

# LEGACYAI: A Comprehensive Framework For Preserving Professional Expertise Via Ai-Powered Knowledge Hubs

SAGAR BISHT, MRIDUL BHADULA, TEJAS DIXIT, MR. ABDUL QADIR

Department of Computer Science and Engineering (Data Science),

Inderprastha Engineering College, Ghaziabad, India

© The Author(s), under exclusive license to publication division, IPEC Journal of Science & Technology, 2026

**Abstract**—Organizations accumulate critical practical knowledge in experienced employees, but workflows, troubleshooting patterns, project lessons, and tacit know-how can be lost when those employees retire or voluntarily leave. LegacyAI is proposed as an AI-powered organizational knowledge hub that captures multimedia expertise, organizes it by enterprise context, and makes it searchable through an AI assistant. This review examines the knowledge-management foundations of the project, the limitations of conventional learning and knowledge repositories, the role of multimedia capture, and the use of retrieval-augmented generation for context-aware guidance. The reviewed architecture separates presentation, authentication, application, AI processing, data, and infrastructure/security layers and supports Super Admin, Company Admin, and Employee roles. The updated implementation stack uses Python and FastAPI; Ollama with Llama 3.1 and LlamaIndex; OpenAI text-embedding-3-small; PyMuPDF, pdfplumber, python-docx, and RecursiveCharacterTextSplitter; spaCy and scikit-learn; Pinecone, SQLAlchemy, Cloudflare R2; JWT-based authentication; and React.js with Tailwind CSS. The paper positions LegacyAI as an integration of knowledge preservation, multimedia processing, semantic retrieval, and conversational assistance rather than as a claim of a new foundational AI algorithm. Since the project materials define a proposed system and do not provide completed benchmark results, this review specifies an evaluation framework covering ingestion quality, retrieval relevance, answer groundedness, onboarding effectiveness, latency, cost, security, and multi-tenant isolation. The review concludes that the principal research opportunity lies in transforming departing employees' expertise into a secure, traceable, continuously improving organizational memory.

**Keywords**—Organizational knowledge management, tacit knowledge, knowledge preservation, retrieval-augmented generation, Llama 3.1, LlamaIndex, semantic search, Pinecone, multi-tenancy, enterprise AI.

## I. INTRODUCTION

Organizations accumulate knowledge through projects, customer interactions, operational incidents, engineering decisions, and the accumulated experience of employees. A significant part of this knowledge is not written in formal manuals. It exists as practical judgment: why a workflow is performed in a particular sequence, which failure patterns are common, how an exception is handled, and which shortcuts should be avoided. The LegacyAI project is motivated by the loss of this institutional knowledge when experienced employees retire or voluntarily leave an organization. The project synopsis describes a platform that captures training videos, work procedures, best practices, project experiences, common mistakes, productivity tips, and standard operating procedures before the knowledge holder exits [14].

The problem is therefore broader than document management. Conventional knowledge repositories can preserve files, but they do not necessarily preserve the context in which those files are used. Learning-management systems can support structured training, while enterprise knowledge-management platforms can centralize manuals and collaboration artifacts. However, the project identifies a gap in preserving practical expertise from departing employees and making that expertise directly useful to future employees through interactive AI guidance [14].

LegacyAI proposes to close this gap through a knowledge hub that combines multimedia capture, structured metadata, semantic retrieval, and an AI assistant. The architecture shown in the project presentation separates presentation, user access and authentication, application, AI knowledge processing, data, and infrastructure/security layers. The intended flow begins with company registration and role authentication, continues through expert knowledge upload and AI processing, and ends with a searchable knowledge hub that supports employee learning and AI-assisted guidance [10].

This paper reviews the conceptual foundations, related systems, research gap, and proposed technical design of LegacyAI. It treats the system as a proposed knowledge-management and retrieval architecture rather than a completed experimental product. Consequently, expected benefits such as faster onboarding and fewer repeated mistakes are discussed as intended outcomes, not as experimentally established numerical improvements. The review also incorporates the updated implementation stack supplied for the project: Python and FastAPI for the backend; Ollama with Llama 3.1 and LlamaIndex for orchestration; OpenAI text-embedding-3-small for embeddings; PyMuPDF, pdfplumber, python-docx, and RecursiveCharacterTextSplitter for document processing; spaCy and scikit-learn for NLP utilities; Pinecone

and SQLModel for data management; Cloudflare R2 for cloud storage; JWT-based authentication; and React.js with Tailwind CSS for the frontend.

The review is also motivated by the distinction between organizational memory and organizational search. A repository can preserve information while still leaving employees unable to find the right artifact at the moment of need. LegacyAI attempts to combine preservation with retrieval by turning source material into indexed knowledge units and placing an AI interface over those units. In this sense, the system is designed as a bridge between long-term organizational memory and short-term task support. The quality of that bridge depends on accurate capture, metadata, retrieval, authorization, and evidence-grounded generation.

## II. FOUNDATIONS OF ORGANIZATIONAL KNOWLEDGE MANAGEMENT

Knowledge management distinguishes between information that can be readily codified and knowledge that is embedded in human experience. Nonaka and Takeuchi describe organizational knowledge creation as a dynamic process in which tacit and explicit knowledge interact [14]. This distinction is particularly relevant to LegacyAI because an employee's practical expertise may not be recoverable from formal documents alone. A procedure can state what to do, while an experienced practitioner can explain why the procedure matters, what signals indicate an abnormal situation, and what common mistakes should be avoided.

Davenport and Prusak emphasize that knowledge is more valuable when it is connected to experience, context, and interpretation rather than treated as isolated data [1]. In an organizational setting, this means that a repository should preserve not only documents but also the narratives, examples, decisions, and lessons that make those documents actionable. LegacyAI reflects this principle by explicitly collecting project experiences, best practices, mistakes and solutions, productivity tips, and training material in addition to SOPs [14].

Alavi and Leidner position knowledge-management systems as socio-technical systems that support knowledge creation, storage/retrieval, transfer, and application [1]. These stages provide a useful conceptual lens for reviewing LegacyAI. The collection phase addresses creation and capture; the repository addresses storage; semantic indexing addresses retrieval; the employee portal and chatbot support transfer; and workflow guidance supports application. The architecture therefore moves beyond a passive archive toward an operational knowledge service.

