

Teachers and Developers Building Together: The ExLAIM Six-Week Co-Design Program for Developing AI-Supported Classroom Tools

BENYAMIN TABARSI, North Carolina State University, United States

SAM GILSON, North Carolina State University, United States

MOHAMED FAAZ MOHAMED IQBAL, University of Pittsburgh, United States

SUSANNA QUAYLE, North Carolina State University, United States

JOEY NICOLE ACKERMAN-LOWERY, Fayetteville Technical Community College, United States

ADITYA BASARKAR, North Carolina State University, United States

DANIEL BRICENO, North Carolina State University, United States

MATHEUS MARQUES, The College of New Jersey, United States

LOKESH SHARMA, UC Irvine, United States

XIAOYI TIAN, Kennesaw State University, United States

JOEY HUANG, North Carolina State University, United States

DONGKUAN XU, North Carolina State University, United States

TIFFANY BARNES, North Carolina State University, United States

Generative AI offers new ways to support teaching and build software. However, faster software development does not determine what is worth building for classrooms. We present ExLAIM, a six-week program bringing nine K–12 teachers and five undergraduate developers together to co-design AI-supported tools within an existing educational platform. Following two weeks of developer preparation, teams spent four weeks building, testing, and revising five tools. Drawing on program records, developer interviews, and teachers’ School AI Plans, we examine how classroom concerns shaped implementation. Teacher feedback informed decisions about course materials, information sharing, and review of generated content. Developers described AI assistance with interpreting requests and writing code alongside challenges with debugging and time to act on feedback. Teachers connected plans for school use to curricular needs, reliable software, institutional approval, and continuing support. These experiences inform practical guidance for sustaining teacher–developer collaboration as classroom ideas become working tools.

CCS Concepts: • **Human-centered computing** → **Participatory design**; • **Applied computing** → **Interactive learning environments**; • **Software and its engineering** → *Collaboration in software development*.

Additional Key Words and Phrases: Co-design, Teacher–developer collaboration, K–12 education, Generative AI, AI-supported educational tools, Undergraduate developers, AI-assisted software development

Authors’ Contact Information: Benyamin Tabarsi, btaghiz@ncsu.edu, Game2Learn Lab, North Carolina State University, Raleigh, North Carolina, United States; Sam Gilson, Game2Learn Lab, North Carolina State University, Raleigh, North Carolina, United States; Mohamed Faaz Mohamed Iqbal, University of Pittsburgh, Pittsburgh, Pennsylvania, United States; Susanna Quayle, North Carolina State University, Raleigh, North Carolina, United States; Joey Nicole Ackerman-Lowery, Fayetteville Technical Community College, Fayetteville, North Carolina, United States; Aditya Basarkar, North Carolina State University, Raleigh, North Carolina, United States; Daniel Briceno, dabricen@ncsu.edu, North Carolina State University, Raleigh, North Carolina, United States; Matheus Marques, The College of New Jersey, Trenton, New Jersey, United States; Lokesh Sharma, UC Irvine, Irvine, California, United States; Xiaoyi Tian, xtian5@kennesaw.edu, Department of Information Technology, Kennesaw State University, Marietta, Georgia, United States; Joey Huang, joeyhuang@ncsu.edu, North Carolina State University, Raleigh, North Carolina, United States; Dongkuan Xu, dxu27@ncsu.edu, Computer Science, North Carolina State University, Raleigh, North Carolina, United States; Tiffany Barnes, tmbarnes@ncsu.edu, Computer Science, North Carolina State University, Raleigh, North Carolina, United States.

1 Introduction

Generative AI (GenAI) is entering educational practice and software development at the same time. Teachers are exploring it to prepare classroom materials and to support assessment and feedback. Developers are using it to generate, explain, debug, and revise code. For educational technology, the second change does not settle the first — *making software faster to build does not determine what is worth building for a classroom*. Teachers bring knowledge of students, instructional goals, classroom practice, and school constraints; developers bring experience turning ideas into systems that run. Co-design is an established way of letting both kinds of expertise shape the same technology [6, 13, 24].

Teacher–developer collaborations have carried ideas through prototyping, revision, and classroom use [6, 21, 22, 24]. Nevertheless, a review of 108 human-centered learning analytics and AI-in-education studies found uneven stakeholder involvement, with passive involvement exceeding active involvement in the combined evaluation, refinement, and production phase [3]. Recent educational GenAI research has examined teachers’ design priorities, auditing practices, customization needs, and chatbot authoring [14, 25, 30, 33, 34]. We extend this work by examining the continuity of teacher–developer collaboration through implementation: how classroom concerns enter working software, change through testing and revision, and inform plans for school use.

