TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
Koohestani, R., Bateni, P., Ebrahimi, A., Etezadi, B., Karimi, K., & Izadi, M. (2026). Code4MeV2: A Research-oriented Code-completion Platform. In *Proceedings - 2026 IEEE/ACM 48th International Conference on Software Engineering: Companion Proceedings, ICSE-Companion 2026* (pp. 101-104). (Proceedings - 2026 IEEE/ACM 48th International Conference on Software Engineering: Companion Proceedings, ICSE-Companion 2026). Association for Computing Machinery (ACM). https://doi.org/10.1145/3774748.3787616

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

Code4MeV2: a Research-oriented Code-completion Platform

Roham Koohestani
Delft University of Technology
Delft, Netherlands
rkoohestani@tudelft.nl

Parham Bateni
Delft University of Technology
Delft, Netherlands
pbateni@tudelft.nl

Aydin Ebrahimi
Delft University of Technology
Delft, Netherlands
aydinebrahimi@tudelft.nl

Behdad Etezadi
Delft University of Technology
Delft, Netherlands
betezadi@tudelft.nl

Kiarash Karimi
Delft University of Technology
Delft, Netherlands
makarimi@tudelft.nl

Maliheh Izadi
Delft University of Technology
Delft, Netherlands
m.izadi@tudelft.nl

Abstract

The adoption of AI-powered code completion tools in software development has increased substantially, yet the user interaction data produced by these systems remain proprietary within large corporations. This creates a barrier for the academic community, as researchers must often develop dedicated platforms to conduct studies on human–AI interaction, making reproducible research and large-scale data analysis impractical. In this work, we introduce Code4Me V2, a research-oriented, open-source code completion plugin for JetBrains IDEs, as a solution to this limitation. Code4Me V2 is designed using a client-server architecture and features inline code completion and a context-aware chat assistant. Its core contribution is a modular and transparent data collection framework that gives researchers fine-grained control over telemetry and context gathering. Code4Me V2 achieves industry-comparable performance in terms of code completion, with an average latency of 200ms. We assess our tool through a combination of an expert evaluation and a user study with eight participants. Feedback from both researchers and daily users highlights its informativeness and usefulness. We invite the community to adopt and contribute to this tool. More information about the tool can be found on https://app.code4me.me.

CCS Concepts

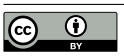
• **Human-centered computing** → **Human computer interaction (HCI)**; • **Software and its engineering**;

Keywords

AI-assisted Programming, Research Tooling, Open Science, AI4SE

ACM Reference Format:
Roham Koohestani, Parham Bateni, Aydin Ebrahimi, Behdad Etezadi, Kiarash Karimi, and Maliheh Izadi. 2026. Code4MeV2: a Research-oriented Code-completion Platform. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-Companion '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3774748.3787616

[*] Authors contributed equally.



This work is licensed under a Creative Commons Attribution 4.0 International License.
ICSE-Companion '26, Rio de Janeiro, Brazil
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2296-7/26/04
https://doi.org/10.1145/3774748.3787616

1 Introduction

With the increased use of AI in software engineering, developer workflows are undergoing a fundamental transformation. Industry surveys report widespread adoption of AI coding assistants [16], with empirical studies demonstrating significant productivity gains; for instance, GitHub Copilot users have been observed to complete tasks up to 55% faster [11]. Similarly, JetBrains reports that users save up to 8 hours per week with its AI Assistant [8], and an Amazon study found that developers using CodeWhisperer completed tasks 57% faster than those who did not [1].

While the benefits of these tools are evident [10], the study of interaction modalities and the wide-scale analysis of developer behavioral data remain limited. This is because the most powerful systems are commercial and closed-source, which presents three critical challenges for academic research: a lack of transparency into the models' decision-making processes, an inability to control experimental conditions (e.g., model versions or data sources), and no access to the rich telemetry data required for fine-grained analysis. Existing academic studies have often required the creation of bespoke, single-purpose tools, consequently restricting the ability to reuse the infrastructure [6] [9].

To address this gap, we propose Code4Me V2, an extensible and research-oriented framework for empirical analysis of AI-assisted software development. As an open-source plugin for JetBrains IDEs, Code4Me V2 provides core functionalities, such as inline code completion and an interactive chat assistant. Its primary purpose, however, is to serve as a platform for experimentation. The architecture is explicitly modular, and it separates user-facing components from a configurable data collection and model inference backend.