A further foundation is the treatment of knowledge as an organizational asset. Drucker's discussion of the knowledge-based organization highlights the increasing importance of knowledge workers and knowledge as a productive resource [14]. In LegacyAI, the central design question becomes how to transform individual expertise into a durable

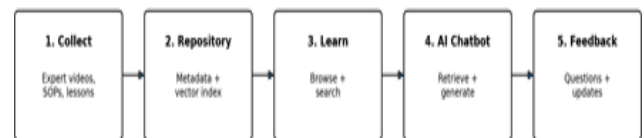
organizational asset without requiring the original expert to remain continuously available. The platform is intended to reduce single-point dependence while keeping human experience accessible through reusable digital artifacts.

From these foundations, four requirements emerge: the system must capture context, preserve multiple modalities, retrieve relevant evidence, and present the result in a form that supports action. A repository that satisfies only storage is insufficient. LegacyAI's proposed integration of multimedia knowledge capture and AI retrieval can be understood as an attempt to satisfy all four requirements within one enterprise workflow.

The framework of knowledge creation, storage, transfer, and application also clarifies why feedback is necessary. Knowledge is not static once it has been captured. New projects can invalidate old procedures, organizational roles can change, and lessons learned can become obsolete. A useful knowledge-management platform therefore needs mechanisms for revision and social validation. LegacyAI's feedback stage can be interpreted as a lightweight governance mechanism through which employees identify gaps and administrators decide which new evidence should be added or updated.

## III. TACIT KNOWLEDGE LOSS AND MULTIMEDIA CAPTURE

Figure 2. Five-phase LegacyAI knowledge lifecycle



The principal business problem identified by the project is knowledge loss during employee exit. The synopsis lists six connected effects: loss of domain knowledge, increased training time, repeated mistakes, dependence on senior employees, lack of centralized learning resources, and reduced productivity during onboarding [14]. These effects form a chain rather than isolated problems. When expertise is unavailable, new employees spend more time asking senior staff for help; when guidance is unavailable, avoidable errors are more likely to recur; and repeated support requests increase dependence on a small number of experts.

Traditional documentation has an important role but has practical limitations. Formal documents tend to reflect standardized processes, whereas expert demonstrations often contain exceptions, troubleshooting sequences, and tacit decision rules. The LegacyAI presentation therefore emphasizes multimedia capture, particularly videos, SOPs, documents, and project narratives [10]. A video can preserve not only the words used by an expert but also the sequence of actions and the demonstration of a workflow. Such content can subsequently be converted into searchable knowledge

through processing and indexing.

The five-phase methodology in the project materials provides a practical lifecycle: collect knowledge, organize it in a repository, enable employee learning, provide an AI chatbot, and continuously improve the repository through feedback [14],[10]. During collection, departing or retiring experts contribute training videos, work procedures, best practices, project experiences, common mistakes and solutions, productivity tips, and SOPs. During repository formation, content is categorized by department, technology, project, skill level, and work process.

The multimedia orientation introduces additional engineering requirements. Documents require format-specific extraction, while videos require an additional conversion from audiovisual content into searchable representations. The architecture in the presentation explicitly includes OCR and speech-to-text in the AI knowledge-processing layer [10]. The updated stack supplied for the project specifies mature document-processing components but does not name a dedicated speech-to-text engine. Therefore, speech-to-text should be regarded as an architectural requirement whose concrete implementation can be selected separately, rather than as a completed capability attributed to the current stack.

The most important design implication is that capture should be structured rather than treated as arbitrary file upload. Metadata such as department, technology, project, skill level, employee role, source type, and upload date can make later retrieval more precise. It also creates a foundation for multi-tenant isolation, role-based access, content lifecycle management, and analytics.

A useful capture process should also preserve the provenance of expert knowledge. Each artifact should identify who contributed it, what role the contributor held, which department or project it relates to, and whether the content represents a formal procedure, a personal recommendation, or a historical lesson. Provenance allows users to interpret knowledge appropriately and allows administrators to resolve conflicts between sources. It also supports later analytics on which expertise areas are well covered and which depend on only one contributor.

Multimedia capture also raises a distinction between raw media and knowledge-ready media. A long training video may contain several independent procedures, examples, and digressions. Treating the entire file as one retrieval object would make precise search difficult. The processing pipeline should therefore identify meaningful segments, preserve timestamps, and associate each segment with descriptive metadata. A document can be treated similarly by identifying headings, sections, tables, and paragraphs before chunking. The goal is to transform a large artifact into a collection of coherent evidence units without losing the link to the original source.

#### **IV. REVIEW OF EXISTING KNOWLEDGE AND LEARNING SYSTEMS**

The project synopsis identifies learning-management

systems such as Moodle, Blackboard, and TalentLMS as examples of platforms used for employee training, while knowledge-management systems such as SharePoint and Confluence are cited as repositories for documents and manuals [14]. These categories solve meaningful parts of the organizational learning problem, but the project frames their limitation as a document-centric orientation. The review gap is not that these platforms are incapable of storing information; rather, it is that they are not specifically organized around capturing the practical expertise of employees at the point of exit and making that expertise conversationally actionable.

Learning-management systems are generally strong at course organization, assessment, learner enrollment, and structured training. Their limitation for the LegacyAI problem is that valuable knowledge may not exist as a predefined course. An expert may instead possess a collection of troubleshooting demonstrations, project lessons, and informal explanations. A knowledge-management platform can store such material, but the user may still need to know where to look and how multiple documents relate to a question.

AI assistants improve the interaction model by allowing employees to ask questions in natural language. Retrieval-augmented generation (RAG) is particularly relevant because it separates the generative model from the organization's knowledge corpus. Lewis et al. introduced RAG as a framework that combines a parametric generator with a non-parametric external memory [15]. In the context of LegacyAI, this principle supports answers grounded in the organization's stored documents and expert material rather than relying solely on general model knowledge.

The emergence of instruction-following and locally runnable language models also changes the deployment trade-off. LegacyAI specifies Ollama with Llama 3.1 for the LLM layer and LlamaIndex for orchestration. A locally served model can provide a controlled inference boundary for selected workloads, while an embedding model can independently represent knowledge chunks for semantic retrieval. This separation is consistent with a modular architecture in which ingestion, indexing, retrieval, generation, and access control can evolve independently.

The comparative lesson is that LegacyAI should not attempt to replace every LMS or KMS feature. Its distinct role is to act as an organizational memory layer that captures expert knowledge, connects it to structured metadata, and provides retrieval and guidance. Integration with existing HR, LMS, or enterprise platforms is therefore a logical future direction rather than a prerequisite for the core system.

The review therefore suggests a complementary rather than competitive relationship between LegacyAI and existing enterprise platforms. Existing KMS and LMS products can remain systems of record or systems of structured learning, while LegacyAI can operate as an intelligent knowledge-access layer. Such an architecture would allow organizations to preserve existing governance and content workflows while adding semantic retrieval and conversational access.

