






JELAI: Integrating AI and Learning Analytics in Jupyter Notebooks

Manuel Valle Torre^(✉) , Thom van der Velden, Marcus Specht ,
and Catharine Oertel 

Delft University of Technology, Delft, Netherlands

{m.valletorre,thomvanderveld,m.m.specht,c.r.m.m.oertel}@tudelft.nl
<https://www.tudelft.nl/en/eemcs/>

Abstract. Generative AI offers potential for educational support, but often lacks pedagogical grounding and awareness of the student’s learning context. Furthermore, researching student interactions with these tools within authentic learning environments remains challenging. To address this, we present JELAI, an open-source platform architecture designed to integrate fine-grained Learning Analytics (LA) with Large Language Model (LLM)-based tutoring directly within a Jupyter Notebook environment. JELAI employs a modular, containerized design featuring JupyterLab extensions for telemetry and chat, alongside a central middleware handling LA processing and context-aware LLM prompt enrichment. This architecture enables the capture of integrated code interaction and chat data, facilitating real-time, context-sensitive AI scaffolding and research into student behaviour. We describe the system’s design, implementation, and demonstrate its feasibility through system performance benchmarks and two proof-of-concept use cases illustrating its capabilities for logging multi-modal data, analysing help-seeking patterns, and supporting A/B testing of AI configurations. JELAI’s primary contribution is its technical framework, providing a flexible tool for researchers and educators to develop, deploy, and study LA-informed AI tutoring within the widely used Jupyter ecosystem.

Keywords: AI devices and tools · Pedagogy and LLMs · Learning Analytics · Open-source

1 Introduction

Generative AI (GenAI) tools like ChatGPT offer potential for on-demand student support, but are generally designed to solve problems rather than support learning pedagogically [7]. They typically lack insight into students’ learning context (e.g., task, progress, challenges) and can foster reliance on quick answers over deeper understanding [6]. Novices struggle to provide effective context [8], leading to AI responses that may be unhelpful or misaligned with instructional goals [16]. Furthermore, research on GenAI in education often relies on self-reports

or high-level comparisons, lacking detailed analysis of student-AI interactions needed to understand learning mechanisms and design effective, pedagogically-grounded support [1, 20].

Integrating Learning Analytics (LA) with GenAI offers a path forward [18]. Digital environments like Jupyter Notebooks capture rich interaction logs [11]. Analyzing these logs alongside chat interactions can provide the context needed for adaptive, real-time interventions [13, 19] and enable research into student cognitive and metacognitive behaviours [6, 12]. While Jupyter is widely used [3], existing extensions typically focus on isolated functions like grading (NBGrader¹) or provide basic logging/tutoring without deep LA integration [5]. Recognizing the need for pedagogically-sound AI in programming, researchers have developed specialized systems like CodeTutor [14], CodeAid [9], and CodeHelp [17], which use techniques like targeted prompting and avoiding direct solutions. However, these systems often operate separately from the primary coding environment, and their design typically does not focus on the analysis of integrated student workflows within that environment. To enable such integrated analysis, there is a need for systems that tightly couple fine-grained LA (capturing detailed coding processes and errors) with context-aware conversational AI tutoring *within* the notebook environment. This integration could bridge the gap between flexible LLMs and more structured learning systems, enabling personalized feedback based on student models and promoting more effective help-seeking [2].

To address this gap, we present JELAI (*Jupyter Environment for Learning Analytics and AI*), an experimental platform integrating LLM-based tutoring with an LA pipeline inside Jupyter notebooks (Fig. 1). JELAI is designed as a modular, extensible framework to:

1. Capture and process fine-grained student activity (code edits, executions, errors) and chat interactions into meaningful learning traces.
2. Enrich LLM prompts with real-time context (e.g., recent code, errors, objectives, chat history) for adaptive scaffolding.
3. Provide a platform for educational research on student-AI interaction patterns and the impact of different AI configurations (e.g., prompting strategies), leveraging the integrated nature of the interaction data.

