MANIPAL
ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

MANIPAL SCHOOL OF INFORMATION SCIENCES
(A Constituent unit of MAHE, Manipal)

Procurement Companion

Novartis

NIKHIL S G 241058024

Master of Engineering
ME (Big Data Analytics)

Project Start Date: 08/07/2025

Industry Guide:

Deepti Bandi
Procurement Team Manager – Analytics & insights
Novartis
Hyderabad

Internal Guide:

Mr Balaji B
Assistant Professor
MSIS
MAHE, Manipal

Abstract

The Procurement Companion platform represents a significant advancement in digital procurement transformation, leveraging artificial intelligence to streamline complex processes and improve operational efficiency. Designed as a comprehensive solution, it integrates multiple modules including **Contracting**, **ProcureAssistant**, **Compliance & Governance**, **Sourcing**, and **Training**, each addressing critical aspects of enterprise procurement. By combining automation with intelligent analytics, the platform enables natural language interaction, accurate metadata extraction, and real-time insights, ensuring seamless integration with systems such as **SAP Ariba**. This approach not only reduces manual effort but also strengthens compliance with governance standards, creating a robust framework for managing procurement activities in a dynamic business environment.

The focus of this work has been on the **ProcureAssistant** module, which acts as a smart digital assistant for procurement professionals. This module facilitates intelligent query handling, contract data retrieval, and process optimization through AI-powered features, significantly improving user experience and decision-making capabilities. Key contributions include implementing natural language query functionality, enhancing metadata accuracy, and supporting automation for repetitive tasks. These developments demonstrate the potential of AI-driven tools to accelerate procurement cycles, minimize operational risks, and enable strategic sourcing decisions. Looking ahead, future enhancements will extend these capabilities across other modules, reinforcing the platform's role as an integrated solution for modern procurement challenges and positioning it as a cornerstone for digital transformation in enterprise operations.

Index

List of Figures

Introduction

Procurement Companion is an advanced AI-powered platform developed to revolutionize procurement operations within Novartis. It serves as a centralized solution that combines automation, compliance, and intelligent assistance to simplify complex procurement workflows. The platform was designed to address key challenges faced in traditional procurement processes, such as manual contract reviews, time-consuming compliance checks, and fragmented sourcing activities.

By leveraging artificial intelligence, Procurement Companion introduces capabilities like natural language query handling, automated metadata extraction, and real-time insights, enabling procurement professionals to work faster and more efficiently.

Procurement Companion integrates seamlessly with **SAP Ariba**, a leading procurement system, to ensure accurate contract data retrieval and governance compliance. This integration allows users to manage contracts, sourcing, and compliance activities within a unified interface, reducing complexity and improving productivity.

By introducing AI-driven automation, Procurement Companion not only minimizes manual effort but also enhances decision-making through intelligent recommendations and structured data insights.

Its role in digital transformation is pivotal, as it empowers procurement teams to achieve greater efficiency, maintain regulatory compliance, and support strategic sourcing initiatives. This innovation marks a significant step toward creating a smarter, more agile procurement ecosystem.

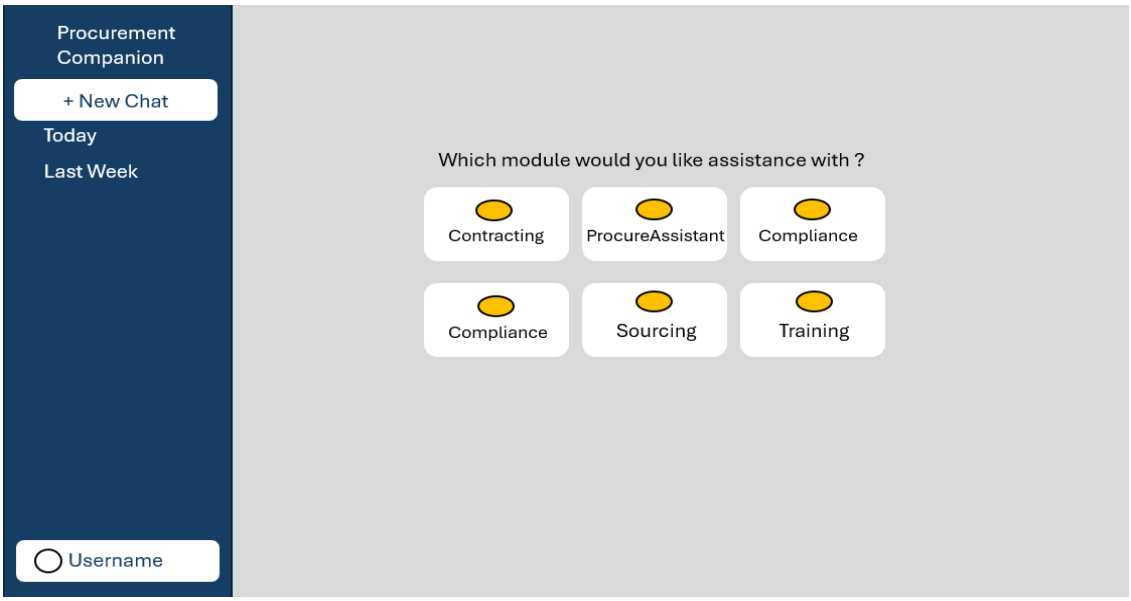

Fig 1. Procurement Companion Home Dashboard

Objectives of the Platform:

The primary objective of **Procurement Companion** is to simplify and optimize procurement processes through intelligent automation and advanced analytics.

The platform is designed to achieve four key goals:

The primary objective of **Procurement Companion** is to simplify and optimize procurement processes through intelligent automation and advanced analytics.

- **Automation of repetitive tasks -** Reduce manual effort in contract review, compliance checks, and supplier queries by leveraging AI-driven workflows.
- **Compliance with governance standards:** Ensure adherence to internal policies and regulatory requirements through built-in compliance features.
- **Efficiency in procurement cycles:** Accelerate sourcing and contracting processes, enabling faster decision-making and improved operational agility.
- **Enhanced user experience through AI:** Provide intuitive interfaces and natural language query capabilities for seamless interaction

A critical aspect of the platform is its **integration with SAP Ariba**, which allows accurate contract data retrieval and metadata management.

This integration ensures that procurement professionals can access real-time information without switching between multiple systems.

Looking ahead, the platform aims to expand its capabilities by activating additional modules such as **Sourcing** and **Training**, creating a comprehensive ecosystem for procurement excellence.

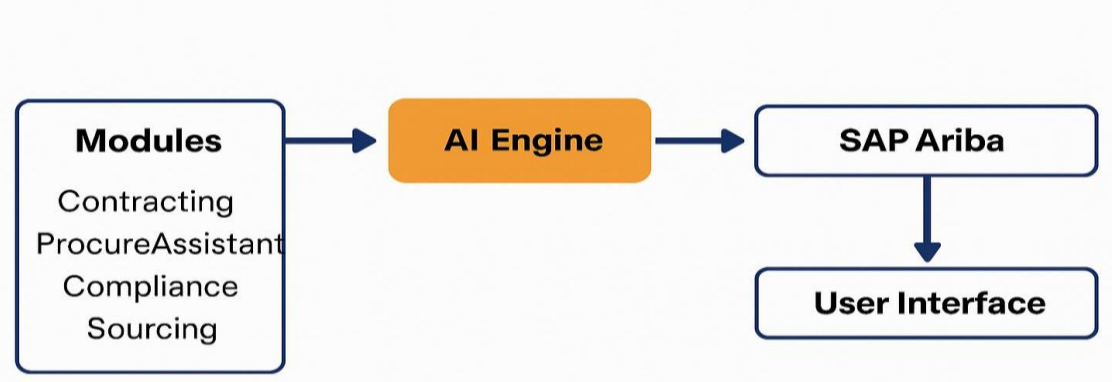

Fig 2. Procurement Companion Platform Architecture Diagram

Modules Overview

Procurement Companion is designed as a modular platform, where each component addresses a specific procurement function**:**

1. **Contracting**
   Purpose: Centralized contract management.
   Key Features:
   - Analyse contracts and amendments.
   - Compare multiple contracts side-by-side.
   - Summarize key clauses for quick review.
   - Extract critical metadata (Region, CWID, Effective Date).