The future integration direction identified by the project presentation is consistent with this layered view.

The conversational interface also changes the user's cognitive load. Instead of learning the information architecture of a repository, an employee can express a task in natural language and receive a response that points to relevant evidence. This is particularly useful during onboarding because users may not yet know the internal terminology used to classify documents. However, conversational access should supplement rather than replace direct browsing. Employees need the ability to inspect original sources, review related material, and understand whether a recommendation is current or historical.

## V. COMPARATIVE REVIEW AND RESEARCH GAP

The literature and project materials suggest an integration gap across four capabilities: practical expertise capture, multimedia knowledge organization, semantic retrieval, and context-aware guidance. Conventional KMS solutions provide strong storage and collaboration capabilities but may leave practical experience fragmented. LMS platforms provide structured learning but are not primarily designed to preserve tacit knowledge from departing employees. Generic AI chatbots provide conversational interaction but can lack access to organization-specific evidence. LegacyAI combines these capabilities around a single organizational knowledge lifecycle.

**Table I. Comparison of organizational knowledge approaches.**

| Approach   | Primary strength             | Main limitation               | LegacyAI distinction            |
|------------|------------------------------|-------------------------------|---------------------------------|
| KMS        | Centralized documents        | Context may remain fragmented | Expert context + retrieval      |
| LMS        | Structured learning          | Course-oriented content       | On-demand knowledge             |
| Generic AI | Natural-language interaction | No inherent company corpus    | Tenant-specific RAG             |
| LegacyAI   | Multimedia + semantic AI     | Needs governance/evaluation   | Expert capture + RAG + feedback |

Table I summarizes the distinction. A document repository can preserve explicit artifacts but offers limited support for tacit expertise. An LMS structures learning but may require content to be converted into formal courses. A generic AI assistant offers natural-language interaction but does not inherently preserve company-specific knowledge. LegacyAI is designed around the missing intersection: capture the expertise before exit, process heterogeneous material, organize it by enterprise context, retrieve relevant evidence, and provide employee guidance through an AI assistant.

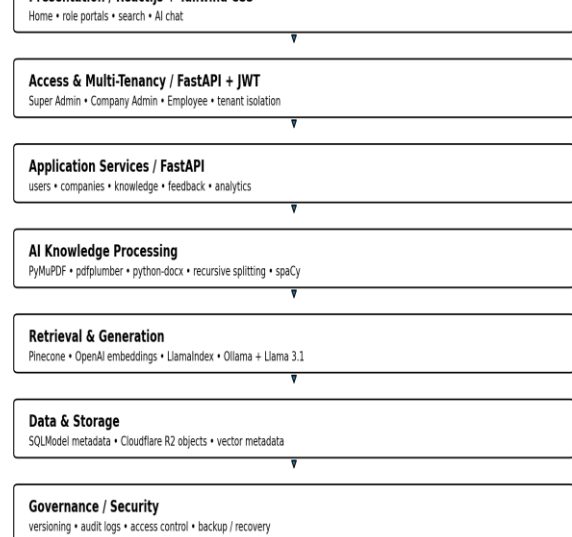
The research gap is consequently an integration and actionability gap rather than a claim that knowledge management, multimedia learning, embeddings, or LLMs are individually novel. The project materials explicitly identify document bias, limited mechanisms for retiree/departing-employee knowledge capture, weak context-aware AI integration, missing practical demonstrations, and limited

The research gap can also be framed in terms of the unit of knowledge. Traditional systems often organize information around files, pages, courses, or messages. LegacyAI proposes organizing information around reusable knowledge units connected to enterprise context. A chunk from an SOP, a segment of an expert video, and a project lesson can all become retrievable units, while their relationships to a department, technology, project, and skill level remain available as metadata. This unit-level representation is important for building an assistant that can answer narrowly scoped operational questions.

The proposed system can therefore be positioned across three axes: preservation depth, retrieval intelligence, and organizational context. Preservation depth increases when the platform captures both formal documentation and expert demonstrations. Retrieval intelligence increases when semantic indexing and RAG replace manual navigation. Organizational context increases when tenant, department, project, role, and skill metadata influence retrieval. LegacyAI's research value lies in combining all three axes in one lifecycle rather than optimizing only one component.

## LEGACYAI FRAMEWORK AND PROPOSED ARCHITECTURE

**Figure 1. Logical architecture of the proposed LegacyAI knowledge hub**



The proposed architecture in the project presentation is

layered. The presentation layer exposes the home page and entry points such as registration, login, contact, and support. The user-access layer separates Super Admin, Company Admin, and Employee roles. The application layer provides portals for each role. The AI knowledge-processing layer handles uploaded content and performs NLP, OCR, speech-to-text, chunking, and embeddings. The data layer maintains user, company, employee, knowledge, metadata, logs, and analytics information. Infrastructure and security provide cloud hosting, access control, privacy, encryption, backup, and recovery [10].

The system flow is equally important. A company registers; an administrator approves or manages the organization; company administrators invite experts and manage employees; experts upload knowledge; the AI processing layer converts that knowledge into searchable representations; the knowledge hub becomes available to employees; and employees learn and obtain support through the AI response quality.

The updated backend uses Python and FastAPI. FastAPI is suitable for separating authentication, company management, knowledge ingestion, search, chatbot, feedback, and analytics endpoints into modular services. JWT-based authentication provides the proposed identity mechanism, while tenant identifiers can be attached to authenticated requests and propagated through every data-access path. This is particularly important because the architecture supports multiple companies rather than a single shared knowledge base.

The frontend is implemented conceptually as a React.js application styled with Tailwind CSS. The role-specific portals shown in the project architecture can therefore be represented as protected React routes and reusable interface components. The Employee Portal can emphasize search, AI questions, assigned content, and learning progress; the Company Admin Portal can emphasize employee management, expert invitations, uploads, knowledge management, and analytics; and the Super Admin Portal can emphasize company approval, user administration, and platform-level reporting.

At the storage boundary, Cloudflare R2 provides object storage for original documents and multimedia assets, while Pinecone provides vector retrieval and SQLModel provides relational application metadata. This division is useful because binary content, semantic vectors, and transactional entities have different access patterns. The architecture should preserve stable object identifiers and tenant metadata so that retrieval results can be traced back to an authorized source artifact.

Figure 1 represents the resulting logical architecture. The key design principle is separation of concerns: source files are not the vector database; vector retrieval is not the LLM; and the LLM is not the authorization layer. Each layer has a defined responsibility and can be tested

independently.