This paper details JELAI’s architecture, implementation, and demonstrates its feasibility through preliminary system performance data and two proof-of-concept use cases. The primary contribution is the technical design and implementation of a platform enabling LA-informed AI tutoring within a widely used educational environment. The open-source repository (BSD-3-Clause license) is available for use and collaboration².

¹ <https://nbgrader.readthedocs.io/en/stable/>.

² <https://github.com/mvallet91/JELAI>.

2 System Design and Implementation

The design of JELAI was guided by several key requirements necessary for integrating LA with AI tutoring effectively within an educational context:

- **Granular Data Logging:** Capture fine-grained interaction data (code edits, executions, errors, chat) as the foundation for LA and context-aware AI [11, 18].

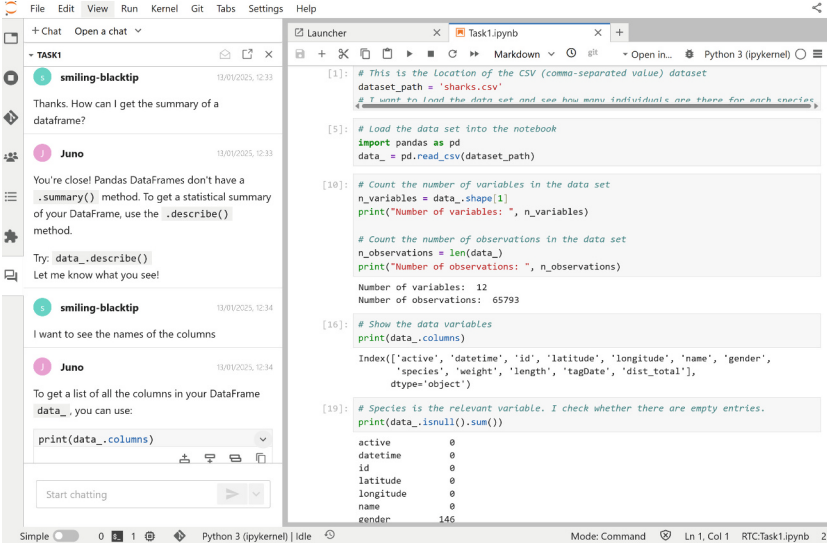


Fig. 1. JELAI Interface: Jupyter Notebook with integrated AI tutor (Juno).

- **Real-time Context Enrichment:** Leverage logged data to provide LLMs with immediate context (recent code, errors, conversation history, task goals) to improve relevance and mitigate generic responses [16].
- **Pedagogical Alignment & Scaffolding:** Allow instructors to configure AI behaviour (via system prompts, intervention strategies) to align with learning objectives and enable adaptive scaffolding based on student activity [2, 12].
- **Modularity and Extensibility:** Design components (logging, LA, LLM interaction) independently to facilitate updates, integration of new techniques, and research experimentation.
- **Scalability and Privacy:** Support multiple concurrent users efficiently while ensuring data isolation and privacy, particularly when using local LLMs.

To meet these requirements, JELAI is implemented as a modular, containerized system using open-source technologies (Docker, JupyterHub, JupyterLab, Ollama) and custom Jupyter extensions (Jupyter-Chat, JupyterLab-Pioneer).

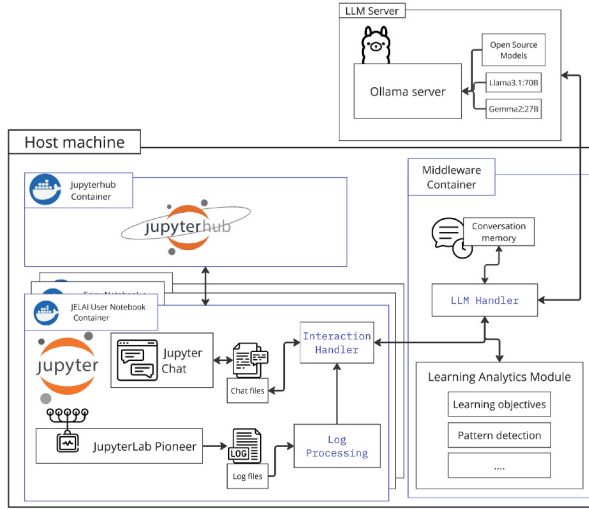


Fig. 2. JELAI System Architecture.

