Software paper for submission to the Journal of Open Research Software

To complete this template, please replace the blue text with your own. The paper has three main sections: (1) Overview; (2) Availability; (3) Reuse potential.

Please submit the completed paper to: editor.jors@ubiquitypress.com

---

(1) Overview

Title

MiND: A Deterministic Middleware for Auditing, Reproducing, and Logging Large Language Model Interactions in Research Workflows

Paper Authors

1. Melo, Edervaldo José de

Paper Author Roles and Affiliations

1. Author and developer, independent researcher.

Abstract

Large Language Models (LLMs) are increasingly integrated into research workflows, yet their use poses persistent challenges related to auditability, reproducibility, and traceability of interactions. MiND is a deterministic middleware that addresses these challenges by providing a structured execution layer between researchers and LLM APIs. The software enforces controlled prompt execution, deterministic configuration handling, and comprehensive logging of inputs, outputs, and runtime parameters. By externalizing interaction state and control logic, MiND enables researchers to reproduce defined runs, audit interaction traces, and generate persistent records suitable for inspection and verification. MiND is intentionally non-agentic and does not modify underlying models; it focuses on infrastructural transparency and portability of LLM interaction pipelines. Source code and documentation are publicly available in a versioned repository.

Keywords

deterministic middleware; reproducible AI; LLM auditing; research software infrastructure; logging systems

Introduction

The increasing use of large language models in research workflows has exposed practical limitations related to auditability, reproducibility, and long-term traceability of model interactions. In many experimental and applied research settings, prompt execution, context handling, and interaction history are managed implicitly by

external platforms or ad-hoc application logic, making it difficult to reproduce results, inspect execution conditions, or verify how outputs were produced.

MiND addresses this need by providing a deterministic middleware layer that externalizes control over LLM interactions. By explicitly managing prompt execution, configuration parameters, and interaction logs outside the model, MiND enables researchers to reproduce defined runs, audit interaction traces, and maintain persistent records suitable for inspection and verification. The software is intended for researchers and developers who require controlled and inspectable LLM usage as part of experimental pipelines, evaluation studies, or privacy-sensitive research environments, without modifying or fine-tuning the underlying models.

Implementation and architecture

MiND is implemented in Python as a minimal deterministic middleware positioned between users and Large Language Model (LLM) APIs. The architecture externalizes interaction control, state management, and logging from the model, allowing each LLM invocation to be treated as an explicit and inspectable computational step.

The core execution flow is initialized through the backend entry point located at `src/backend/main.py`. Interactions are processed through predefined execution modules responsible for input handling, context management, request routing, and response recording. This modular design enforces strict separation of responsibilities and prevents unintended cross-contamination of conversational state.

Interaction state is managed externally and may be persisted across sessions. During execution, MiND generates structured artifacts, including JSON-based interaction records and persistent logs stored in a relational database. These artifacts capture inputs, outputs, configuration parameters, and execution metadata, enabling post-hoc inspection and reproducibility.

MiND is API-agnostic and does not rely on provider-specific conversation histories or proprietary state mechanisms. The middleware does not implement agentic behaviors, autonomous planning, or internal model modification. Its contribution is infrastructural: providing a deterministic and inspectable orchestration layer around existing LLM systems.

Quality control

MiND has been manually tested in local development environments using Python 3.9 and above. Functional validation consists of executing controlled interaction cycles through the backend entry point and verifying that structured JSON logs are generated correctly and persistently stored.

Basic validation includes:

- Verification that input prompts are correctly routed through the orchestrator.

- Confirmation that configuration parameters are loaded deterministically.
- Inspection of generated JSON interaction logs.
- Validation that execution metadata (timestamps, parameters, responses) are properly recorded.

Because MiND acts as a middleware layer rather than a model implementation, correctness is evaluated by confirming deterministic execution flow and reproducibility of logged interaction traces. Example execution workflows and configuration instructions are provided in the repository documentation to allow independent verification by users.

(2) Availability

Operating system
Cross-platform (tested on Windows 11 and Ubuntu 22.04; compatible with any operating system supporting Python 3.9 or later).

Programming language
Python (version 3.9 or higher).

Additional system requirements
Internet connection required for interaction with LLM APIs. Standard workstation-class hardware sufficient for Python execution.

Dependencies
Dependencies are specified in the requirements.txt file included in the archived release. Core dependencies include the OpenAI Python client library and standard Python packages.

List of contributors
Edervaldo José de Souza Melo (design, implementation, documentation).

Software location:

Archive (e.g. institutional repository, general repository) (required – please see instructions on journal website for depositing archive copy of software in a suitable repository)

Name: Zenodo
Persistent identifier: https://doi.org/10.5281/zenodo.18637799
Licence: GNU General Public License v3.0 only
Publisher: Edervaldo José de Souza Melo
Version published: 1.0.1
Date published: 14/02/2026

Code repository (e.g. SourceForge, GitHub etc.) (required)

Name: GitHub – edersouzamelo/nemosine-10-MiND
Identifier:
https://github.com/edersouzamelo/nemosine-10-MiND
Licence: GNU General Public License v3.0 only
Date published: 14/02/2026

Emulation environment (if appropriate)

Name: Not applicable.
Identifier: Not applicable.
Licence: Not applicable.
Date published: Not applicable.

Language
English

(3) Reuse potential

MiND can be reused in research contexts where controlled, auditable, and reproducible interaction with large language models is required. The software is particularly suitable for experimental pipelines, benchmarking studies, qualitative analysis workflows, evaluation frameworks, and privacy-sensitive environments in which external platforms do not provide sufficient execution transparency.

Because MiND externalizes configuration, orchestration logic, and logging, it can be integrated into existing research software stacks as a middleware layer without requiring modification of the underlying LLM provider. Its API-agnostic design allows adaptation to different LLM services, facilitating provider substitution and long-term portability of interaction pipelines.

The modular architecture enables further extension, such as integration with alternative logging backends, database systems, or evaluation modules. Researchers may also adapt MiND for educational purposes, software auditing studies, or methodological investigations into reproducibility of LLM-based systems.

As an infrastructural component rather than a model implementation, MiND's reuse potential lies in providing a stable execution scaffold for reproducible and inspectable LLM experimentation.

Acknowledgements

The author thanks the open-source research software community for ongoing discussions on reproducibility, research transparency, and LLM evaluation practices, which indirectly informed the design principles of MiND.

Funding statement

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Competing interests

The author declares that he has no competing interests.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.

[2] Pineau, J., Vincent-Lamarre, P., Sinha, K., et al. (2021). Improving reproducibility in machine learning research: a report from the NeurIPS 2019 reproducibility program. Journal of Machine Learning Research, 22(164), 1–20.

[3] Stodden, V., Seiler, J., & Ma, Z. (2018). An empirical analysis of journal policy effectiveness for computational reproducibility. Proceedings of the National Academy of Sciences, 115(11), 2584–2589.