We expect this tool to serve as a useful infrastructure component for the AI4SE community. By offering a shared, transparent, and extensible platform, Code4Me V2 enables researchers to focus on experimental design and data analysis rather than on building and maintaining their own data collection systems. In the subsequent sections, we analyze the domain in section 2, describe the design of Code4Me V2 in section 3, present a preliminary analysis in section 4, and discuss limitations and future work in section 5. All relevant links for Code4Me V2 can be found at https://app.code4me.me.

2 Domain Analysis

The current ecosystem of AI coding assistants is dominated by powerful commercial tools, which, despite their utility, present

significant limitations for research. Platforms like **GitHub Copilot** [5], **Cursor** [3], and **JetBrains AI** [7] offer robust, integrated user experiences. However, their proprietary nature makes them effectively "black boxes." Researchers cannot inspect the suggestion logic, control for model updates that could act as confounding variables, or access raw interaction data.

Open-source alternatives, such as **Cody** [15] and **Code4Me** [6], offer more transparency but were not designed with research extensibility as a primary goal. Their architectures often lack modularity, which complicates systematic studies of developer–AI interaction. For example, adapting these systems to support different experimental configurations or data collection strategies typically requires substantial modification of the core codebase.

To better understand these challenges, we analyzed seven widely used coding assistants and their documentation. [1] Finding that both commercial and open-source options fall short of research needs, we designed Code4Me V2 for the academic community, which is transparent, with an open-source platform from plugin to backend; controllable, giving researchers fine-grained control over data collection through a modular system; and extensible, allowing new components such as telemetry modules or context providers to be added with minimal effort.

3 Code4Me V2

Code4Me V2 is a flexible, extensible research platform built on modularity and separation of concerns. This section outlines its architecture and design, with Figure 2 and Figure 3 showing the plugin and analytics dashboard.

3.1 General Architecture

The system is designed using a client-server architecture to enforce a strict separation of concerns. This allows for the different components to be modified separately from each other, and allows for further extension down the line (e.g., by adding a VSCode Client, or adding a new model). In terms of the current implementation, **The Client** is a lightweight JetBrains IDE plugin. Its sole responsibilities are rendering the user interface (ghost text and chat), collecting user-configurable context and telemetry from the editor, and dispatching requests to the server. **The Server** handles all heavy lifting: user authentication, persistent data storage in a relational database, and the computationally intensive AI model inference pipeline. The decision for using the client-server architecture comes into play here as well. It minimizes the performance overhead on the developer's IDE, as all significant computation is offloaded. In addition to these main components, we have developed the **Analysis Platform**, which serves as the user interface for the researchers. This component is in direct communication with the backend and helps the researcher analyze the collected data, gain insights into usage patterns, and set up user studies. In the sub-sections to follow, we will go more in-depth into each of these components.

[1] The platforms analyzed include: GitHub Copilot, Cursor, Cody, Devin, Replit Ghostwriter, Code4Me [6], and JetBrains AI.

3.2 Backend

The backend is a Python application built with modern, scalable technologies. The API is served by **FastAPI** [4], chosen for its asynchronous capabilities, which allow it to handle many concurrent requests efficiently. A critical design choice was to decouple long-running tasks, such as LLM inference and persistence, from each other and from the main API request-response cycle. We achieve this using **Celery** [14], a distributed task queue, with **Redis** [13] as a message broker. When a completion request arrives, the API immediately places it on the task queue for both the inference engine and the database manager. A Celery worker picks up the task, performs the model inference, and publishes the result. This asynchronous design ensures the user's IDE never hangs while waiting for irrelevant processes to the actual completion to finish. The infrastructure provides support for both WebSocket and HTTP-based requests, which in turn gives the front-end developer a lot of flexibility in how they want to implement features.

All data, including user profiles, telemetry, and suggestion metadata, is stored in a **PostgreSQL** [12] database, with **SQLAlchemy** [2] as the ORM to ensure data integrity and adherence to ACID principles. A graphical depiction of the processes and responsibilities of the backend components has been presented in part (a) of Figure 1.