We examine this process through ExLAIM, a university program comprising six active weeks around MerryQuery, an AI-powered educational platform designed and piloted beforehand. Five undergraduate developers spent two weeks learning the codebase and GenAI development practices. Following a one-week break, nine K–12 teachers joined for four weeks of co-design and implementation in five stable teams, each with one developer and one or two teachers. Teams moved from classroom concerns through implementation, teacher and cross-team testing, revision, and presentation. Two development leads, a senior PhD coordinator, four faculty members, and industry practitioners supported this work. The program combined sustained teacher participation with undergraduate implementation within an existing platform and a layered mentoring structure.

GenAI was both part of the tools being designed and an optional development resource. The program anticipated that GenAI, preparation, the shared platform, and mentorship would help make implementation feasible within four weeks. Developers chose whether and how to use GenAI; their accounts describe perceived assistance without establishing its effect on development time or quality. We treat GenAI as a condition of the collaboration rather than a separately evaluated intervention.

This practice reflection draws on program materials, project papers, user guides, developer interviews, and teachers’ School AI Plans to give a situated account of the collaboration, working prototypes, and intended next steps. We combine a program account (Section 3), tool descriptions (Section 4), and thematic analyses of interviews and plans (Section 5). We contribute three accounts for HCI researchers and practitioners organizing sustained teacher–developer collaboration:

- A program structure supporting co-design through implementation: developer preparation, stable teams, reflection, mentorship, code review, and cross-team testing (Section 3), with arrangements to retain or revise (Section 6).
- Five cases tracing classroom needs into working AI-supported tools, documenting decisions about workflows, source material, privacy, review, and automation, including capabilities declined or deferred (Section 4).
- Developer experiences and teachers’ plans for future AI use, analyzed from interviews and School AI Plans (Section 5) and guided by two research questions:

- **RQ1.** How do undergraduate developers describe the effectiveness, challenges, and enabling conditions of the teacher–developer co-design process?
- **RQ2.** How do K–12 teachers envision integrating AI-supported tools and practices into their schools after participating in ExLAIM?

2 Related Work

We situate ExLAIM in teacher participation in design, programs pairing undergraduate developers with stakeholders, GenAI as artifact and development tool, and classroom integration.

2.1 Teachers as Co-Designers of Educational Technology and AI

Participatory design holds that the people who will use a technology should take part in its design [9, 28]. Educational technology research has applied this principle by involving teachers in the design of learning environments, assessment tools, and learning analytics systems [6, 13, 24]. Teachers contribute knowledge of curricula, students, and classroom practice, helping teams connect technical possibilities to instructional purposes. This work also depends on how participation is organized: who can shape a proposal, how ideas are revised, and how teachers’ concerns inform implementation.

Sustained collaboration with software developers has clear precedents. In Project WHIRL, seven teachers worked with researchers and developers on handheld formative-assessment software through requirements, prototypes, and classroom testing over a school year [24]. Cober et al. [6] describe two-month collaborations in Canada and Singapore in which teachers worked with researchers and developers on pedagogical and technical design, and Musaeus et al. [21] describe six teachers, two developers, and four researchers developing CoTinker prototypes and implementing them in classrooms with 83 students. These studies show that teachers can contribute through development and use, as well as during initial problem framing.

Other work examines how teacher participation reaches technical decisions and classroom practice. Holstein et al. [13] describe the iterative co-design, development, and classroom evaluation of a real-time classroom orchestration tool, prototyped through replayed student-interaction data and teacher role-play [12, 13]. Nicholson et al. [22] report a two-year co-teaching collaboration that produced three digital classroom tools, connecting design to technology integration. These are precedents for examining implementation as part of an ongoing collaboration.

The broader literature nevertheless shows uneven involvement. In a review of 108 human-centered learning analytics and AI-in-education studies, Alfredo et al. [3] found that exploration, concept generation, and evaluation, refinement, and production received more attention than planning or launch and monitoring. In the combined evaluation, refinement, and production phase, studies reporting passive stakeholder involvement outnumbered those reporting active involvement (31% versus 25%). These categories combine several activities and stakeholder groups; they do not isolate teachers’ influence during coding.

Tan et al. [30], examining ten co-design sessions in which teachers and students defined data requirements with ML engineers, UX designers, and legal practitioners, argue that meaningful participation requires structured support for continuous iteration and co-evaluation across areas of expertise. Building on these precedents, ExLAIM explores how sustained collaborative teams carry classroom concerns through implementation, testing, and revision, with support from development leads, academic mentors, and industry practitioners.