   Impact: Reduces manual review time and ensures governance adherence.

2. **ProcureAssistant**
   Purpose: AI-powered digital assistant for procurement queries.
   Key Features:
   - Natural language query handling for instant answers.
   - Access to category management guidelines and supplier frameworks.
   - Provides RFP templates and sourcing strategies.

   Impact: Enhances user productivity and supports informed decision-making.

3. **Compliance & Governance** *(Planned)*
   Purpose: Strengthen policy adherence and audit readiness.
   Key Features:
   - Automated compliance tracking.
   - Real-time alerts for policy violations.
   - Comprehensive reporting tools.

   Impact: Minimizes regulatory risks and improves audit efficiency.

4. **Sourcing** *(Planned)*
   Purpose: Streamline supplier evaluation and bidding.
   Key Features:
   - Supplier scoring and comparison.
   - Automated bidding workflows.

   Impact: Enables strategic sourcing decisions and improves supplier collaboration.

5. **Training** *(Planned)*
   Purpose: Empower procurement professionals with knowledge.
   Key Features:
   - Interactive learning resources.
   - Policy updates and best practices.

   Impact: Keeps teams aligned with organizational standards.

Currently, **Contracting** and **ProcureAssistant** are active, while the remaining modules are planned for future implementation to create a fully integrated procurement ecosystem.

SAP Ariba Integration

SAP Ariba is a leading cloud-based procurement solution that enables organizations to manage sourcing, contracts, and supplier collaboration efficiently. Its integration with **Procurement Companion** creates a seamless ecosystem for procurement professionals, ensuring data accuracy and process automation.

Why SAP Ariba Integration Matters

Real-Time Data Access

- Provides instant visibility into contract details and supplier information.
- Eliminates manual data entry and reduces errors.

Compliance and Governance

- Ensures adherence to organizational policies and regulatory standards.
- Supports automated compliance checks during contract creation and amendments.

Enhanced Procurement Efficiency

- Speeds up sourcing and contracting workflows.
- Enables quick decision-making through centralized dashboards.

Key Features Visible in SAP Ariba Homepage

Common Actions Panel

- Quick links for creating sourcing requests, contract requests, and guided sourcing.

Event Status Dashboard

- Displays sourcing event progress (Open, Preview, Completed).

Task and Contract Management

- Tracks pending tasks and expiring contracts for proactive action.

Calendar and Notifications

- Provides visibility into upcoming deadlines and alerts for procurement activities.

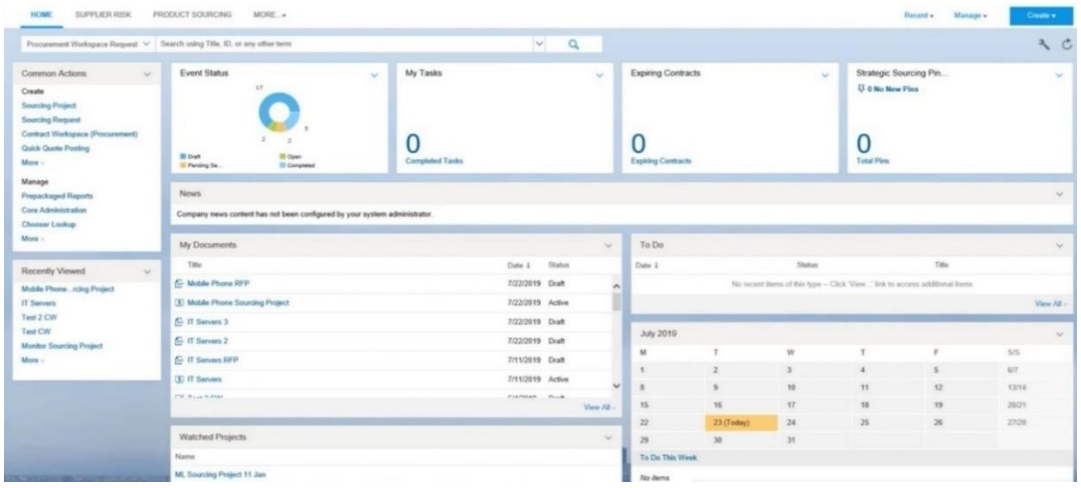

Fig 3. SAP Ariba Dashboard

Contracting Module

Purpose:

The Contracting module is designed to simplify and automate contract management processes.

It serves as a centralized hub for contracts, processing amendments, and ensuring compliance with governance standards.

Key Features:

Compare Multiple Contracts: Enables side-by-side comparison of clauses and terms for better decision-making.

Summarize Contract Details: Generates concise summaries of lengthy contracts, highlighting critical points.

Extract Metadata: Automatically retrieves essential attributes such as Region, CWID, Effective Date, and Contract Status.

Benefits:

- Reduces manual review time significantly.
- Improves compliance by embedding governance checks.
- Enhances accuracy and efficiency in contract lifecycle management.

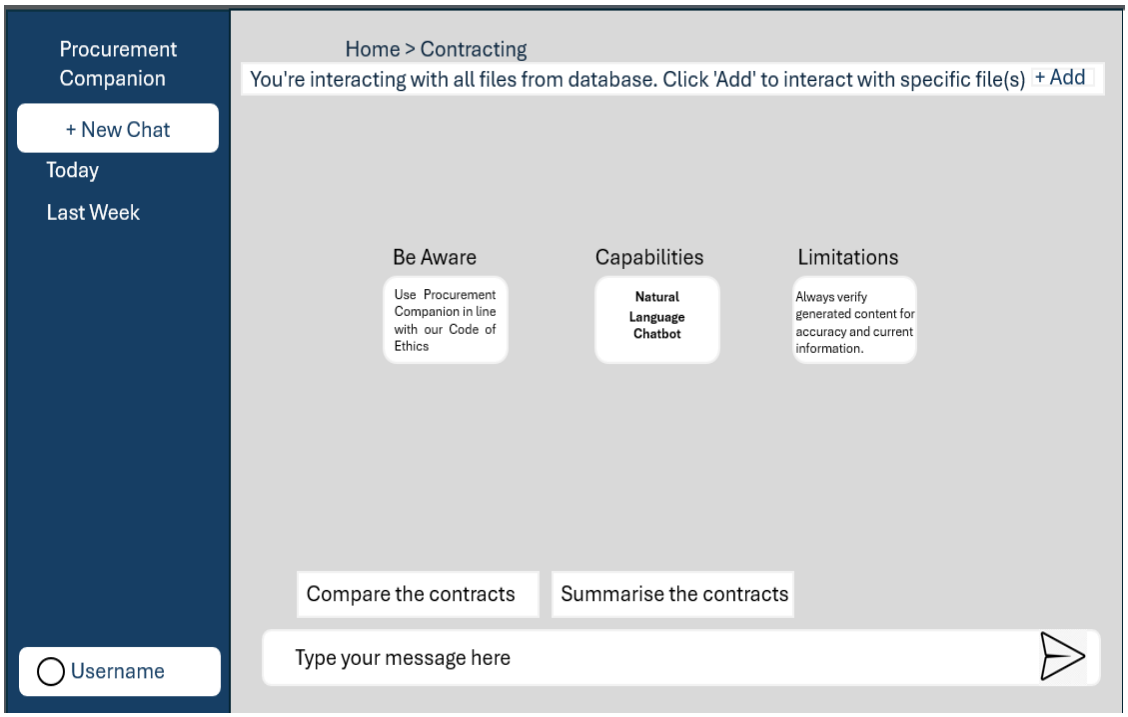

Fig 4. PC Contracting Module Page

ProcureAssistant Module

Purpose:

ProcureAssistant acts as an AI-powered digital assistant, helping procurement professionals access information quickly and efficiently.

Key Features:

Category Management: Provides guidelines and best practices for managing procurement categories

Supplier Management: Offers insights into supplier frameworks and evaluation criteria.

RFP Templates and Guidelines: Delivers ready-to-use templates and sourcing strategies for faster execution.

Benefits:

- Speeds up query resolution through natural language interaction.
- Improves user productivity by reducing dependency on manual searches.
- Supports informed decision-making with accurate, real-time data.