Role separation should be reflected consistently across the architecture. A Super Admin operates at the platform level, a Company Admin governs one organization, and an Employee consumes assigned knowledge and asks questions. Expert contributors can be modeled as employees with elevated content-contribution permissions or as a dedicated role if the implementation requires it. The presentation's distinction between company administration and employee learning supports this separation[10]. Such a model reduces the risk that content Video content requires a parallel route. The presentation identifies management, platform administration, and knowledge consumption become mixed into a single privilege set.

A modular architecture also supports phased development. An initial release can implement authentication, tenant management, document upload, extraction, chunking, embeddings, vector search, and a grounded chatbot. Subsequent releases can add video transcription, learning progress, feedback analytics, richer administration, and integrations. This sequencing is consistent with the five-phase methodology because the system can first establish a reliable repository and then progressively increase the intelligence and personalization of the learning experience.

**Table II. Updated LegacyAI technology stack.**

| Layer               | Technology                                                       | Role                                  |
|---------------------|------------------------------------------------------------------|---------------------------------------|
| Backend             | Python + FastAPI                                                 | APIs, services, auth integration      |
| LLM / orchestration | Ollama + Llama 3.1; LlamaIndex                                   | Local inference and RAG orchestration |
| Embeddings          | OpenAI text-embedding-3-small                                    | Semantic vector representations       |
| Processing          | PyMuPDF; pdfplumber; python-docx; RecursiveCharacterTextSplitter | Extraction and chunking               |
| NLP                 | spaCy; scikit-learn                                              | Entity and text utilities             |
| Data                | Pinecone; SQLModel                                               | Vector search and relational metadata |
| Storage / auth      | Cloudflare R2; JWT                                               | Object storage and identity           |
| Frontend            | React.js; Tailwind CSS                                           | Role-based web interface              |

## VI. MULTIMEDIA INGESTION AND KNOWLEDGE PROCESSING

Knowledge quality begins with ingestion. The updated

stack includes PyMuPDF and pdfplumber for PDF processing and python-docx for Word documents. These components can extract text while retaining useful document boundaries and metadata. The ingestion service should normalize extracted content into a common internal representation containing source identifier, tenant identifier, document type, section or page information where available, upload metadata, and content text.

After extraction, content must be segmented into retrieval units. The project specifies langchain.text\_splitter with RecursiveCharacterTextSplitter. Recursive splitting is appropriate for enterprise documents because it attempts to preserve larger semantic boundaries before falling back to smaller separators. Chunk size and overlap should be treated as evaluation parameters rather than arbitrary constants. Chunks that are too small can lose context, while very large chunks can dilute retrieval precision and increase prompt size.

NLP utilities provide additional structure. spaCy can support sentence segmentation, entity recognition, linguistic normalization, and extraction of domain terms. scikit-learn can support lightweight classification or similarity utilities where deterministic statistical methods are sufficient. The purpose is not to replace the language model but to enrich the ingestion and metadata pipeline with predictable preprocessing operations.

The embedding stage converts chunks into dense vector representations. The project specifies OpenAI text-embedding-3-small for this role. Each vector should be stored with metadata sufficient for filtering, including tenant, department, project, technology, skill level, work process, source type, and document identifier. Metadata filtering is essential in enterprise retrieval because semantic similarity alone does not enforce access policy or organizational scope.

speech-to-text as part of the AI processing layer [10]. A practical pipeline would extract or receive a transcript, normalize it, associate time offsets where available, segment it into retrieval chunks, generate embeddings, and retain a reference to the original video object. The review does not treat a specific speech-to-text model as finalized because the updated stack does not specify one. Future implementation can compare a local or managed transcription service based on accuracy, cost, latency, language coverage, and privacy requirements.

Ingestion should be idempotent. If the same document is uploaded twice, the system should be able to detect duplicate or unchanged content using a content hash or stable version identifier. This prevents redundant vector records and allows controlled replacement when a document is updated. Versioning is especially important for SOPs because employees should be able to distinguish current procedures from superseded guidance.

Chunk-level provenance is particularly valuable when answers need to be audited. If a chatbot states that a

particular process should be followed, the system should be able to identify the document or video from which the supporting chunk was retrieved. For video, a timestamp can provide a direct route to the relevant demonstration. For PDFs and DOCX files, page or section information can provide a similarly useful navigation target. Provenance therefore improves both user trust and administrator review without requiring the model to reveal internal retrieval details.

The ingestion service should treat errors as first-class states. Password-protected or corrupted PDFs, malformed DOCX files, unsupported formats, empty transcripts, and extraction failures should not silently enter the vector store. Instead, the system should record a failure reason and expose it to the administrator. This is important because a knowledge hub can appear operational while quietly omitting difficult documents. Processing dashboards can therefore report successful, failed, pending, and archived artifacts and provide administrators with a clear route to remediation.

## VII. RETRIEVAL-AUGMENTED AI KNOWLEDGE ENGINE

The central intelligence of LegacyAI is a retrieval-and-generation pipeline. LlamaIndex provides orchestration around the knowledge index and retrieval workflow, while Ollama with Llama 3.1 supplies the language-model layer. The conceptual sequence is query understanding, retrieval, context assembly, generation, and response presentation. The system should prefer evidence from the organization's indexed knowledge rather than allowing the model to answer from general knowledge when a company-specific procedure is expected.

A retrieval request should first be associated with the authenticated tenant and, where relevant, the employee's department, role, project, and access scope. Pinecone can then perform semantic similarity search over authorized vectors. Metadata filters can reduce the candidate set before or alongside vector similarity. The top retrieved chunks are assembled into a context package containing source identifiers and relevant excerpts. This approach reduces the chance that the model receives unrelated knowledge from another department or tenant.

RAG changes the reliability problem from 'Can the model know everything?' to 'Can the system retrieve the right evidence and make the model use it faithfully?' Lewis et al. demonstrate the value of combining a generative model with an external knowledge source [15]. For LegacyAI, this is particularly important because object-storage metadata, and vector metadata so that a single missing organizational procedure can be proprietary, versioned, and different from general industry practice. The knowledge hub should therefore prioritize source-grounded responses and expose references to the underlying material

where possible.

The LlamaIndex orchestration layer can support modular components for loaders, indexes, retrievers, prompt templates, and response synthesis. This modularity also allows future experiments with hybrid retrieval, reranking, query expansion, and conversation memory. However, these additions should be introduced only when evaluation shows a retrieval bottleneck. Complexity should not be added merely because a component is available.

