




# Integrating AI Assistants with Electronic Lab Notebooks using the LabArchives Model Context Protocol Server

Samuel N. Brudner <sup>1</sup>

<sup>1</sup> Molecular, Cellular, and Developmental Biology, Yale University, USA

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: 

Submitted: 01 October 2025

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/))

## Summary

Electronic lab notebooks (ELNs) are essential tools for modern research data management, providing digital alternatives to traditional paper notebooks. Since they often contain unstructured data like ad-hoc notes, they can be challenging to search and integrate. Recent advances in large language models (LLMs) and AI assistants have demonstrated powerful capabilities for natural language interaction with structured data ([Brown et al., 2020](#); [OpenAI, 2023](#)). However, ELN platforms have remained largely isolated from these AI tools, requiring researchers to manually copy-paste content between systems. `lab_archives_mcp` bridges this gap through two complementary components: (1) a Model Context Protocol (MCP) server that connects AI assistants to LabArchives—a widely adopted commercial ELN platform—and (2) a modular, backend-agnostic vector database framework for semantic search. This dual architecture enables researchers to conversationally query their lab notebooks, perform semantic searches across experimental records using configurable embedding models and vector indices, and seamlessly incorporate lab data into AI-assisted research workflows. The vector backend is designed as a standalone library, usable independently of the MCP server for custom search pipelines.

## Statement of Need

Research laboratories generate vast amounts of structured and unstructured data stored in ELNs. While ELNs excel at organizing and archiving experimental records, extracting insights from historical data remains challenging. Researchers typically rely on manual keyword searches or must reconstruct experimental contexts from memory. This friction becomes particularly acute when:

1. **Cross-referencing experiments:** Finding related protocols, reagents, or observations across multiple notebook pages
2. **Onboarding new lab members:** Searching for institutional knowledge buried in years of lab records
3. **Reproducing analyses:** Locating specific experimental parameters or conditions from past work
4. **Writing manuscripts:** Retrieving methodological details documented months or years earlier

Recent advances in large language models (LLMs) and AI assistants have demonstrated powerful capabilities for natural language interaction with structured data ([Brown et al., 2020](#); [OpenAI, 2023](#)). However, ELN platforms have remained largely isolated from these AI tools, requiring researchers to manually copy-paste content between systems.

`lab_archives_mcp` addresses this gap by implementing an MCP server ([Anthropic, 2024](#)) that

exposes LabArchives notebooks to AI assistants like Claude Desktop and Windsurf. The Model Context Protocol enables AI assistants to discover and invoke tools through self-describing schemas—each tool provides its name, purpose, and parameter specifications, allowing the AI to autonomously determine when and how to call appropriate functions based on user queries. By connecting AI assistants to laboratory notebooks, researchers can:

- Query their experimental history using natural language—the AI assistant automatically navigates notebook hierarchies, retrieves relevant pages, and synthesizes answers (e.g., “Which fly lines did I use in navigation experiments last summer?”)
- Ask the AI to perform semantic searches across all notebook content using vector embeddings to find conceptually related experiments
- Request cross-notebook analyses where the AI assistant autonomously aggregates data from multiple experimental records
- Archive computational outputs with Git provenance metadata through AI-directed uploads

## Design and Implementation

### Architecture

lab\_archives\_mcp consists of four primary components:

1. **Authentication Layer** (auth.py): Implements HMAC-SHA512 request signing for the LabArchives REST API, OAuth-based user ID resolution, and secure credential management
2. **API Client** (eln\_client.py): Provides async HTTP methods for listing notebooks, navigating page hierarchies, and reading entry content, with automatic XML→JSON transformation using Pydantic models (Colvin & others, 2024)
3. **MCP Server** (mcp\_server.py): Exposes notebook operations as MCP tools using the FastMCP framework (Lowin, 2024), enabling AI assistants to invoke API methods through natural language
4. **Vector Backend** (vector\_backend/): A modular semantic search framework with configurable text chunking strategies, embedding model abstraction (OpenAI, extensible to local models), and backend-agnostic vector index operations supporting Pinecone (Inc., 2024) and Qdrant. This component is designed as a standalone library with Hydra-based configuration management (Yadan, 2024), enabling researchers to find conceptually related content beyond keyword matching and integrate semantic search into custom pipelines independent of the MCP server.