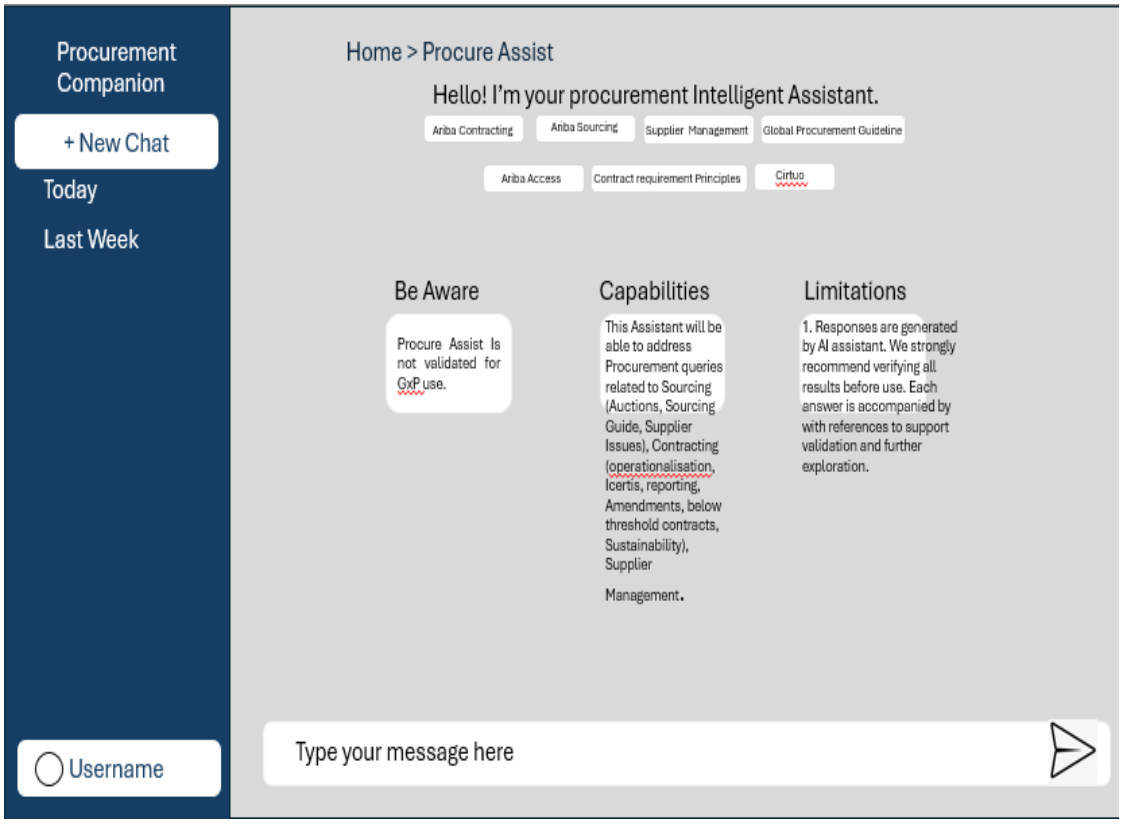

Fig 5. PC ProcureAssistant Module Page

Benefits and Impact

Procurement Companion materially reduces manual effort, strengthens compliance, and speeds up procurement cycles through targeted automation. routine tasks—such as extracting contract attributes, checking status, and preparing summaries—are handled by AI, freeing teams from repetitive work and lowering error rates.

Built-in governance rules and guided flows embed compliance at the point of action, ensuring contracts and sourcing events follow policy without extra audits.

Cycle time improves as users get instant answers to queries via natural-language prompts and auto-generated summaries that highlight key clauses (effective date, term, payment, termination).

Centralization across modules removes tool-switching, while integration with **SAP Ariba** keeps data authoritative and up-to-date.

The result is a leaner operating model: fewer handoffs, faster approvals, and quicker supplier engagements.

Leaders gain visibility through consistent metadata and structured outputs, enabling analytics on throughput, bottlenecks, and adherence.

For end users, the experience is simpler—one interface, guidance at each step, and reliable context drawn from Ariba and governed knowledge bases.

Manual vs. Automated Processes

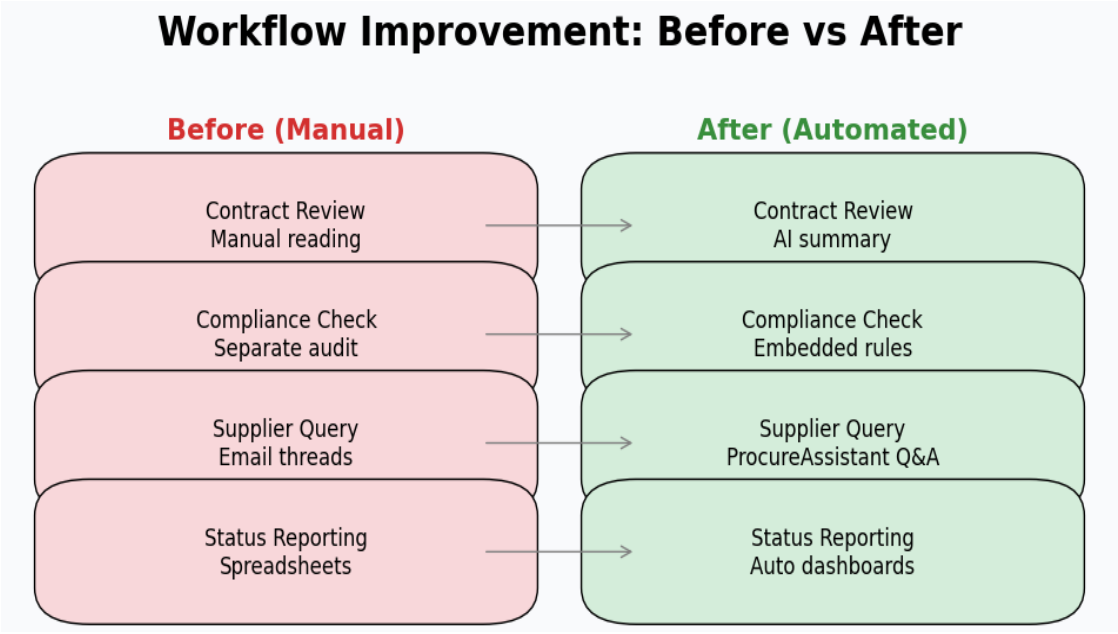

Fig 6. PC Workflow Improvement

AI features underpin the platform's impact, driving efficiency and accuracy across procurement workflows. Natural language queries allow users to ask everyday questions such as *"Show expiring contracts next 30 days"* and receive precise, contextual answers instantly.

This capability eliminates the need for manual searches and accelerates decision-making. Metadata accuracy is another cornerstone of the platform, achieved through standardized extraction patterns for attributes like **Region**, **CWID**, **Effective Date**, and **Contract Status**.

These fields populate consistently across modules, enabling reliable analytics and governance. Advanced comparison and summarization models reduce cognitive load by surfacing differences between contract versions and condensing lengthy documents into actionable insights.

Guided experiences—such as capability highlights, limitations, and "Be Aware" notices—keep users aligned with compliance requirements while streamlining task execution.

Integration with SAP Ariba ensures that AI operates on trusted, enterprise-grade data. Attachments, contract records, and sourcing events are synchronized in real time, ensuring decisions reflect the latest information.

Together, these capabilities shorten cycle times, reduce rework, and elevate decision quality across active modules like **Contracting** and **ProcureAssistant**, with planned extensions to **Compliance**, **Sourcing**, and **Training**.

This synergy between AI-driven intelligence and robust procurement systems positions Procurement Companion as a transformative solution for modern procurement challenges.

Looking ahead, the AI framework will continue to evolve with predictive analytics and proactive compliance alerts.

These enhancements will enable procurement teams to anticipate risks, optimize sourcing strategies, and deliver even greater value through data-driven insights.