A robust prompt policy should instruct the model to distinguish retrieved organizational evidence from unsupported inference. When the retrieved context does not contain a reliable answer, the assistant should say that the knowledge base does not provide enough information and guide the employee toward an administrator or expert. This behavior is preferable to fabricating a company-specific procedure. The feedback system can then capture the unanswered question as a candidate knowledge-gap signal.

Evaluation should separate retrieval quality from generation quality. Retrieval can be measured with recall-oriented and ranking-oriented metrics on a curated question set, while answer quality can be assessed for groundedness, completeness, relevance, and refusal behavior. The project materials do not report such measurements rather than established results.

The assistant should also support conversational context carefully. A follow-up question such as “What if the first method fails?” may depend on the previous turn, but conversation memory must remain within the user's authorized tenant and knowledge scope. Context should not become an alternative route around authorization. A safe design can maintain a short conversation state while reapplying tenant and permission filters to every retrieval operation. This preserves the usefulness of conversational interaction without allowing earlier retrieved content to silently broaden access.

A useful retrieval pipeline can also adopt a confidence-aware response policy. If several highly relevant chunks agree, the assistant can answer directly and cite the supporting sources. If relevant chunks conflict, the assistant should identify the conflict and prefer the newest approved version where governance metadata supports that decision. If retrieval scores are weak or no authorized source is found, the assistant should avoid presenting a speculative company-specific answer. This policy makes retrieval confidence part of the interaction design rather than an invisible backend score.

### **VIII. MULTI-TENANCY, SECURITY, AND DATA MANAGEMENT**

LegacyAI is designed as a platform that can serve multiple organizations, making tenant isolation a core architectural property. The project presentation explicitly separates

Super Admin, Company Admin, and Employee roles and includes security and access control within the infrastructure layer [10]. JWT-based authentication in the updated stack provides a mechanism for carrying identity and authorization claims between the frontend and backend. Authentication alone, however, is not sufficient for multi-tenancy; every data-access operation must enforce tenant boundaries.

A practical authorization model can associate each user with a company or tenant identifier and one or more roles. Company administrators can manage employees and expert uploads within their organization. Employees can access only the knowledge assigned or authorized for their organization. Super administrators can manage registered companies and platform-level operations. The same tenant identifier should be carried into relational records, object-storage metadata, and vector metadata so that a single missing filter does not expose cross-tenant content. Cloudflare R2 is suitable for storing original documents, videos, and other large artifacts, while SQLModel can represent transactional metadata such as users, companies, roles, content records, upload status, versions, and feedback. Pinecone can store embeddings and retrieval metadata. This separation enables the system to avoid placing large binary files in the relational layer while still maintaining strong references between source content and searchable representations.

Security must also cover the retrieval path. A common failure mode in RAG systems is enforcing access control at upload time but not at query time. LegacyAI should apply authorization before vector retrieval or through tenant-aware metadata filters and should validate the returned sources before including them in the LLM context. The model should never receive unauthorized content even if the vector store contains it.

Data governance is equally important. Enterprise knowledge may contain confidential procedures, customer information, credentials, or personal data. The system should therefore support content ownership, retention rules, versioning, deletion propagation, and audit logging. When a source document is deleted or access is revoked, its object, relational record, vector records, and cached representations should be removed or invalidated consistently.

JWT secrets, API credentials, embedding-service keys, and storage credentials should remain outside source code. The architecture should use environment-based secret management and short-lived access tokens where practical. Future deployments should additionally consider encryption in transit, encryption at rest, backup and recovery, rate limiting, input validation, file-type validation, malware scanning, and prompt-injection defenses.

Prompt injection is another enterprise-specific concern. Uploaded documents may contain instructions that are irrelevant to the employee's question but attempt to influence the model. The retrieval pipeline should treat retrieved material as data to be analyzed rather than as higher-priority system instructions. The response layer can also restrict tool access and prevent the model from executing arbitrary actions. These controls are especially important when the knowledge hub later integrates with HR, ticketing, or other operational systems.

Deletion and access revocation should be treated as distributed consistency problems. Removing a source from R2 while leaving its vector representation searchable creates a residual information path. Similarly, changing a document's permissions without updating vector metadata can leave old content accessible through semantic search. The implementation should therefore define source lifecycle events that update object storage, relational metadata, vector records, caches, and audit logs together. A periodic consistency job can identify orphaned vectors or objects and flag them for remediation.

## IX. EVALUATION FRAMEWORK AND RESEARCH QUESTIONS

Because LegacyAI is currently a proposed system, evaluation should be designed before performance claims are made. The primary objective is to determine whether the integrated architecture preserves knowledge, retrieves relevant evidence, produces grounded assistance, and reduces operational friction. The project synopsis provides intended outcomes such as faster onboarding, fewer repeated mistakes, improved learning efficiency, and continuous access to expert guidance [14]. These outcomes can be translated into measurable research dimensions without assuming that improvement has already been demonstrated.

RQ1: Can the ingestion pipeline reliably convert heterogeneous enterprise documents into searchable, metadata-rich knowledge units? Evaluation can measure extraction success rate, chunk validity, duplicate detection, processing time, and metadata completeness. RQ2: Can the retrieval layer return the relevant source passages for employee questions? A benchmark of representative questions can measure retrieval recall, precision at k, mean reciprocal rank, or nDCG. RQ3: Can the AI assistant produce grounded answers from retrieved evidence? Human reviewers can rate relevance, completeness, faithfulness to sources, and unsupported claims.

RQ4: Does metadata-aware retrieval improve enterprise relevance compared with vector similarity alone? This can be tested through controlled ablations in which tenant, department, project, and skill-level filters are

progressively enabled. RQ5: Does the feedback loop reveal actionable knowledge gaps? Questions that fail to retrieve

useful content can be categorized by topic and mapped to missing or outdated source material. RQ6: Does the architecture preserve tenant isolation under adversarial or cross-tenant query attempts? Security testing should include authorization failures, vector-filter bypass attempts, direct-object access attempts, and prompt-injection scenarios.

For onboarding effectiveness, a controlled study can compare a conventional training condition with a LegacyAI-assisted condition using equivalent tasks. Useful measures include time-to-task-completion, number of support requests, number of repeated errors, and learner confidence. Because the current project materials do not provide experimental data, these measures should be presented as proposed validation criteria rather than claimed improvements.

The evaluation should also include system-level measurements: API latency, ingestion throughput, vector search latency, LLM response latency, storage cost, embedding cost, and memory/compute utilization. Reliability tests should include large files, concurrent users, duplicate uploads, malformed documents, incomplete transcripts, and unavailable model services. These tests can reveal whether the architecture remains practical beyond a small demonstration.