### Key Features

**Secure API Integration:** All requests are signed using LabArchives API credentials with HMAC-SHA512, ensuring secure access without exposing passwords. The authentication flow supports both temporary password tokens and OAuth-based UID resolution.

**Structured Data Access:** The software transforms LabArchives’ XML API responses into validated JSON schemas using Pydantic, providing type-safe interfaces and comprehensive error handling. All API responses are normalized into consistent Python dataclasses.

**Reproducible Environments:** The project uses conda-lock to pin all dependencies, ensuring bit-for-bit reproducibility across development, testing, and production environments. This approach aligns with FAIR data principles (Wilkinson et al., 2016) for computational research.

**Modular Semantic Search Framework:** The vector backend provides a flexible, configuration-driven pipeline for semantic search with pluggable components fully configurable through YAML files. Text chunking parameters (chunk size, overlap, tokenizer, boundary preservation), embedding models (OpenAI API with dimensions, batch size, timeout, extensible to sentence-transformers), and vector storage backends (Pinecone, Qdrant, local Parquet) are all specified in

Hydra-managed configuration (Yadan, 2024). This enables reproducible search configurations across environments. The local Parquet backend stores embeddings as compressed files on institutional infrastructure with version isolation, providing maximum data sovereignty. Researchers can search notebooks by concept rather than exact keywords; for example, querying “olfactory navigation behavior” retrieves relevant pages using terminology like “odor-guided flight” or “chemotaxis assays.” The framework is usable as a standalone library for custom search implementations beyond the MCP integration.

**Experimental Upload Tool:** An experimental upload tool allows researchers to use AI assistants to archive computational outputs (notebooks, figures, analysis scripts) directly to LabArchives with Git provenance metadata, supporting reproducible research workflows.

## Security Considerations

LabArchives is a cloud-based platform that provides secure storage for sensitive research data with access controls, audit trails, and compliance features. When using the vector search functionality, researchers should be aware that **indexing notebook content into external vector stores (Pinecone, Qdrant cloud) creates additional copies of research data on separate third-party infrastructure.** These copies are subject to the vector store providers’ security policies, creating an additional layer of data custody beyond LabArchives itself.

For sensitive or proprietary research data, this repository provides multiple deployment options with increasing levels of institutional control:

1. **Use local Parquet persistence** - Store embeddings as compressed Parquet files on institutional infrastructure without any external services. This provides maximum data sovereignty while enabling semantic search through local vector similarity computations. Supports concurrent multi-user indexing via per-notebook file locking, making it suitable for research teams sharing a common embedding repository via DVC.
2. **Use self-hosted vector stores** - Deploy local Qdrant instances within institutional boundaries for optimized semantic search with institutional control over all data. Vector databases offer additional features like filtered search and may provide better performance for very high-throughput scenarios (>100 concurrent writers).
3. **Index only non-sensitive notebooks to cloud services** - For non-sensitive research data, cloud vector stores (Pinecone) offer convenience and scalability.
4. **Review vector store provider security policies** to ensure compliance with institutional data governance requirements.
5. **Implement namespace isolation** in shared vector indices by using per-user or per-notebook namespaces (or partitions) so embeddings and search results from one tenant are invisible to others, preventing cross-user data exposure.

Namespaces are configured via Hydra (conf/vector\_search/default.yaml) with overrides such as index.namespace=notebook\_123. Deployed services must inject the namespace derived from the authenticated LabArchives tenant and deny requests that omit or override it. Pinecone and Qdrant both support per-key access scoping; administrators should issue per-tenant API keys (or run a gateway that signs requests) so that end users cannot query or upsert outside their assigned namespace.

The vector store metadata schema includes notebook\_id, notebook\_name, author, date, and optional tags fields, enabling fine-grained access control through metadata filtering. These fields allow administrators to restrict search results to specific notebooks, authors, or date ranges, supporting multi-tenant deployments where users should only access their own data.

The MCP server exposes both read and write capabilities. Read operations (listing notebooks, reading pages) query LabArchives on-demand without creating external copies. The experimental upload\_to\_labarchives tool allows AI assistants to write files directly to notebooks with Git provenance metadata. **For production deployments, administrators can disable write**

capabilities by setting the `LABARCHIVES_ENABLE_UPLOAD=false` environment variable, preventing unintended modifications to research records. Alternatively, approval workflows can be implemented at the AI assistant level. When using cloud vector stores, the indexing workflow creates external data copies as described above; however, local Parquet persistence keeps all data within institutional boundaries. Researchers must evaluate whether the semantic search and upload benefits justify any security tradeoffs for their specific use case and deployment model.