3.3 Frontend

The frontend architecture is central to Code4Me V2's effectiveness as both a user-focused application and a research instrument. It is centered around a highly modular system. In the design, the core unit is defined by a Module. A module can be responsible for collecting data for the completion, running some update process, or running an arbitrary post-acceptance process. Modules follow a hierarchical system. To gain a better idea, the **Module Manager** is the central coordinator that receives data collection requests from the plugin's main logic. **Aggregators** are modules that directly receive dispatched messages from the manager. These are responsible for a specific class of data. Each aggregator invokes one or more individual, self-contained **Modules**. For example, the `Behavioral Telemetry Aggregator` might call the `Typing Speed` and `Time Since Last Completion` modules. Keep in mind that Aggregators themselves are modules, so this *self-contained module* can also be an aggregator. This composable structure allows researchers and developers to have highly complex interaction systems.

This design makes the system exceptionally extensible. To add a new piece of telemetry (e.g., tracking copy-paste events), a researcher only needs to implement a simple module interface and register it in the configuration. No other part of the system needs to be modified; this drastically lowers the engineering effort required to design new experiments. In line with the guiding principles of IntelliJ Action systems [2], actions initiated by the plugin occur exclusively when a user submits a completion request, whether inline or through the chat module. We classify module actions into two categories: the data collection loop and the after-insertion loop (these actually also correspond to two separate methods in the Module interface). These represent stages within the request lifecycle where modules can integrate and execute necessary actions. The complete lifecycle of this process is detailed in part b of Figure 1.

[2] IntelliJ Action System principles

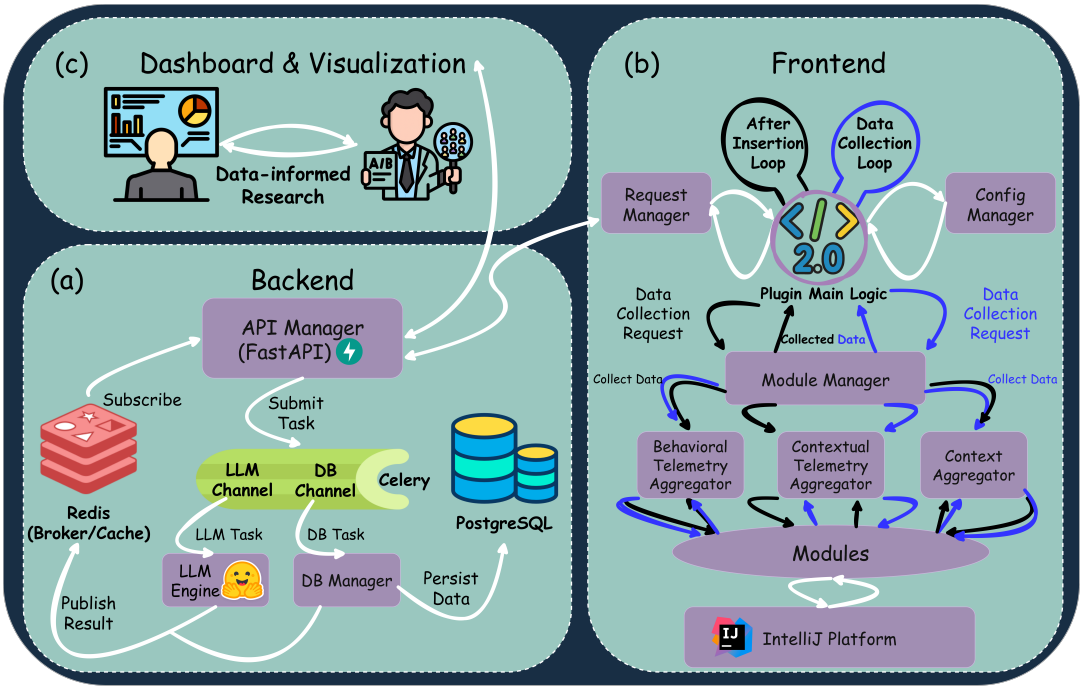


Figure 1: Graphical depiction of the high-level architecture of Code4Me V2. (a) represents the backend processes of the server. (b) gives an architectural view of the implemented IntelliJ Plugin for Code4Me V2, and (c) represents the admin dashboard.