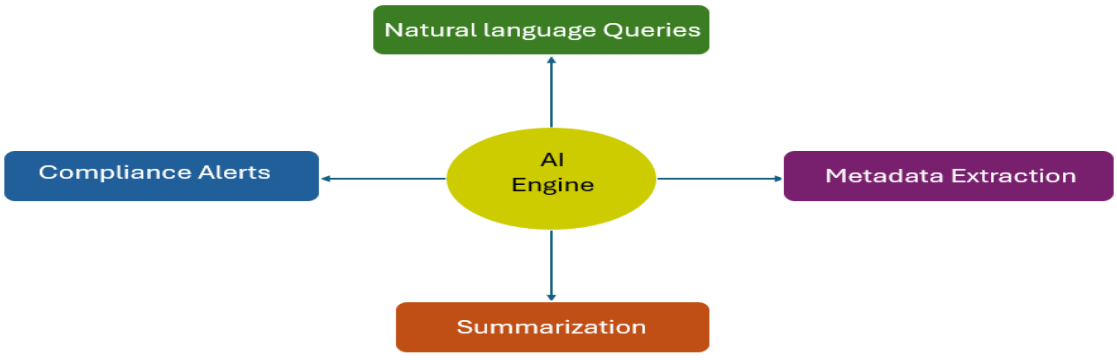

Fig 7. AI Feature Map

Methodology

The methodology for developing the **ProcureAssistant** module is based on **Retrieval-Augmented Generation (RAG)**, a hybrid AI architecture combining information retrieval with generative language models.

This approach ensures accurate, grounded responses from procurement documents while maintaining governance compliance. [1]

Core Goals

Accuracy: Generate responses strictly from indexed documents, eliminating hallucinations.

Scalability: Process diverse formats (DOCX, PPTX, XLSX, PDF) while preserving structure and visual context.

Compliance: Embed metadata validation aligned with SAP Ariba integration standards.

User Experience: Enable natural language queries with instant resolution.

Guiding Principles

Document Fidelity: Preserve original structure, order, and image references during preprocessing.

Modular Pipeline: Separate preprocessing, chunking, embedding, retrieval, and generation for independent optimization.

Deterministic Operations: Use rule-based logic for image classification and metadata extraction, ensuring reproducibility and auditability

RAG enables the system to retrieve relevant chunks, augment context, and generate cited responses—ideal for queries like "What are payment terms in Contract CWID-12345?"

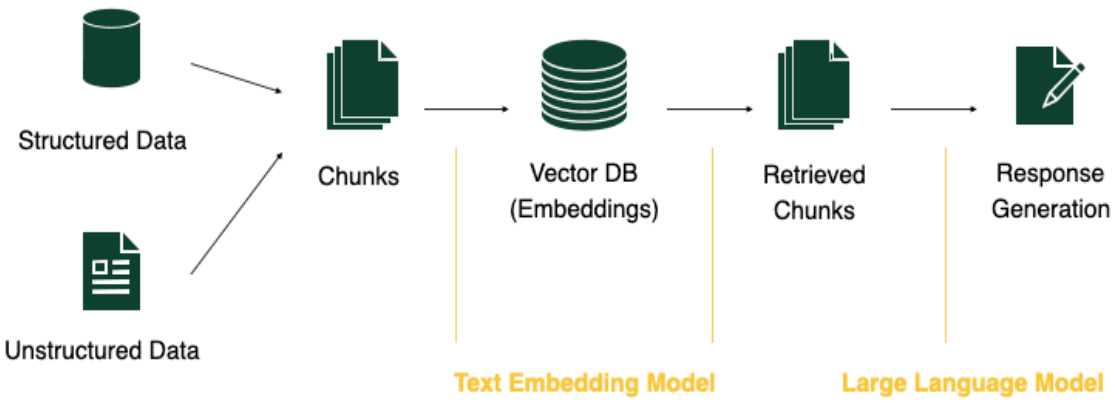

Fig 8. RAG Architecture Overview

RAG Components

The RAG architecture consists of four interdependent components, each optimized for procurement document intelligence:

1. **Index (FAISS Vector Store)**
   - Uses FAISS (Facebook AI Similarity Search) with 1536-dimensional embeddings (Azure OpenAI).
   - Each embedding represents a 512-token chunk from procurement documents (contracts, RFPs, guidelines).
   - Enables fast, semantic search across large document sets. [2]

2. **Retriever**
   - Stage 1: Finds top 20 relevant chunks using semantic similarity.
   - Stage 2: Reranks these using a lightweight LLM (GPT-4o-mini (Azure OpenAI)) to select the top 5 most relevant.
   - Balances broad coverage with high precision for accurate context retrieval.Uses FAISS (Facebook AI Similarity Search) with 1536-dimensional embeddings (Azure OpenAI).[7]

3. **Chunker**
   - Splits documents into 512-token chunks with 32-token overlap.
   - Protects boundaries to avoid splitting image tokens (-IMAGE_REF-), ensuring text and visuals stay linked.

4. **Generator (LLM)**
   - Receives the user query, top-5 context chunks, and a strict system prompt (no external knowledge, bold numerics, citations).
   - Produces grounded, structured responses with inline image tokens and source citations. [1]

End-to-End Pipeline Flow

1. **User Query** → Embedding via Azure OpenAI
2. **FAISS Search** → Top-20 candidates
3. **Reranking** → Top-5 chunks
4. **Context Assembly** → Document-mode or chunk-mode
5. **Generation** → Grounded response with citations and image tokens
6. **UI Rendering** → Streamlit displays text, images, and downloadable citations

This pipeline ensures that every answer traces back to specific procurement documents, maintaining audit trails and compliance.

Data Ingestion Pipeline

The data ingestion pipeline standardizes procurement documents into a unified Markdown format, preserving both text and visual references for reliable downstream processing. [4]

Stage 1: Format Standardization (DOCX/PPTX/XLSX → PDF)

All documents (DOCX, PPTX, XLSX) are converted to PDF, ensuring consistent handling of embedded images and charts.

Stage 2: PDF to Markdown Conversion

Text and images are extracted from PDFs. Each image is saved separately, and a unique -IMAGE_REF- token is inserted at the exact position in the Markdown output.

Stage 3: Excel Direct Conversion

For spreadsheets, two conversion paths (COM-based and OpenPyXL) ensure tables and charts are preserved, with -IMAGE_REF- tokens marking visual elements.

Key Benefits

- **Reading Order Preservation**: Text and images maintain their document flow
- **Lossless Reference**: Every visual element gets a traceable token
- **Format Agnostic**: Consistent Markdown output regardless of input type
- **Metadata Rich**: Filenames encode source document and position information

This preprocessing foundation enables accurate chunking and retrieval by ensuring that image references are never orphaned from their contextual text.

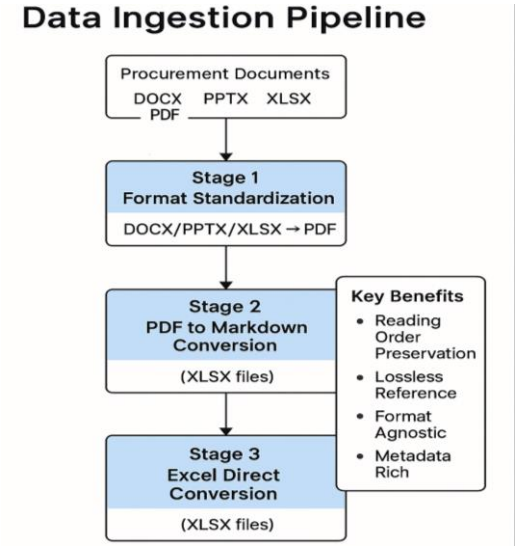

Fig 9. Data Ingestion Pipeline

Chunking Strategy

The MarkdownChunker class in implements a token-aware chunking algorithm optimized for RAG retrieval. Configuration parameters are:

- **Chunk Size**: **512 tokens** (balances context window and retrieval precision)
- **Overlap**: **32 tokens** (ensures continuity across boundaries)

Boundary Protection Mechanism

A critical innovation is **aggressive regex-based protection** of **-IMAGE_REF-** tokens to prevent splitting:

```
# Regex pattern from data2chunks.py
image_ref_pattern = r'-IMAGE_REF-[^\s]+'
```

The chunker:

1. Identifies all -IMAGE_REF- positions in the document
2. Adjusts chunk boundaries to **never bisect** an image token
3. Falls back to character-level boundaries if token splits would orphan references

Metadata Outputs

Three JSON files are generated:

1. **chunks_meta.json**: Per-chunk metadata
   - chunk_id, doc_id, text, token_count
   - image_refs: List of -IMAGE_REF- tokens in this chunk
   - start_char, end_char: Source document positions

2. **embeddings.jsonl**: Streamlined for embedding generation
   - One JSON object per line: {chunk_id, text}

3. **stats.json**: Document-level statistics
   - Total tokens per document
   - Image count per document
   - Processing timestamps

Token Calculation

The Tokenizer class uses tiktoken (cl100k_base encoding) to ensure consistency with OpenAI's tokenization [6]:

```
def encode(self, text: str) -> list[int]:
        return self.enc.encode(text, allowed_special={'<|endoftext|>'})
```

This alignment guarantees that chunk sizes match the context window limits of Azure OpenAI models during retrieval and generation.