JELAI is deployed using `docker-compose`, simplifying setup and ensuring user isolation for scalability and privacy.

The architecture (Fig. 2) comprises four main components:

1. **User Notebook Container:** Hosts the student's JupyterLab instance. It includes the `Jupyter-Chat`³ extension for the AI tutor interface and the `JupyterLab-Pioneer`⁴ extension for capturing detailed telemetry (keystrokes, cell executions, errors, UI interactions). An **Interaction Handler** within this container preprocesses student chat messages, enriching them with immediate context (e.g., recent edits and errors, task objective) before forwarding them.
2. **Middleware Container:** Acts as the central orchestration hub. It contains the **Learning Analytics Module**, which processes incoming telemetry logs and chat data, potentially storing them or making them available for real-time analysis. It also houses the **LLM Handler**, responsible for retrieving relevant conversation history and context from the LA module, applying instructor-defined pedagogical rules or scaffolding logic (e.g., modifying prompts based on recent student errors or help-seeking patterns), constructing the final prompt, and managing communication with the LLM Server.
3. **JupyterHub Container:** Manages user authentication, spawns individual User Notebook Containers, and handles proxying, enabling multi-user deployment.
4. **LLM Server:** Provides the interface to the chosen language model. JELAI uses Ollama to support various local open-source LLMs, ensuring data privacy.

³ <https://github.com/jupyterlab/jupyter-chat>.

⁴ <https://jupyterlab-pioneer.readthedocs.io/en/latest/>.

and control, but can also be configured to connect to external APIs compatible with the OpenAI standard.

A typical interaction workflow proceeds as follows: (1) A student interacts within their JupyterLab notebook (e.g., writing code, running cells). (2) **JupyterLab-Pioneer** logs these actions and sends them asynchronously to the Middleware’s LA Module. (3) The student sends a message to the AI tutor via **Jupyter-Chat**. (4) The local **Interaction Handler** intercepts the message, adds immediate context (e.g., code from the current cell), and sends it to the Middleware. (5) The Middleware’s **LLM Handler** retrieves the message, fetches relevant context from the LA Module, applies any configured pedagogical logic or prompt engineering, and sends the final, context-enriched prompt to the LLM Server. (6) The LLM generates a response, which is sent back through the Middleware to the **Jupyter-Chat** interface for the student. This architecture ensures that AI responses are informed by both immediate actions and broader interaction history, allowing for more adaptive and pedagogically relevant support.

3 Proof-of-Concept Demonstration

To demonstrate JELAI’s feasibility and capabilities for capturing interaction data and enabling comparative studies, we conducted preliminary evaluations focusing on system performance and two use cases. These serve as proof-of-concept illustrations rather than conclusive studies, given the small sample sizes inherent in initial system testing.

3.1 System Performance

JELAI demonstrated stable performance on an A40 GPU. With local LLMs, such as Llama3.1-70b and Gemma2-27b, average response latencies were 5–9 s and 2–3 s, respectively, acceptable for interactive use. Telemetry processing occurred in near real-time. The JupyterHub deployment supported 20+ concurrent users on a moderate server (6 CPU cores, 16GB RAM) with low per-user resource consumption (CPU < 0.25 cores, RAM 400MB steady state), indicating potential for scalability.

3.2 Use Case Examples

Two small-scale studies illustrate JELAI’s data collection and experimental capabilities.