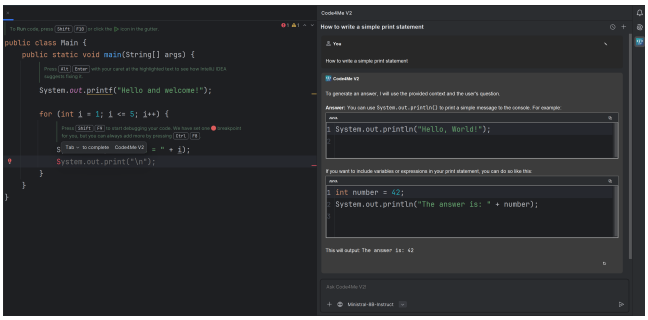


Figure 2: Plugin screenshot (inside IntelliJ IDEA)



Figure 3: Model calibration pane in the analytics dashboard

3.4 Analytics Platform

The analytics subsystem transforms telemetry collected from various stack layers into research-quality metrics and summaries. The event streams, which include metaqueries, model generations, and contextual and behavioral telemetry, are stored in PostgreSQL and examined through SQL window functions and time bucketing methods. The public API provides endpoints that calculate time-series aggregates, acceptance statistics with confidence intervals, and latency percentiles. This supports both descriptive and comparative analysis of model performance in realistic scenarios. Role-based access control is uniformly enforced; non-admin users have access to their analytics, whereas admins can request cross-user comparisons and handle experiments and configurations. Model-specific endpoints allow for side-by-side comparisons of acceptance, latency, and confidence distributions, along with calibration analyses. These metrics allow for studies into confidence validity and the correlation between predicted confidence and actual acceptance. Additionally, the system supports A/B testing and configuration management to enable controlled platform experiments. Only admins can access study endpoints, which are responsible for lifecycle management and random server-side configuration assignments.

4 Preliminary Evaluation

The platform has been evaluated for performance, stability, and its suitability for research. The system's architecture was designed to minimize client-side overhead by offloading all model inference to

the server. To assess the server-side latency, we create a pipeline using a recognized Fill-in-the-Middle task dataset [3] and continuously send requests to the server to measure the average delay. In the same manner, for the chat functionality, we use a well-known program synthesis dataset [4] to instruct the model to clarify and execute the task, mimicking a scenario where a user is inquiring from within an IDE. For code completion, we use *deepseek-coder-1.3b-base*, and for the chat model, we use *Ministral-8B-Instruct*. End-to-end latency for code completion requests, from the trigger event in the IDE to the rendering of ghost text, is consistently 186.31 ms (± 139.50), with an average of 18.66 tokens, which is well within the threshold for a non-disruptive user experience. For chat completions, the latency is 8369.78 ms (± 840.48) with an average of 277.08 tokens.

To evaluate the plugin's usability and research extensibility, we conducted a two-phase user evaluation. First, a formative user study was conducted with four expert researchers from the AI for Software Engineering (AI4SE) faculty. Participants were deliberately chosen based on their familiarity with AI4SE and their frequency of using AI coding assistants. They were then asked to complete a basic multi-file programming task, which involved creating a 'Shape' interface and 'Circle' and 'Rectangle' implementing classes. During this task, they were expected to interact with both the inline completion and chat features.

Feedback was gathered through a semi-structured interview format and a questionnaire. The questionnaire focused on three key areas: **General Usability** (e.g., relevance and timeliness of suggestions), **Setup & Configuration** (e.g., clarity of data collection settings), and **Extensibility for Research** (e.g., modifying the plugin for experiments). The qualitative feedback from this initial study strongly validated the platform's core premise: participants unanimously praised the modularity and extensibility, and expressed confidence in their ability to adapt the tool for their own research workflows. The study also identified areas for improvement, primarily regarding the intuitiveness of the module management UI and inconsistencies in the timing of automatic suggestions.

Based on the results of this formative study, we made several iterations on the initial design. We then conducted a secondary evaluation with four daily users using the same protocol. Our findings revealed that the individuals did not echo the issues previously identified by experts. Furthermore, fewer participants mentioned timing issues with the suggestions, experiencing fewer challenges with frequency and response time. Nonetheless, one user noted that the suggestions could benefit from a slightly increased speed. Lastly, a participant requested the inclusion of an *Agent* feature in the plugin, highlighting (1) a rapidly evolving domain and (2) the necessity for quick adaptation from academia, the later of which is now facilitated by Code4Me V2.