Evaluation should use realistic enterprise questions rather than only generic fact-retrieval benchmarks. A useful dataset can contain procedural questions, troubleshooting questions, terminology questions, cross-document questions, intentionally unanswerable questions, outdated-procedure questions, and questions whose correct answer differs by department. This test design would expose whether the system understands organizational scope rather than merely finding semantically similar text. Human reviewers should additionally verify that the response reflects the retrieved evidence and does not introduce unsupported policy.

A particularly useful experiment is an ablation study of the complete pipeline. A baseline can use keyword or manual search, followed by vector-only retrieval, metadata-filtered retrieval, and full RAG with source grounding. The comparison can reveal which layer contributes most to useful answers. A second ablation can compare different chunk sizes and overlaps. A third can compare local Llama inference settings. Such experiments would turn architectural choices into measurable research questions and reduce the risk of selecting components based only on popularity.

**Table III. Proposed evaluation dimensions.**

| Dimension  | Measure                                                   | Purpose                    |
|------------|-----------------------------------------------------------|----------------------------|
| Ingestion  | Extraction success, metadata completeness, throughput     | Pipeline reliability       |
| Retrieval  | Recall@k, MRR/nDCG, source relevance                      | Evidence retrieval quality |
| Generation | Groundedness, relevance, completeness, unsupported claims | Answer quality             |
| Learning   | Task time, support requests, repeated errors              | Onboarding impact          |
| Security   | Cross-tenant leakage tests, auth failures                 | Tenant isolation           |
| System     | Latency, cost, failure rate                               | Operational feasibility    |
| Feedback   | Unanswered-question categories, closure rate              | Continuous improvement     |

## X. DISCUSSION

The strongest aspect of LegacyAI is the alignment between a concrete organizational problem and an end-to-end architecture. The project does not treat AI as an isolated chatbot feature. Instead, the chatbot is placed after knowledge capture, processing, indexing, and storage. This ordering matters because a conversational interface without a reliable knowledge foundation cannot solve the underlying knowledge-loss problem.

The multimedia emphasis is another important strength. Documents are valuable, but expert knowledge frequently includes procedural demonstrations and contextual explanations. The architecture's explicit inclusion of video, PDF, document, OCR, speech-to-text, chunking, and embeddings indicates a deliberate attempt to preserve more than text alone [10]. The updated stack strengthens this design by providing specialized document extraction tools and a vector retrieval layer.

The architecture also benefits from modularity. FastAPI can expose clean service boundaries; LlamaIndex can orchestrate retrieval; Pinecone can handle semantic indexing; SQLModel can manage transactional metadata; R2 can store large objects; and React/Tailwind can provide role-specific interfaces. Such separation makes the platform easier to evolve than a monolithic implementation. It also enables component-level testing and replacement.

However, the system's main technical risk is retrieval reliability. Enterprise questions are often ambiguous and context-dependent. A semantically similar chunk may still be operationally wrong if it belongs to an outdated SOP, a different department, or another project. Metadata filtering, source versioning, access control, and answer citation are therefore not optional enhancements; they are central to trustworthiness.

A second risk is knowledge freshness. Organizational knowledge changes continuously. The feedback loop described in the project is a valuable mechanism, but it should be expanded into a lifecycle model involving content ownership, expiry or review dates, version status, and administrator approval. A knowledge hub should be able to distinguish current guidance from historical experience.

Finally, the project should preserve a clear boundary between expert knowledge and model-generated interpretation. The AI assistant can make stored expertise easier to access, but it should not silently turn uncertain evidence into authoritative policy. Source attribution, uncertainty-aware responses, and escalation to human experts are important design principles for an enterprise system.

The proposed architecture also creates an opportunity to measure organizational learning as a system-level phenomenon. If repeated employee questions are grouped by topic, the platform can identify recurring friction points. A high volume of questions about one workflow may indicate that the workflow is poorly documented, poorly taught, or intrinsically complex. In this way, the knowledge hub can evolve from a repository into an analytics-supported improvement mechanism, provided that feedback data are interpreted carefully and privacy requirements are respected.

There is also a human-factors dimension. Employees may hesitate to trust an AI assistant if it produces confident but incorrect answers, while administrators may hesitate to maintain a system that does not show how its knowledge is being used. Visible sources, feedback controls, content freshness indicators, and transparent escalation paths can address these concerns. The platform should make it easy to report a wrong answer, identify the underlying source, and request an update. This creates a feedback channel that benefits both system quality and organizational governance.

## XI. LIMITATIONS AND FUTURE DIRECTIONS

The current project definition has several limitations. First, the synopsis and presentation specify the architecture and expected outcomes but do not provide completed experimental results. This review therefore does not claim measured reductions in onboarding time, error rates, or support workload. Second, the updated technology stack does not specify a concrete speech-to-text component even though the architecture includes speech-to-text for multimedia processing [10]. This remains an implementation choice.

Third, retrieval quality depends strongly on source quality. Poorly recorded videos, incomplete SOPs, contradictory documents, or outdated project notes can produce weak answers even when the retrieval system is technically correct.

Future work should therefore investigate knowledge curation, expert validation, content scoring, and source freshness. The platform could assign confidence or freshness indicators to knowledge artifacts.

Fourth, future work can introduce automated video summarization and role-based learning paths, both identified in the project presentation [10]. A role-based learning engine could use department, skill level, assigned content, and learning progress to recommend a sequence of materials. Analytics could identify topics that generate repeated questions, revealing areas where documentation or training should be improved.

The system can also be extended with multilingual support, integration with HR and LMS platforms, and enterprise search connectors. These directions are explicitly identified in the presentation's future scope [10]. Additional retrieval research could compare dense retrieval with hybrid lexical-semantic retrieval, reranking, query decomposition, and conversational retrieval. Model-serving research could compare local Llama inference against managed LLM endpoints for different privacy, cost, and latency requirements.

A more advanced future architecture could represent organizational knowledge as a graph linking employees, departments, projects, technologies, procedures, documents, and lessons learned. Graph-based relationships could complement vector similarity for questions that require structured navigation across multiple entities. Another direction is automated knowledge-gap mining from unanswered questions and repeated support tickets, creating a prioritization queue for administrators.

Finally, deployment at enterprise scale will require stronger governance. Future work should include audit trails, approval workflows, content retention policies, fine-grained permissions, secure deletion, model and prompt versioning, evaluation datasets, and red-team testing for prompt injection and data exfiltration. These are necessary if the knowledge hub is to become a dependable organizational system rather than only a prototype chatbot.