2.2 Programs That Pair Undergraduate Developers with Stakeholders

Capstone and project courses provide established settings for students to build software with real stakeholders. Reviewing 127 articles, Tenhunen et al. [31] found that software-engineering capstones generally lasted one semester, organized students into groups of four or five, often expected a proof-of-concept product, and in a slight majority of courses involved clients external to the course staff. Steghöfer et al. [29] developed a model for planning and reflecting on external stakeholder involvement in project courses, highlighting the importance of creating conditions for effective student–stakeholder collaboration.

Sanusi et al. [27] describe a semester-long research course in which university students developed software for teaching K–12 learners about AI and tested the tools with learners at school-based AI Expos. Ojeda-Ramirez et al. [23] describe a five-week participatory design program with three eleventh-grade students and three high-school teachers who negotiated how GenAI would be used and taught in their classrooms.

ExLAIM adds an account of undergraduate developers and K–12 teachers working together throughout implementation within a shared platform. The undergraduates’ role matters to this account because they were responsible for a tool while still developing their technical and collaborative practice. Their position also allowed them to bring their own experiences as students into discussions with teachers. We examine their accounts of working across these kinds of knowledge alongside the support they received. That support included two recently graduated undergraduate development leads, a senior PhD candidate who coordinated and mentored the work alongside faculty, and industry mentors who reviewed emerging ideas and implementations in weekly meetings. This wider group was part of how the collaboration operated, and must be considered when assessing what the undergraduate developers were able to accomplish.

2.3 Generative AI as Artifact and as Development Tool

Generative AI appears in this literature in two roles. In the first, it is the system being designed. Ravi et al. [25] developed design recommendations for LLM support in project-based learning with teachers, who emphasized support for their professional roles and constraints on classroom integration, and Zhao et al. [34] co-designed community-centered AI auditing tools with 22 public-school educators in Hawai‘i, foregrounding cultural knowledge and collective responsibility. These studies show how educators shape the purposes and criteria for designing and assessing educational GenAI.

Customization and teacher authoring bring additional work into view. Among ten categories of chatbot customization, the university STEM instructors surveyed by Hou et al. [14] consistently prioritized alignment with course materials and pedagogical strategies, while other priorities varied across teaching contexts. MerryQuery’s retrieval over teacher-supplied materials provided one resource for addressing content alignment; how each tool would support a classroom workflow remained a design question. In interviews with seven K–12 teachers who created pedagogical chatbots, Yoo et al. [33] found that novices struggled with initial design and implementation, while experienced teachers continued to face technical challenges and difficulties analyzing conversational data. These accounts highlight the work required to translate instructional intentions into chatbot behavior, including when teachers have access to authoring tools.

In the second role, GenAI is part of how software is written. Product team members interviewed by Li et al. [17] described faster iteration and broader participation alongside problems with code reliability, integration, and overreliance, and a controlled study of developers new to coding agents by Chen et al. [4] highlights both the capabilities of agents and the work required to understand and supervise them. This literature suggests that generating code changes the distribution of development effort while leaving substantial work in directing, checking, and integrating the result.

For ExLAIM, these findings inform the expectation that GenAI could help undergraduate developers attempt substantial implementation during a short program. They also highlight the importance of review and technical support, which ExLAIM provided through code review, teacher testing, and cross-team testing. Our developer interviews examine how undergraduate developers describe using GenAI while interpreting teacher feedback and addressing implementation demands within this mentoring structure. These accounts support reflection on the collaboration without isolating GenAI’s effect on development speed or software quality.

2.4 From a Co-Designed Tool to a Classroom

Classroom integration extends beyond functioning software. Sustained co-design and co-teaching studies examine this transition directly [21, 22, 24]. Across the broader corpus reviewed by Alfredo et al. [3], however, only 9% of studies addressed launch and monitoring. This finding indicates limited coverage of that phase within human-centered learning analytics and AI-in-education research; it does not establish that co-design generally ends before classroom use.

Research on GenAI use identifies several conditions that matter for integration. Teachers in an early-adopting U.S. district negotiated GenAI use in relation to their values, district policies, professional norms, and relationships [8]; computing instructors faced barriers while adapting courses to students’ use of commercial GenAI tools [16]; and K–12 educators co-designing an AI microcredential requested accessible activities and timely resources for use throughout the school year [15]. Together, these studies show that pedagogical fit, institutional conditions, and continuing support matter alongside technical functionality. Design can address and anticipate these conditions, while their consequences also unfold in use.

At the end of ExLAIM, teachers wrote School AI Plans describing what they intended to teach and when, the tools they anticipated using, their institutional permission status, alternatives if MerryQuery was unavailable, and the support they needed. These plans let us examine how teachers connected their program experience to anticipated school use. They provide evidence of intentions and perceived conditions at the program’s close.