5 Limitations and Future Work

Our preliminary analysis also identified several limitations. As noted in our user study, the user interface for experiment configuration, while powerful, requires a learning curve that could be steep for non-expert users. While improved in the second round of iterations, there is still room for improvement, both in terms of how settings are shown and whether settings should be shown at all. While the platform is performant for research purposes, its completion speed and model quality do not yet match that of well-resourced commercial tools. The current context retrieval, while rather effective in single-file and multi-file settings, still requires attention to be on par with commercial counterparts. A primary objective is the implementation of project-wide context retrieval to enhance the accuracy of code suggestions. Moreover, continuously improving and modifying the current framework is crucial to continue meeting the community's needs (see the request for an agent feature in section 4). Therefore, we invite community members to engage with this project to ensure mutual benefit. We will be overseeing the repositories and welcome community contributions.

[3] Santacoder FIM task.
[4] We use the MBPP dataset.

6 Conclusion

Code4Me V2 is an open-source, modular platform designed to lower the barrier for empirical research in AI-assisted software development. Our preliminary evaluations demonstrate that the platform is both practical for day-to-day use and adaptable for experimental needs. By providing a transparent, controllable, and extensible tool, we enable the academic community to rigorously investigate the complex dynamics of human-AI collaboration in programming. We believe Code4Me V2 can serve as a foundational piece of infrastructure for the next wave of AI4SE research.

References

[1] Inc. Amazon Web Services. 2023. Amazon CodeWhisperer. AWS presentation. https://pages.awscloud.com/rs/112-TZM-766/images/ADC-C3%20-%20Amazon%20CodeWhisperer.pdf

[2] Mike Bayer. 2025. *SQLAlchemy: The Database Toolkit for Python.* Retrieved Sep 15, 2025 from https://www.sqlalchemy.org/

[3] Cursor 2025. *Cursor - The AI Code Editor.* Retrieved Sep 15, 2025 from https://cursor.com/

[4] FastAPI 2025. *FastAPI: Fast and High-performance Web Framework for APIs.* Retrieved Sep 15, 2025 from https://fastapi.tiangolo.com/

[5] GitHub 2025. *GitHub Copilot.* Retrieved Sep 15, 2025 from https://github.com/features/copilot

[6] Maliheh Izadi, Jonathan Katzy, Tim Van Dam, Marc Otten, Razvan Mihai Popescu, and Arie Van Deursen. 2024. Language models for code completion: A practical evaluation. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[7] JetBrains 2025. *JetBrains AI | Intelligent Coding Assistance, AI Solutions, and More.* Retrieved Sep 15, 2025 from https://www.jetbrains.com/ai/

[8] Irina Mariasova, Yanina Ledovaya, Olga Lvova, and Mikhail Bogdanov. 2024. Developers save up to 8 hours per week with JetBrains AI Assistant. JetBrains Blog. https://blog.jetbrains.com/ai/2024/04/developers-save-up-to-8-hours-per-week-with-jetbrains-ai-assistant/

[9] Hussein Mozannar, Gagan Bansal, Adam Fourney, and Eric Horvitz. 2024. Reading between the lines: Modeling user behavior and costs in AI-assisted programming. In *Proceedings of Conference on Human Factors in Computing Systems.*

[10] Ruchika Pandey, Prabhat Singh, Raymond Wei, and Shaila Shankar. 2024. Transforming Software Development: Evaluating the Efficiency and Challenges of GitHub Copilot in Real-World Projects. (2024). arXiv:2406.17910 [cs.SE]

[11] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. 2023. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. (2023). arXiv:2302.06590 [cs.SE]

[12] PostgreSQL Global Development Group 2025. *PostgreSQL: The World's Most Advanced Open Source Relational Database.* Retrieved Sep 15, 2025 from https://www.postgresql.org/

[13] Redis 2025. *Redis - The Real-time Data Platform.* Retrieved Sep 15, 2025 from https://redis.io/

[14] Ask Solem. 2025. *Introduction to Celery.* Retrieved Sep 15, 2025 from https://docs.celeryq.dev/en/latest/getting-started/introduction.html

[15] Sourcegraph 2025. *Cody | AI coding assistant from Sourcegraph.* Retrieved Sep 15, 2025 from https://sourcegraph.com/cody

[16] StackOverflow 2025. *AI | 2025 Stack Overflow Developer Survey.* Retrieved Sep 15, 2025 from https://survey.stackoverflow.co/2025/ai#1-ai-tools-in-the-development-process