Another limitation is the dependence on model and embedding choices. Changes in the embedding model can alter vector neighborhoods, while changes in the LLM can alter response style and groundedness. Production deployments should therefore version embeddings, prompts, model configurations, and evaluation sets. Re-indexing may be necessary when an embedding model changes. Similarly, local Llama inference through Ollama may have different latency and resource requirements across deployment hardware, making infrastructure benchmarking an important future activity.

Scalability is another open question. Vector databases can support large collections, but the number of chunks can grow rapidly when organizations ingest years of documents and

many hours of video. Efficient deduplication, metadata filtering, archival policies, and selective re-indexing will become increasingly important. The system may also need separate indexes or namespaces for tenants, departments, or knowledge domains. These choices should be evaluated against operational complexity and isolation requirements rather than treated as purely technical optimizations.

## XII. IMPLEMENTATION CONSIDERATIONS

The implementation should be organized around independently testable services. A FastAPI backend can expose authentication, tenant management, user management, content ingestion, knowledge search, chatbot, feedback, and analytics endpoints. Background workers can process uploaded documents asynchronously so that large files do not block interactive API requests. Processing states such as uploaded, extracting, chunking, embedding, indexed, failed, and archived can make the ingestion lifecycle observable. The ingestion pipeline should produce deterministic identifiers for source documents and chunks. A source document can be represented in SQLModel with fields for tenant, owner, title, type, version, status, object location, checksum, and timestamps. Each chunk can retain its source identifier and semantic metadata. The corresponding Pinecone record should use a stable vector identifier and the same tenant and source metadata, creating traceability from an AI answer back to the original artifact.

The query pipeline can be implemented as a sequence: authenticate request; resolve tenant and user permissions; normalize the query; apply metadata constraints; retrieve candidate chunks from Pinecone; optionally rerank; assemble context; invoke the Llama 3.1 model through Ollama; validate that the response is consistent with retrieved evidence; and return the answer with source references. This sequence makes authorization a prerequisite to retrieval rather than an afterthought. The frontend should expose role-specific workflows without duplicating backend authorization logic. React components can consume FastAPI endpoints through a typed API layer. Tailwind CSS can provide a consistent design system for dashboards, upload interfaces, search results, chatbot conversations, progress indicators, and administrative tables. The employee experience should make the knowledge source visible so that users can move from an AI answer to the underlying SOP, document, or video when additional detail is required. Observability should cover both traditional application metrics and AI-specific signals. API logs can record request duration and errors; ingestion logs can record processing failures; retrieval logs can record query latency and number of returned sources; and AI evaluation logs can record groundedness or reviewer feedback without storing unnecessary sensitive content. Monitoring should distinguish infrastructure failure from retrieval failure and model failure. The project should be developed with reproducibility in mind.

Configuration, prompt templates, embedding settings, chunking parameters, model versions, and retrieval filters should be version-controlled. A curated evaluation set should be maintained separately from production data. This makes it possible to compare changes to chunking, embeddings, retrieval, prompts, and models without relying on subjective demonstrations. Testing should follow the same boundaries as the architecture. Unit tests can validate extraction and chunking; integration tests can validate SQLModel, Pinecone, and R2 interactions; authorization tests can validate tenant isolation; retrieval tests can validate ranking behavior; and end-to-end tests can validate the complete question-to-answer path. This layered test strategy reduces debugging ambiguity because a poor chatbot response can be traced to source extraction, chunking, indexing, retrieval, prompt construction, or generation rather than being treated as one undifferentiated AI failure. API design should remain stable even when the underlying AI components change. For example, a chatbot endpoint should expose a consistent contract for question, conversation identifier, source references, response, and status even if the model changes from one Llama version to another. Similarly, the ingestion API should abstract the processing pipeline so that additional file types or transcription providers can be introduced without redesigning the frontend. Stable interfaces are particularly valuable in a project that is expected to grow from a final-year prototype toward an enterprise platform.

## CONCLUSION

LegacyAI is proposed as an AI-powered organizational knowledge hub that turns professional expertise into a durable and searchable organizational asset. The project addresses a practical knowledge-management problem: experienced employees may leave while valuable workflows, troubleshooting patterns, project lessons, and practical guidance remain incompletely documented. The synopsis and presentation therefore center on multimedia knowledge capture, centralized organization, employee learning, an AI chatbot, and a feedback-driven improvement loop [1], [2], [14],[10]. The review shows that the individual building blocks of LegacyAI are established technologies and concepts: organizational knowledge management provides the conceptual foundation; multimedia repositories preserve practical evidence; embeddings enable semantic retrieval; RAG connects external knowledge to generative models; and role-based application architecture provides a route to enterprise deployment. The project's contribution is primarily the integration of these components around the specific lifecycle of expert knowledge capture and organizational reuse. The updated technology stack provides a coherent implementation path. Python and FastAPI support the **service** layer; Ollama with Llama 3.1 and LlamaIndex provide the local LLM and orchestration layer; OpenAI text-embedding-3-small supports semantic indexing; PyMuPDF, pdfplumber, python-docx, and RecursiveCharacterTextSplitter support document processing; spaCy and scikit-learn provide NLP utilities;

Pinecone, SQLModel, and Cloudflare R2 separate semantic, transactional, and object storage; JWT supports authentication; and React.js with Tailwind CSS supports the user-facing portals. The central research opportunity is to validate whether this integration can reliably transform expert content into useful, secure, and grounded assistance. Evaluation should focus on ingestion quality, retrieval relevance, answer groundedness, knowledge freshness, onboarding effectiveness, latency, cost, and tenant isolation. Until such evaluation is completed, the benefits described by the project should remain expected outcomes rather than established performance claims. In its intended form, LegacyAI can become more than a repository. By preserving practical experience, connecting it to enterprise context, and making it accessible through evidence-grounded AI, the platform offers a path toward institutional memory that remains available after individual experts depart. Its future value will depend on disciplined knowledge governance, reliable retrieval, secure multi-tenancy, continuous feedback, and measurable validation.

