DEEVIA SOFTWARE (INDIA) PVT. LTD.
#921, Lakshmi Towers, Level 2, 5th Main
Sector 7, HSR Layout
Bengaluru – 560102, Karnataka, India

AI-DMS

AI-Driven Intelligent Document Management System

Deevia Software India Pvt Ltd
GridDB Cloud IOT Hackothan
Toshiba Software India Pvt. Ltd.

Table of Contents

1. Inspiration

- Organizations everywhere are still buried under mountains of PDFs, scanned files, manuals, and records with no smart way to search them. Finding one small piece of information can take hours of manual digging.
- Most document systems just store files — they don't actually understand what's inside them. Teams constantly ask, "Where is that document?" or "Where was that clause mentioned?"
- We want to turn document collections into something you can actually *talk to* — an intelligent, searchable knowledge base.
- And we want to show that GridDB isn't just for IoT and sensor data — it can power smart document intelligence too.

2. What It Does

- Turns static documents into an intelligent, searchable knowledge base — it understands content, not just file names.
- Enables AI-powered semantic search and chat over documents, including text extraction from scanned files using OCR as part of the AI pipeline.
- Provides secure Role-Based Access Control (RBAC) across folders and documents.
- Reduces LLM costs by reusing previous responses through semantic similarity matching on stored Q/A embeddings.
- Maintains an immutable, filterable, and exportable audit trail of all users, documents, chats, and system events.
- Supports document versioning, archival, structured folder management, and metadata tagging.

3. How We Built It

- **Backend:** Built with Python FastAPI (async-enabled) and Socket.IO for real-time WebSocket communication.

- **Frontend:** Developed using React and TypeScript for a responsive, modern UI.
- **Database:** Powered by GridDB (via JPype) using a container-per-file architecture — each document has its own container to enable parallel reads and writes without contention.
- **Lifecycle Management:** Leveraged GridDB's built-in partition expiry to automatically manage and clean up chat history, Q/A data, and audit logs.

4. Challenges We Ran Into

- Designing a complex ACL tree to control access at the file and folder level — simplified by leveraging GridDB's container model, where access directly maps to allowed containers.
- Performing in-file search across thousands of documents without triggering expensive table scans — solved through a container-per-file architecture that eliminates full-table scans.
- Handling document version comparisons that were becoming heavy SQL-style operations — optimized using GridDB's time-ordered container model for natural time-series queries.
- Managing chat and Q&A lifecycle (TTL, cache invalidation, cleanup) — traditionally requiring tools like Redis — but handled natively using GridDB's partition expiry.

5. Accomplishments We're Proud Of

- Proved that GridDB is not limited to sensor/IoT time-series — it excels wherever data behaves like time-ordered events (documents, chats, questions, versions, audits).
- Designed the entire system around GridDB's container philosophy instead of forcing it into a relational database pattern.
- Built cost-effective Q&A that avoids redundant LLM calls by reusing past answers through semantic similarity search.

- Achieved RBAC-aware search that is naturally scoped — no full table scans, no join-heavy ACL traversal, just direct container lookup.
- Eliminated the need for external caching (e.g., Redis) and scheduled cleanup (cron jobs) by leveraging GridDB's native partition expiry.
- Delivered a complete end-to-end intelligent document platform — from OCR ingestion to semantic chat — as a working prototype.

6. What We Learned

- Documents, chats, Q&A pairs, versions, and audit logs behave like event streams — making a time-series database a more natural fit than a traditional RDBMS.
- A container-per-file architecture eliminates table contention and simplifies access control compared to row-level filtering with complex joins.
- GridDB's hybrid in-memory + disk model enables built-in hot/cold data separation without requiring a separate caching layer.
- Working with JPype requires disciplined thread-safety management, but the architectural and performance benefits make it worthwhile.
- Designing around the database's strengths — instead of forcing relational patterns — results in a cleaner and simpler system architecture.

7. What's Next for AIDMS

- Enable parallel uploads, searches, and chats that scale naturally since each document operates within its own container — avoiding table contention.
- Extend version comparison and document evolution into native time-series analytics capabilities.

- Optimize hot/cold data movement so active data remains in memory while older data shifts to disk — scaling efficiently to millions of files without redesign.
- Introduce cross-department knowledge graphs linking related documents across organizational boundaries.
- Enhance multi-language OCR and translation pipelines for broader accessibility.
- Scale the system into a production-grade deployment suitable for real PSU and enterprise environments.
- Integrate a structure-first RAG that navigates a document's TOC tree instead of using embeddings, chunking, or vector search.

8. Built With

Python · FastAPI · React · Llama 3.1 · GridDB · PaddleOCR