption-strategy/>
64. AI Adoption: Why People Are Key to Early Adoption - Whatfix, accessed July 11, 2025, <https://whatfix.com/blog/ai-adoption-strategy/>
65. How to drive internal AI adoption - Mind the Product, accessed July 11, 2025, <https://www.mindtheproduct.com/how-to-drive-internal-ai-adoption/>