## PROPOSED METHODOLOGY AND QUERY WORKFLOW

### Proposed Five-Phase LegacyAI Methodology

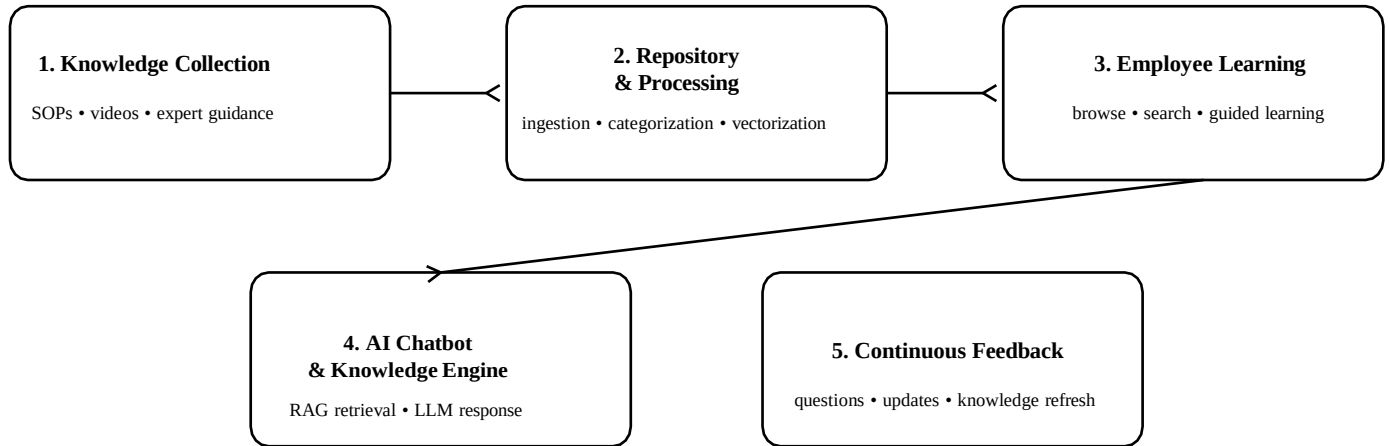


Fig. 11. Proposed five-phase methodology, adapted from the enhanced LegacyAI implementation methodology.

### Query and Retrieval Workflow

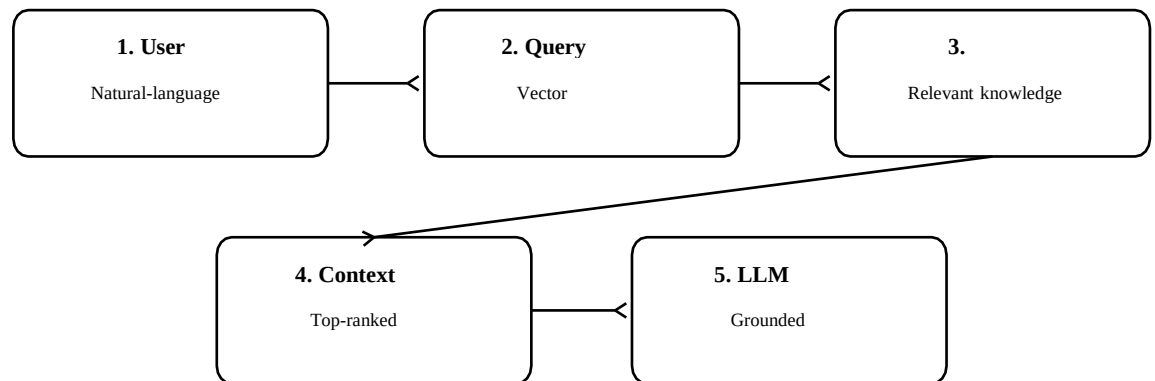


Fig. 12. Query-to-response workflow adapted from the enhanced RAG workflow.

The workflow follows the enhanced document's sequence: natural-language query, vector retrieval, context assembly from top-ranked evidence, and a grounded LLM response.

## REFERENCES

- [1] S. Sumbal and Q. Amber, “ChatGPT: A game changer for knowledge management in organizations,” *Kybernetes*, vol. 54, no. 6, pp. 3217–3237, 2025.
- [2] L.-C. Chen, M. S. Pardeshi, Y.-X. Liao, and K.-C. Pai, “Application of retrieval-augmented generation for interactive industrial knowledge management via a large language model,” *Computer Standards & Interfaces*, vol. 94, Art. no. 103995, 2025.
- [3] S. Es, J. James, L. Espinosa Anke, and S. Schockaert, “RAGAs: Automated evaluation of retrieval augmented generation,” in *Proc. 18th Conf. European Chapter of the Association for Computational Linguistics*, pp. 150–158, 2024.
- [4] J. Chen, H. Lin, X. Han, and L. Sun, “Benchmarking large language models in retrieval-augmented generation,” in *Proc. AAAI Conf. Artificial Intelligence*, vol. 38, no. 16, pp. 17754–17762, 2024.
- [5] J. Li, J. Chen, R. Ren, X. Cheng, X. Zhao, J.-Y. Nie, and J.-R. Wen, “The dawn after the dark: An empirical study on factuality hallucination in large language models,” in *Proc. 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 10879–10899, 2024.
- [6] H. Qian, Z. Liu, K. Mao, Y. Zhou, and Z. Dou, “Grounding language model with chunking-free in-context retrieval,” in *Proc. 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 1298–1311, 2024.
- [7] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, “Large language models for software engineering: A systematic literature review,” *ACM Trans. Software Engineering and Methodology*, vol. 33, no. 8, Art. no. 220, 2024.
- [8] G. Li, X. Zhou, and X. Zhao, “LLM for data management,” *Proc. VLDB Endowment*, vol. 17, no. 12, pp. 4213–4216, 2024.
- [9] The Llama Team, “The Llama 3 herd of models,” *arXiv preprint arXiv:2407.21783*, 2024.
- [10] S. Yin, C. Fu, S. Zhao, K. Li, X. Sun, T. Xu, and E. Chen, “A survey on multimodal large language models,” *National Science Review*, vol. 11, no. 12, Art. no. nwae403, 2024.
- [11] A. Kibriya, W. Z. Khan, A. Siddiga, and M. K. Khan, “Privacy issues in large language models: A survey,” *Computers & Electrical Engineering*, vol. 120, Art. no. 109698, 2024.
- [12] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” in *Proc. 40th International Conference on Machine Learning*, PMLR, vol. 202, pp. 28492–28518, 2023.
- [13] C. I. Nakpih, “A modified vector space model for semantic information retrieval,” *Natural Language Processing Journal*, vol. 8, Art. no. 100081, 2024.
- [14] R. Lenart, “Generative artificial intelligence and organizational learning: A systematic literature review,” *Journal of Organizational Change Management*, 2026.
- [15] A. Gan, H. Yu, K. Zhang, Q. Liu, W. Yan, Z. Huang, S. Tong, and G. Hu, “Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey,” *arXiv preprint arXiv:2504.14891*, 2025.