## Testing and Quality Assurance

The codebase maintains comprehensive test coverage with unit tests for all API methods, integration tests against live LabArchives instances (skipped in CI without credentials), and property-based tests using Hypothesis (MacIver, 2024) for numeric operations. Pre-commit hooks enforce code quality through Ruff linting, Black formatting, isort import sorting, and mypy type checking.

Continuous integration via GitHub Actions runs the test suite on Ubuntu and macOS across Python 3.11 and 3.12, ensuring cross-platform compatibility.

## Usage Example

After configuring LabArchives API credentials, researchers interact with their notebooks through AI assistants:

Researcher: "What protocols did I use for mosquito navigation experiments?"

Claude (via MCP): [calls `list_labarchives_notebooks()`, then  
`list_notebook_pages("Mosquito Navigation")`, then  
`read_notebook_page(page_id="protocols")`]

Claude: "Your Mosquito Navigation notebook contains three main protocols:

1. Wind tunnel setup with IR tracking
2. Odor delivery system calibration
3. Flight trajectory analysis pipeline

The most recent version was updated on August 15, 2025."

This conversational interface reduces the cognitive overhead of manual search and enables rapid knowledge retrieval.

## Comparison to Existing Tools

While commercial ELN platforms (LabArchives, Benchling, eLabFTW) provide web APIs, existing open-source tools have focused primarily on ELN data export (Bio-ITech, 2024) or laboratory information management systems (LIMS) integration. LabArchives' own REST API offers CRUD access to notebook entries but leaves clients to implement indexing and search logic, typically through keyword queries (LabArchives, 2024). None of these offerings provide a turnkey bridge between ELN data and AI assistants, and semantic retrieval remains an exercise for downstream integrators.

The Model Context Protocol is a recently introduced standard for connecting AI systems to external data sources (Anthropic, 2024). `lab_archives_mcp` represents an early academic application of this protocol for research data management, demonstrating a reusable pattern for integrating institutional data systems with AI assistants through both standardized MCP interfaces and modular vector search infrastructure. In contrast to raw REST APIs, this project packages:

- **MCP-native connectors** that expose LabArchives notebooks, pages, and entries directly to assistants without bespoke glue code.
- **Config-driven semantic search** via `conf/vector_search/default.yaml`, enabling reproducible chunking, embedding, and vector indexing beyond simple keyword lookups.
- **Governance controls** including namespace isolation, Hydra-driven credential scoping, and local Parquet persistence so institutions can enforce tenancy boundaries while retaining the option for self-hosted search backends.

## Acknowledgements

This work was performed independently by the author, and validated against a LabArchives account at Yale University. The author thanks LabArchives for API documentation and technical support, and the FastMCP and Anthropic teams for the Model Context Protocol specification.

## References

- Anthropic. (2024). *Model context protocol specification*. <https://modelcontextprotocol.io/>
- Bio-ITech. (2024). *eLabJournal API documentation*. <https://www.elabjournal.com/api/>
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., & others. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901. <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html>
- Colvin, S., & others. (2024). *Pydantic: Data validation using python type hints*. <https://docs.pydantic.dev/>
- Inc., P. S. (2024). *Pinecone: Vector database for machine learning*. <https://www.pinecone.io/>
- LabArchives, L. (2024). *LabArchives electronic laboratory notebook*. <https://www.labarchives.com/>
- Lowin, J. (2024). *FastMCP: A fast, developer-friendly framework for building model context protocol servers*. <https://github.com/jlowin/fastmcp>
- MacIver, D. R. (2024). *Hypothesis: Property-based testing for python*. <https://hypothesis.readthedocs.io/>
- OpenAI. (2023). GPT-4 technical report. *arXiv Preprint arXiv:2303.08774*. <https://doi.org/10.48550/arXiv.2303.08774>
- Wilkinson, M. D., Dumontier, M., Aalbersberg, Ij. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., Silva Santos, L. B. da, Bourne, P. E., & others. (2016). The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3(1), 1–9. <https://doi.org/10.1038/sdata.2016.18>
- Yadan, O. (2024). *Hydra: A framework for elegantly configuring complex applications*. <https://hydra.cc/>