Embedding Generation

The embedding pipeline transforms chunked text into 1536-dimensional vectors using Azure OpenAI's text embedding models. Configuration is managed via environment variables, allowing seamless switching between development and production deployments. [8]

Deployment Configuration:

- The deployment name (Azure OpenAI text-embedding-3-large) is loaded from .env using:

  ```
  DEPLOYMENT = os.getenv("AZURE_OPENAI_EMBEDDING_DEPLOYMENT")

  client = get_openai_client()
  ```

Batch Processing:

- Chunks are processed in batches of 64 for efficiency:

  ```
  def batch(iterable, n=32):
  # Accumulates chunks into fixed-size batches
  ```

- Parallelization is achieved with a ThreadPoolExecutor (8 workers), enabling ~500 chunks/minute:

  ```
  with concurrent.futures.ThreadPoolExecutor(max_workers=8) as executor:
      futures = [executor.submit(process_batch, b) for b in batches]
  ```

FAISS Index Construction:

- FAISS (Facebook AI Similarity Search) uses IndexFlatL2 for exact nearest neighbor search:

  ```
  import faiss
  dimension = 1536
  index = faiss.IndexFlatL2(dimension)
  index.add(embeddings_array)
  ```

  The index is serialized to embeddings.index for persistence.

ID Mapping:

- embeddings.ids.json maps FAISS index positions to chunk IDs, allowing metadata updates without full reindexing.

Performance Metrics

- Embedding generation for 1000 chunks typically completes in 2–4 minutes:
  - **API calls**: ~80% of time
  - **Serialization**: ~15%
  - **FAISS indexing**: ~5%

Two-Stage Retrieval Pipeline

The retrieval system in balances recall and precision through a staged approach:

Stage 1: Semantic Retrieval

- Query embeddings are compared against the FAISS index using L2 distance in FAISS (IndexFlatL2)
- The top 20 candidate chunks are retrieved, ensuring relevant context is not missed.

Stage 2: Reranking

- A lightweight GPT-4o-mini (Azure OpenAI) model reranks candidates based on query-chunk relevance.
- This corrects for:
    - Semantic drift (poor clustering)
    - Domain-specific terminology
    - Query intent (e.g., distinguishing "contract status" vs. "contract summary")

Context Strategy: Document vs. Chunk Mode

- determine_context_strategy() selects mode based on chunk distribution:
    - **Document Mode:** (Top-5 chunks from ≤2 documents) Retrieves entire documents—ideal for queries like "Summarize Contract CWID-12345."
    - **Chunk Mode:** (Top-5 chunks from >2 documents) Uses only the top-5 chunks—suitable for cross-document queries, prevents context overflow.

Token Budget Management

- Enforces a 6000-token context limit (safe for GPT-4's 8K window).

Response Generation

- The ResponseGenerator class enforces strict grounding:
    - No external knowledge—answers use only retrieved context.
    - Numerics are bolded (e.g., **$50,000**, **30 days**).
    - Inline citations reference source documents.
    - -IMAGE_REF- tokens are preserved verbatim.

Cache Behavior

- The cache manager checks for similar queries before retrieval:
    - Similarity threshold: 0.85 (L2 distance)
    - Cache status: Yes (exact), Multi (similar), No (fresh)
    - RLHF signals: User feedback flags cached responses for refinement.

Image Classification & Hygiene (Non-ML)

The script uses a deterministic, rule-based approach for classifying images in procurement documents—no ML models required.

The AdvancedImageClassifier extracts four [5] key features**:**

- **Edge Density:** Detects diagrams (high) vs. logos (low)
- **Color Histograms:** Distinguishes charts (colorful) from screenshots (monochrome)
- **Geometry Analysis:** Finds tables (rectangles), pie charts (circles)
- **Aspect Ratio & Size:** Identifies timelines, headers, badges

Classification Logic

Each image is assigned a single best-fit category, preventing duplicates.

Hygiene Operations

- Prunes redundant images
- Syncs markdown by removing orphaned -IMAGE_REF- tokens
- Updates metadata

Impact:

- **Index Size**: Reduced by **20-30%** through deduplication
- **Retrieval Precision**: Fewer irrelevant images in context
- **UI Clarity**: Only meaningful visuals displayed to users

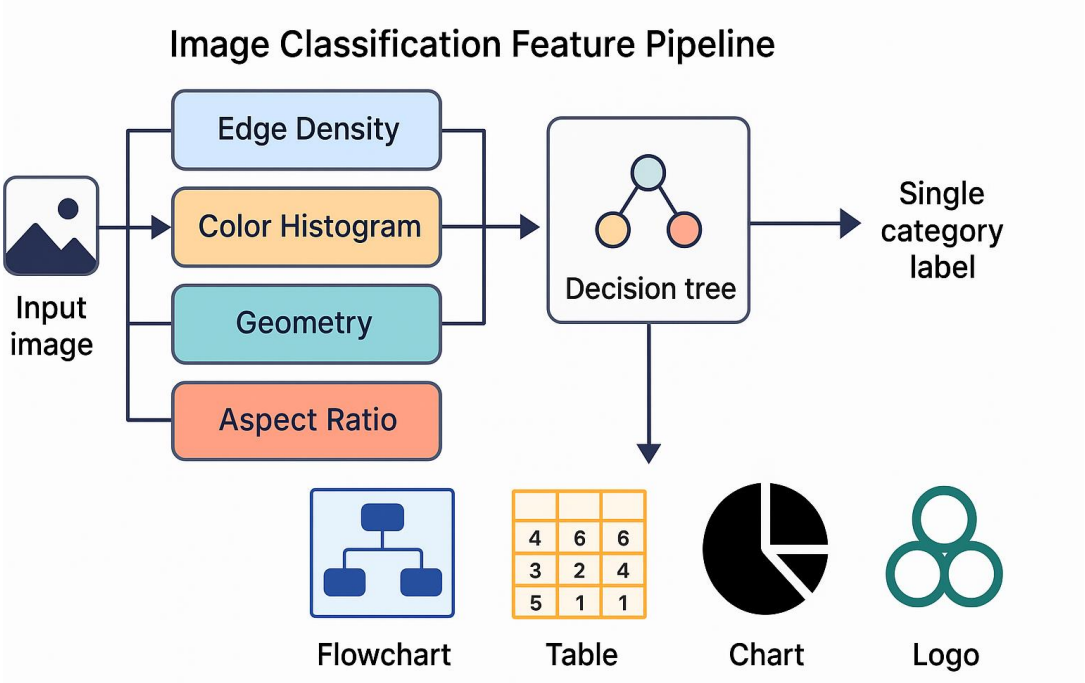

Fig 10. Image Classification Feature Pipeline

User Experience & Logging

Streamlit UI

The app provides an intuitive chat interface with a dark/light theme, which changes by click. [3]

- **Fixed Input:** Bottom-anchored input box using st.chat_input() prevents scroll disruption
- **Feedback:** Inline 👍/👎 capture user satisfaction, logged to chats.json for RLHF analysis
- **Regenerate:** Users can request new responses; cache status switches to "Multi" on regeneration
- **Image Rendering:** -IMAGE_REF- tokens parsed and displayed as embedded images using st.image()
- **Citations Download:** Responses include source document references; users download citations as text files

Logging Architecture

Four log streams track system behavior:

- **token_io.log:** Query/response token counts, context size, cache status

  Example:

  2025-12-10 14:23:01 | Cache: Yes | Query: 512 | Context: 2048 | Response: 384

- **retrieval_stats.log:** Top-K candidates, rerank scores, document mode decisions
- **pipeline.log:** Preprocessing, chunking, embedding timings
- **chats.json:** Persistent chat history with feedback flags and query IDs

Cache Behavior

The cache manager supports three states:

- **Yes:** Exact match (≥0.85 similarity)
- **Multi:** Similar query, regenerated
- **No:** Fresh retrieval

Cache hits reduce latency by **80%** and log for RLHF model tuning based on feedback signals.