1. Help-Seeking in Python Course (N=18): JELAI was used over 7 weeks in an introductory Python course. It logged student code interactions and chat messages with an LLM tutor (Llama3.1-70b). We categorized chat prompts into instrumental vs. executive help-seeking [15]. *Demonstration:* JELAI successfully captured granular, integrated data streams (code activity, chat logs, help-seeking types). This capability allows for exploring relationships between

coding behaviours, help-seeking patterns (e.g., relative frequency of executive vs. instrumental requests), and external measures like grades, as well as identifying specific interaction sequences (e.g., help-avoidance despite errors, post-completion verification checks) for qualitative analysis.

2. Prompt Comparison Pilot (N=19): This study demonstrated JELAI's use for A/B testing AI configurations. Novice programmers worked on 4 data science tasks using JELAI with either a generic "helpful assistant" prompt or a "pedagogical" prompt (Gemma2-27b). *Demonstration:* JELAI facilitated the comparison by logging interactions for both groups and automated classification of learners. The pedagogical prompt increased dialogue length (17.7 vs. 10.7 messages) and engaged in primarily instrumental (27.9% vs. 59.6%), rather than executive requests. Programming logs allowed us to hypothesize: unprompted students ran more code (12.8 vs. 8.3 executions) and encountered more errors (7.4 vs. 5.3), a behaviour consistent with productive struggle.

Together, the pilots confirm that JELAI sustains realtime responsiveness, surfaces actionable helpseeking signals, and enables real-time behaviour analysis and rapid A/B testing of prompt designs without infrastructure changes.

4 Discussion and Future Work

This paper presented JELAI, a novel, open-source technical platform integrating LA and AI tutoring within Jupyter Notebooks. Its modular architecture allows context-aware AI interaction and, crucially, supports research into student behaviour with GenAI by enabling flexible configuration and data capture. The proof-of-concept demonstration confirmed system stability and its capability to capture rich, multi-modal data for analyzing interaction patterns and comparing pedagogical strategies, although the preliminary studies require validation with larger samples.

Key limitations include the computational cost of powerful local LLMs (though rapidly improving models mitigate this) and the technical expertise currently needed for configuration. Future work will focus on: 1) Enhancing AI interaction with multi-step processing for real-time help classification and proactive interventions [4]. 2) Simplifying configuration via templates and higher-level indicators [10]. 3) Integrating RAG and AI Agents for improved tutoring and tool use. 4) Improving interoperability via standards like xAPI and LTI [11]. 5) Enabling student collaboration features. Research will further investigate how JELAI can foster instrumental help-seeking and whether guided AI dialogue improves learning outcomes. In summary, JELAI's main contribution is its flexible technical framework, designed to enable pedagogically-informed, LA-driven AI tutoring research and development within a widely used educational environment, with ongoing work focused on enhancing its usability, capabilities, and interoperability.