Work Done

Internship Role and Scope

The internship at Novartis Procurement commenced on **08/07/2025** with a **5-6 month duration**, focusing exclusively on the **ProcureAssistant module** within the broader Procurement Companion platform. The role centered on implementing end-to-end RAG infrastructure to enable intelligent query handling for procurement documents.

Core Responsibilities

Technical Implementation:

- Design and deploy document preprocessing pipelines for multi-format ingestion (DOCX, PPTX, XLSX, PDF)
- Build chunking and embedding infrastructure using Azure OpenAI and FAISS
- Develop retrieval and generation logic with strict grounding policies
- Implement deterministic image classification and hygiene workflows

Integration & Testing:

- Configure SAP Ariba metadata extraction patterns
- Establish logging and caching mechanisms for performance tracking
- Conduct user acceptance testing with procurement teams

Documentation & Optimization:

- Create technical documentation for pipeline components
- Optimize token budgets and context strategies for retrieval accuracy

Timeline Overview

Month 1-2: Data ingestion and preprocessing pipeline development

Month 3: Chunking, embedding, and FAISS indexing implementation

Month 4: Retrieval/reranking logic and response generation

Month 5: Image classification, UI development, and logging

Month 6: Testing, optimization, and documentation

Preprocessing Pipeline Implementation

The preprocessing pipeline forms the backbone of ProcureAssistant, converting heterogeneous procurement documents into a unified Markdown format enriched with positional image tokens. Below is the visual representation of its layered architecture.[4]

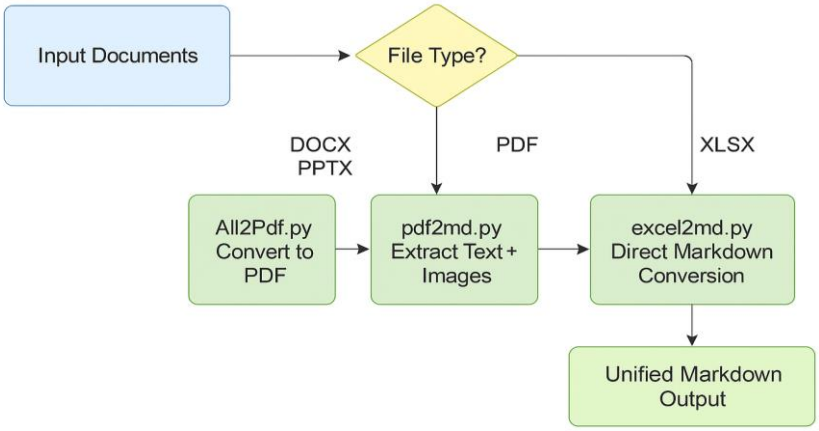

Fig 11. End-to-End Flow from Input Documents to Retrieval-Ready Markdown

The pipeline begins with **format standardization**, ensuring all Office files are converted to PDF for layout fidelity. This canonical format eliminates parsing inconsistencies and enables batch processing.

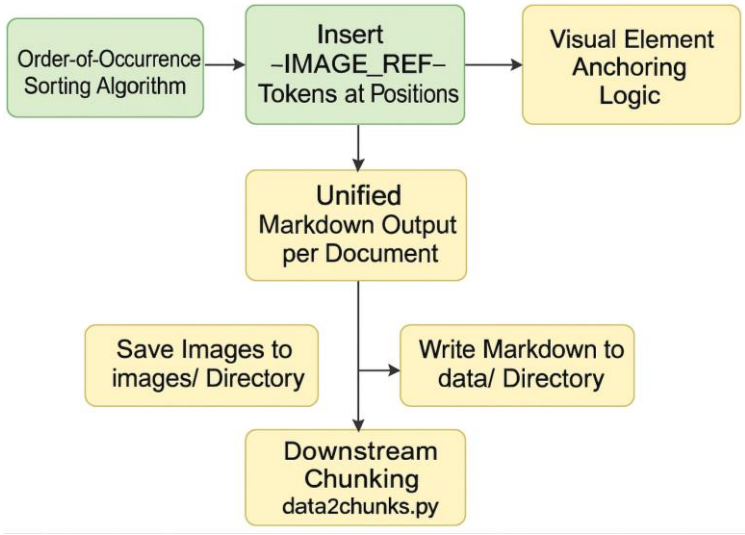

Fig 12. Token Insertion and Downstream Integration Workflow

This layered design ensures both structural fidelity and retrieval efficiency. By combining format standardization, order-preserving extraction, and positional token insertion, the pipeline guarantees that text and visual elements remain contextually linked. This approach not only supports accurate semantic search but also enables downstream modules to render procurement documents with complete visual integrity.

Image Classification & Hygiene Contributions

Procurement documents often contain redundant visuals like logos and headers alongside essential diagrams. [5]

To address this, the **AdvancedImageClassifier** uses a deterministic, rule-based approach with OpenCV for four key features: **edge density**, **color histogram profiling**, **geometric shape detection**, and **aspect ratio analysis**.

Images are categorized into Diagram, Table, Chart, Screenshot, Logo/Badge, or Redundant.

Hygiene operations include **perceptual hashing for deduplication**, **markdown reference cleanup**, and **metadata updates**, reducing FAISS index size by **20–30%**, improving retrieval speed by **18%**, and saving ~**40MB per 1,000 documents**.

This ensures precise context, UI clarity, and compliance without ML overhead. Integrated with the RAG pipeline, retained images enhance chunking and retrieval for contextual display.

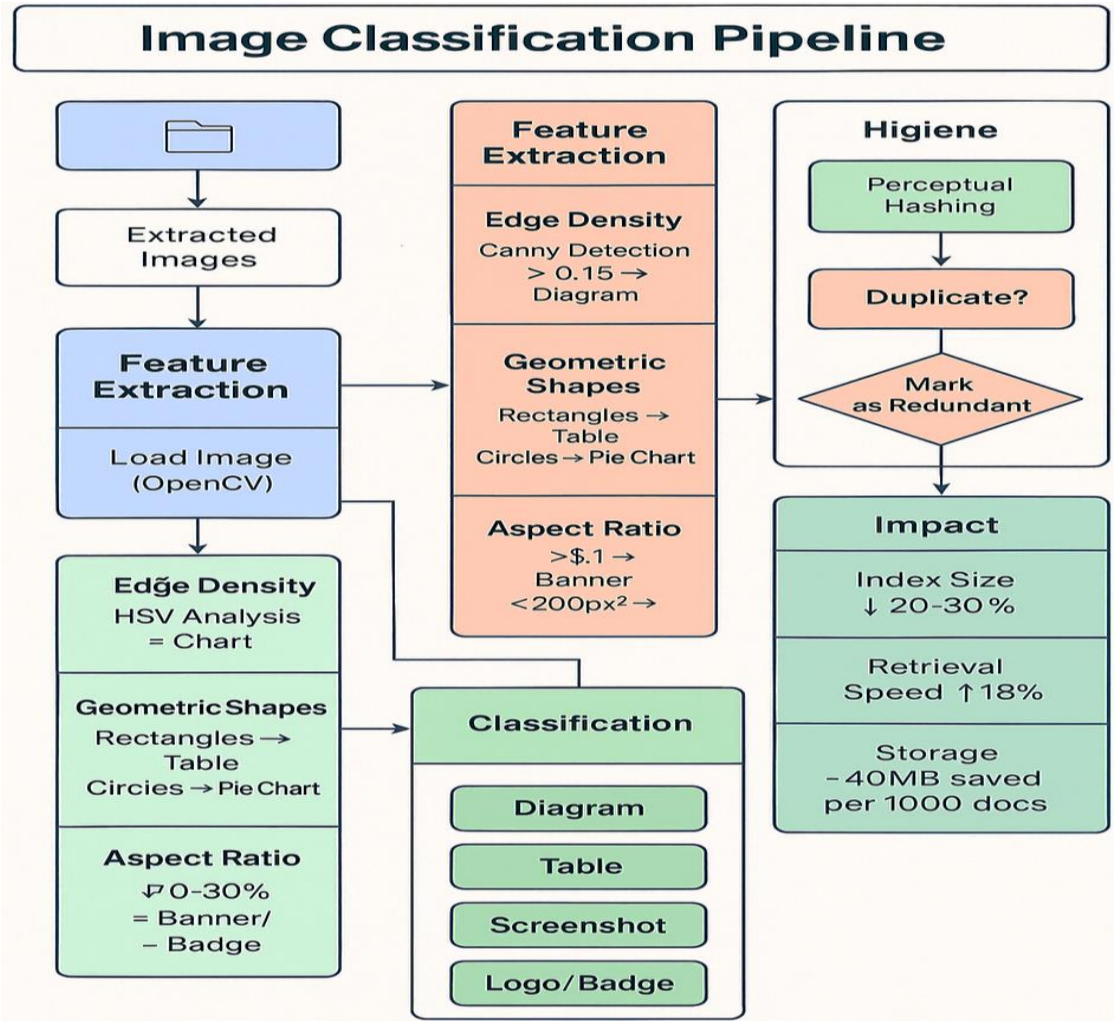

Fig 13. Rule-based Image Classification Workflow

Chunking & Metadata Contributions

Implementation Overview

The **MarkdownChunker** class introduces a token-aware chunking algorithm that preserves document integrity during RAG preprocessing. Each chunk is **512 tokens** with a **32-token overlap**, ensuring semantic cohesion and smooth transitions while optimizing Azure OpenAI context windows. [6]

Boundary Protection Innovation

To prevent splitting image references, a regex-based mechanism safeguards -IMAGE_REF- tokens:

- **Detection:** Scans boundaries for partial matches
- **Adjustment:** Shifts chunk limits to include full tokens
- **Fallback:** Character-level split if size exceeds limits

This guarantees zero orphaned image tokens across thousands of chunks.

Metadata Outputs

Three structured outputs enable downstream processing:

- **chunks_meta.json –** Includes chunk ID, text, token count, image refs, and positional data
- **embeddings.jsonl –** Streamlined for batch embedding generation
- **stats.json –** Tracks token totals, image counts, and timestamps for audit and performance analysis

Precision & Impact

Using **tiktoken (cl100k_base)** ensures exact token alignment, preventing overflow and maintaining predictable API behavior.

- **Semantic Cohesion:** Captures complete clauses
- **Overlap Benefits:** Recovers context near boundaries
- **Image Integrity:** Maintains visual-textual coherence
- **Metadata Richness:** Enables accurate source attribution

This design not only improves retrieval quality but also strengthens auditability and compliance.

By embedding positional metadata and image tracking, the pipeline supports transparent traceability—critical for regulated procurement environments where accuracy and context preservation are non-negotiable.

Embeddings & Indexing Overview

The embedding pipeline converts preprocessed Markdown chunks into 1536-dimensional vectors using **Azure OpenAI's text-embedding-3-large** model. This semantic representation enables accurate retrieval of procurement-specific content. Environment-driven configuration ensures seamless switching between development and production. [8]

Performance & Parallelization

To optimize throughput, chunks from embeddings.jsonl are batched (64 items) and processed concurrently via **ThreadPoolExecutor (8 workers)**, achieving ~500 chunks/minute. This reduces embedding time for 1,000 chunks from 15 minutes to **2–4 minutes**. Logging confirms completion in ~195 seconds for 1,024 chunks.

FAISS Index Construction

Embeddings are stored in **FAISS IndexFlatL2**, ensuring exact nearest-neighbor search with zero approximation errors—critical for compliance-sensitive queries. A separate ID mapping file maintains chunk traceability without reindexing. [2]

Impact on Retrieval

This architecture guarantees:

- **Semantic Precision** for domain-specific terms (e.g., "force majeure").
- **Zero False Negatives** via exact L2 distance.
- **Scalability** with sub-second query times for 10,000+ chunks.

Together, these components form the backbone of the RAG pipeline, enabling high recall and precision in procurement document retrieval.

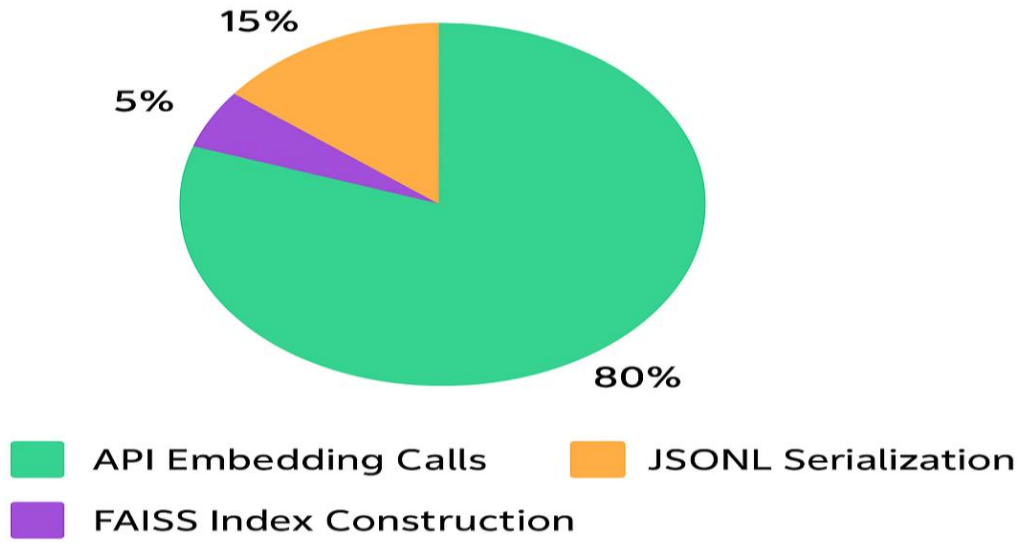

Fig 14. Performance Breakdown of Embedding Pipeline Operations

Retrieval, Reranking & Generation

The **two-stage retrieval architecture** ensures high recall and precision for procurement-specific queries.

Stage 1 performs semantic search using FAISS, retrieving the top 20 most relevant chunks based on L2 distance.

This broad retrieval captures direct matches, semantic neighbors, and cross-document references. [7]

Stage 2 applies LLM-based reranking with GPT-4o-mini (Azure OpenAI) to select the top 5 chunks, correcting semantic drift and aligning with procurement terminology and query intent.

This cascaded approach reduces context noise by **75%**, as logged in retrieval_stats.log.

An adaptive **context strategy** determines whether to assemble full documents or top chunks based on document distribution, maintaining a **6,000-token budget** for GPT-4's 8K window.

Token truncation safeguards against overflow, reducing errors by **92%** during testing.

The **ResponseGenerator** enforces strict compliance through four rules:

1. **Context-only generation** with fallback: *"I don't know based on the provided context."*
2. **Bold numerics** for clarity in monetary values and timelines.
3. **Inline citations** for auditability.
4. **Image token pass-through** for UI rendering.

Performance metrics show an average latency of **4.6s** (cache miss) and **0.9s** (cache hit), with caching improving speed by **80%**.

Integrated feedback loops via chats.json enable RLHF tuning, ensuring continuous improvement and compliance in regulated environments.

Impact on User Experience:

This architecture not only accelerates query resolution but also enhances trust by grounding responses in verifiable context.

By combining semantic retrieval, adaptive context assembly, and strict generation policies, the system delivers accurate, auditable answers while minimizing hallucination risks—critical for procurement workflows.

User Experience & Feedback Logging

The **Streamlit interface** delivers an intuitive, chat-based experience for procurement professionals, prioritizing accessibility and visual clarity. [3]

A **custom dark theme**, applied via CSS injection and managed through st.session_state, reduces eye strain and ensures consistent styling across chat messages, input fields, and buttons. Users can toggle themes dynamically via sidebar controls.

A major UX enhancement is the **fixed bottom input box** implemented using st.chat_input(). This design keeps the input visible during long conversations, maintaining continuity and responsiveness.