References

1. Adams, D., Chuah, K.M., Devadason, E., Azzis, M.: From novice to navigator: students academic help-seeking behaviour, readiness, and perceived usefulness of ChatGPT in learning. *Educ. Inf. Technol.* **29**(11), 13617–13634 (2024)
2. Aleven, V., Roll, I., McLaren, B.M., Koedinger, K.R.: Help helps, but only so much: research on Help Seeking with Intelligent Tutoring Systems. *Int. J. Artif. Intell. Educ.* **26**(1), 205–223 (2016)
3. Cardoso, A., Leitão, J., Teixeira, C.: Using the Jupyter notebook as a tool to support the teaching and learning processes in engineering courses. In: Auer, M.E., Tsiatsos, T. (eds.) *The Challenges of the Digital Transformation in Education*, pp. 227–236. Springer International Publishing, Cham (2019)https://doi.org/10.1007/978-3-030-11935-5_22
4. Chen, A., et al.: Unpacking help-seeking process through multimodal learning analytics: a comparative study of ChatGPT vs Human expert. *Comput. Educ.* **226**, 105198 (2025). <https://doi.org/10.1016/j.compedu.2024.105198>, <https://www.sciencedirect.com/science/article/pii/S0360131524002124>
5. Chen, E., Huang, R., Chen, H.S., Tseng, Y.H., Li, L.Y.: GPTutor: a ChatGPT-powered programming tool for code explanation. In: *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky*, pp. 321–327. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-36336-8_50. iSSN: 1865-0937
6. Fan, Y., et al.: Beware of metacognitive laziness: effects of generative artificial intelligence on learning motivation, processes, and performance. *Br. J. Educ. Technol.* **n/a**(n/a) (2024). <https://doi.org/10.1111/bjet.13544>
7. Giannakos, M., et al.: The promise and challenges of generative AI in education. *Behav. Inf. Technol.*, 1–27 (2024)
8. Kazemitabaar, M., Hou, X., Henley, A., Ericson, B.J., Weintrop, D., Grossman, T.: How novices use LLM-based code generators to solve CS1 coding tasks in a self-paced learning environment. In: *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*, pp. 1–12. Koli Calling '23, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3631802.3631806>
9. Kazemitabaar, M., Ye, R., Wang, X., Henley, A.Z., Denny, P., Craig, M., Grossman, T.: CodeAid: evaluating a classroom deployment of an LLM-based programming assistant that balances student and educator needs. In: *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pp. 1–20. CHI '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3613904.3642773>
10. Kerimbayev, N., Adamova, K., Shadiey, R., Altinay, Z.: Intelligent educational technologies in individual learning: a systematic literature review. *Smart Learn. Environ.* **12**(1), 1 (2025)
11. Kitto, K., Whitmer, J., Silvers, A., Webb, M.: Creating data for learning analytics ecosystems [position paper]. Report, Society for Learning Analytics Research (2020). <https://opus.lib.uts.edu.au/handle/10453/152209>. Accepted: 2021-12-08T00:01:48Z

12. Ko, S.H., Stephens-Martinez, K.: The trees in the forest: characterizing computing students' individual help-seeking approaches. In: Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1. ICER '24, vol. 1, pp. 343–358. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3632620.3671099>
13. Liu, J., Li, S., Dong, Q.: Collaboration with generative artificial intelligence: an exploratory study based on learning analytics. *J. Educ. Comput. Res.* **62**(5), 1234–1266 (2024)
14. Lyu, W., Wang, Y., Chung, T.R., Sun, Y., Zhang, Y.: Evaluating the effectiveness of LLMs in introductory computer science education: a semester-long field study. In: Proceedings of the Eleventh ACM Conference on Learning @ Scale, pp. 63–74. L@S '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3657604.3662036>
15. Nelson-Le Gall, S.: Help-seeking: an understudied problem-solving skill in children. *Dev. Rev.* **1**(3), 224–246 (1981)
16. Pal Chowdhury, S., Zouhar, V., Sachan, M.: AutoTutor meets large language models: a language model tutor with rich pedagogy and guardrails. In: Proceedings of the Eleventh ACM Conference on Learning @ Scale, pp. 5–15. L@S '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3657604.3662041>
17. Sheese, B., Liffiton, M., Savelka, J., Denny, P.: Patterns of student help-seeking when using a large language model-powered programming assistant. In: Proceedings of the 26th Australasian Computing Education Conference, pp. 49–57. ACE '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3636243.3636249>
18. Su, H., Tong, Y., Zhang, X., Fan, Y.: Uncovering students processing tactics towards ChatGPT feedback in EFL education using learning analytics. In: Ma, W.W.K., Li, C., Fan, C.W., U, L.H., Lu, A. (eds.) *Blended Learning. Intelligent Computing in Education*, pp. 238–250. Springer Nature, Singapore (2024). https://doi.org/10.1007/978-981-97-4442-8_18
19. Valle Torre, M., Oertel, C., Specht, M.: The sequence matters in learning - a systematic literature review. In: Proceedings of the 14th Learning Analytics and Knowledge Conference, pp. 263–272. LAK '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3636555.3636880>
20. Zhang, X., et al.: A systematic literature review of empirical research on applying generative artificial intelligence in education. *Front. Digit. Educ.* **1**(3), 223–245 (2024)