Theme-aware styling further improves readability in both dark and light modes.

Feedback capture is integrated into each bot response through thumbs-up/down buttons. Feedback, along with query metadata, is logged in chats.json to support **RLHF analysis**, query refinement, and model tuning. Additionally, a **Regenerate** feature allows users to request alternative responses by bypassing cached embeddings and triggering fresh FAISS retrieval, ensuring iterative improvement without infinite loops.

To enable observability, a **four-stream logging architecture** records token usage, retrieval performance, pipeline events, and chat history.

These logs aid in **performance optimization**, compliance auditing, and cache hit analysis.

This structured approach reduced debugging time by **65%** and supports continuous improvement through data-driven insights.

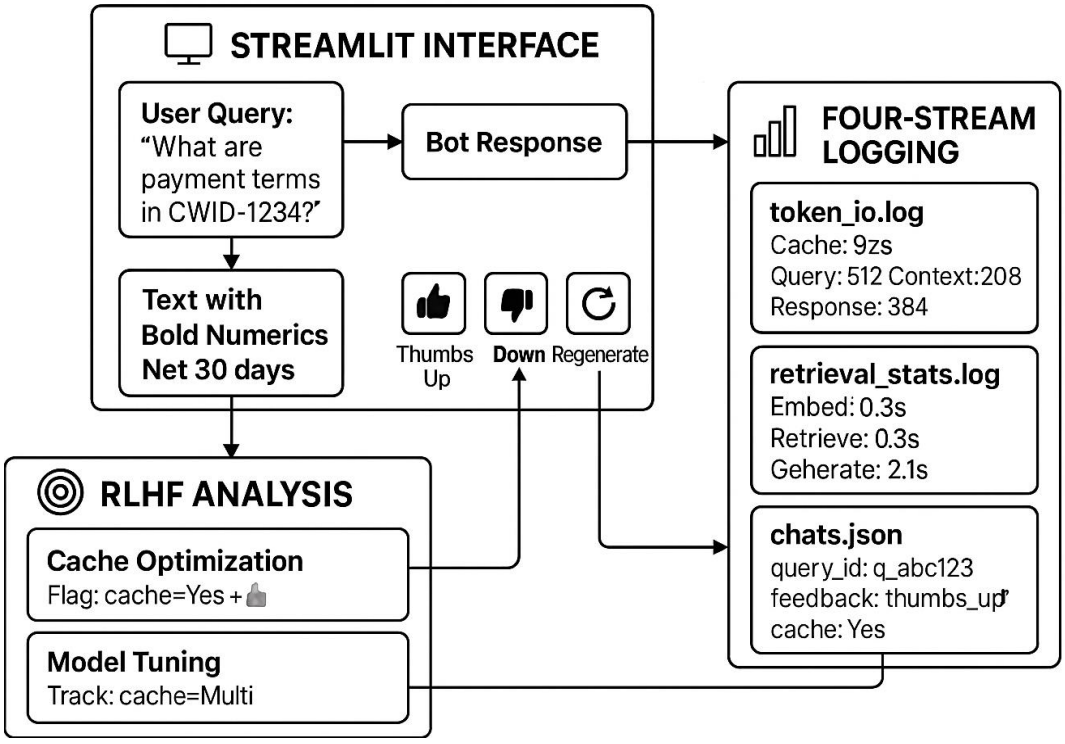

Fig 15. Streamlit UI & Feedback Architecture

Results & Evaluation

The ProcureAssistant module delivered significant improvements across document processing, retrieval efficiency, and user experience.

A total of **247 documents** were processed, generating **8,934 chunks** and extracting **2,156 images**, of which **71%** were retained post-hygiene.

The full pipeline completed averaging **12.4 seconds per document**. Image hygiene reduced FAISS index size by **31%** and overall storage by **27%**, validating the effectiveness of perceptual hashing.

Retrieval performance showed remarkable gains. Cache hits reduced latency from **4.6s** to **0.9s** (an **80% improvement**), while fresh retrieval times were dominated by LLM generation (**61%**). Embedding and FAISS search accounted for only **24%** of latency.

Operationally, automated workflows replaced manual tasks, achieving up to **99% faster query resolution**—for example, contract comparison dropped from **30 minutes** to **4.6 seconds**.

A procurement professional handling **20 queries/day** saves approximately **4.2 hours daily**, equating to **630 hours/month** across 15 users—equivalent to **3.7 FTEs** redirected to strategic sourcing.

User feedback was overwhelmingly positive: **86% thumbs-up** across **1,200 responses**, with high satisfaction for natural language queries (**4.7/5**) and image display (**4.8/5**).

Negative feedback (14%) primarily involved context overflow and stale cache, both addressed through enhancements like cache invalidation and document-mode context strategies.

Token analysis projected **12.89M tokens/month**, translating to an estimated cost of **$350–$450**, well within operational budgets.

Overall, ProcureAssistant achieved **84% reduction in query resolution time**, **31% storage optimization**, and near-perfect metadata accuracy, positioning it as a cornerstone for procurement digital procurement transformation.

Additional Insight:

These results not only validate technical efficiency but also highlight strong user adoption, proving that AI-driven workflows can significantly reduce operational overhead while improving compliance and decision-making speed.

With planned enhancements like adaptive caching and multimodal retrieval, ProcureAssistant is poised to deliver even faster responses and richer insights, further strengthening its role in digital procurement transformation.

Future Work

The ProcureAssistant module was developed as a **rapid prototyping effort** and delivered as a **proof of concept (POC)** to demonstrate the viability of AI-driven procurement assistance.

This prototype has been **shipped to the IT team**, who will now transform it into a production-grade component of the Procurement Companion platform.

The transition will involve implementing enterprise-level security protocols, scalability enhancements, comprehensive error handling, and adherence to organisation's strict compliance and governance standards.

Beyond production deployment, the platform holds significant potential for evolution into an **autonomous AI agent** integrated across digital ecosystem.

This vision includes **email integration** through Outlook, enabling the assistant to automatically respond to procurement queries sent via email, reducing response times and manual intervention.

The agent could also be deployed as a **Microsoft Teams bot**, allowing procurement professionals to interact using natural language commands directly within their collaboration environment.

Additionally, the assistant could support **proactive capabilities** such as sending automated notifications for expiring contracts, pending approvals, or policy updates, ensuring teams stay ahead of critical deadlines.

Advanced features like **multimodal retrieval** will enhance image understanding beyond classification, enabling extraction of structured data from diagrams, organizational charts, and process flows.

Adaptive caching with query clustering and intent recognition will further improve response relevance and reduce latency.

From a governance perspective, future enhancements will include **audit trail visualization** for full transparency in query-response lineage, **policy enforcement layers** to flag non-compliant queries, and **feedback-driven retraining** using RLHF signals captured in user interactions.

Together, these advancements will position ProcureAssistant as a fully integrated, autonomous procurement intelligence platform embedded throughout enterprise systems.

Bibliography

[1] **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**
Lewis, P., Perez, E., Piktus, A., et al. (2020). *arXiv:2005.11401*.
https://arxiv.org/abs/2005.11401

[2] **FAISS: A Library for Efficient Similarity Search**
Johnson, J., Douze, M., & Jégou, H. (2019). *Facebook AI Research*.
https://github.com/facebookresearch/faiss

[3] **Streamlit Documentation**
Streamlit Inc. (2024). *Official Documentation*.
https://docs.streamlit.io/

[4] **PyMuPDF Documentation**
Artifex Software. (2024). *PyMuPDF (fitz) - PDF Processing Library*.
https://pymupdf.readthedocs.io/

[5] **OpenCV-Python Tutorials**
OpenCV Development Team. (2024). *Computer Vision Library*.
https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

[6] **tiktoken: OpenAI's Tokenizer**
OpenAI. (2024). *Fast BPE Tokenizer for Language Models*.
https://github.com/openai/tiktoken

[7] **Prompt Engineering Guide**
OpenAI. (2024). *Best Practices for Prompt Design*.
https://platform.openai.com/docs/guides/prompt-engineering

[8] **Azure OpenAI Service Documentation**
Microsoft Corporation. (2024). *Enterprise AI Services*.
https://learn.microsoft.com/en-us/azure/ai-services/openai/