3 Contribution 1: Program Structure

This section presents a retrospective account of ExLAIM written by authors involved in organizing and running the program, drawing on their direct experience, program schedules, and activity materials. ExLAIM comprised two weeks of developer preparation and, after a one-week break, four weeks of teacher–developer co-design in MerryQuery.

3.1 MerryQuery as the Shared Technical Setting

MerryQuery had been iteratively designed and piloted before ExLAIM. Instructors could create courses, upload materials, enroll students, and review student interactions (Figure 1b). Students could ask questions through a conversational interface (Figure 1a), with responses generated using retrieval-augmented generation over instructor-provided materials. The platform maintained interaction logs and displayed references to source materials alongside generated responses.

Teams extended this shared codebase and its existing workflows, sometimes substantially reworking them. Before ExLAIM, a development lead oversaw migration from MongoDB to PostgreSQL and prepared development and testing environments. Teams were encouraged to incorporate GenAI into their tools, while developers chose whether and how to use it for development. No common GenAI development workflow was prescribed.

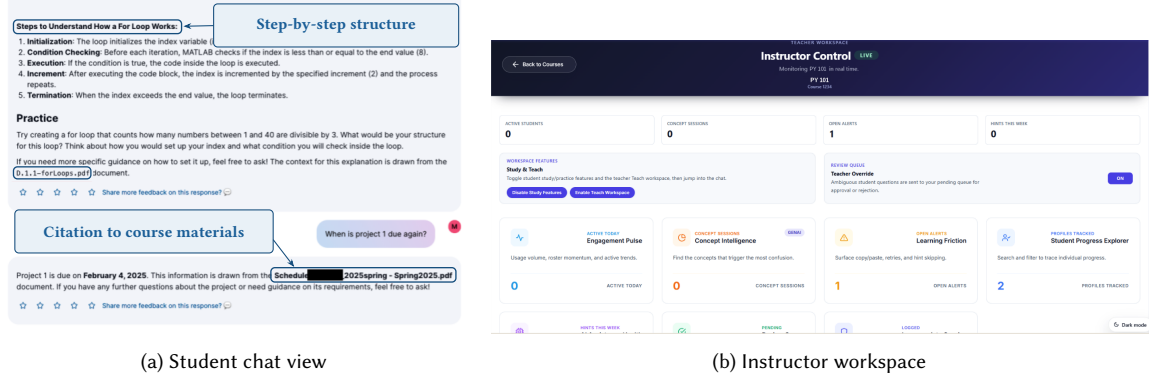


Fig. 1. MerryQuery, the shared technical setting: students asked questions in a chat view, and instructors followed their course in a separate workspace. Course identifiers are redacted; the course shown (“PY 101”) is a demonstration course used for testing.

3.2 Participants and Team Formation

ExLAIM included nine K–12 teachers and five undergraduate developers. Eight teachers taught grades 9–12 and one taught grades 6–8, across computing, digital media, business, career exploration, and mathematics. Reported teaching experience ranged from 5 to 22 years; one teacher’s experience was unspecified. Four developers were aged 18–24 and one was aged 25–34; they ranged from sophomore to senior, studied computer science or computer programming, and reported daily or weekly AI use. Table 2 in Appendix A gives individual backgrounds for both roles, ordered by team.

Teachers joined in assigned groups, watched developers demonstrate the improvements implemented during the preparation phase, and indicated developer preferences through a selection form, which the program coordinator used to match a developer to each teacher group. Four teams comprised one developer and two teachers; Team 1 comprised one developer and one teacher.

3.3 Engineering Support and Mentorship

Two recently graduated undergraduates served as development leads, combining work on the shared platform with support for the undergraduate developers: one focused on software architecture, shared infrastructure for LLM-based features, tooling, code review, and testing, and the other on deploying and running the platform on the server.

A senior PhD candidate coordinated and ran the program alongside faculty, discussed design and technical questions with teams, and sometimes reviewed or contributed to implementation. Four faculty members provided ongoing feedback, with at least one present during each scheduled working block.

Industry practitioners contributed through both cohort sessions and recurring team meetings. An industry guest speaker opened the co-design phase; from its second week, each team selected a mentor with relevant expertise and met with that mentor weekly to discuss ideas, demonstrate emerging implementations, and receive feedback; and further weekly guest speaker sessions introduced AI development practices and applications.

Code review provided another point of contact across teams. Before a proposed change was incorporated into MerryQuery, two other developers reviewed it; in most cases, at least one reviewer was a development lead. Undergraduate implementation thus took place within a shared process of technical assistance and review, alongside teacher feedback on the tools’ design and intended classroom use.

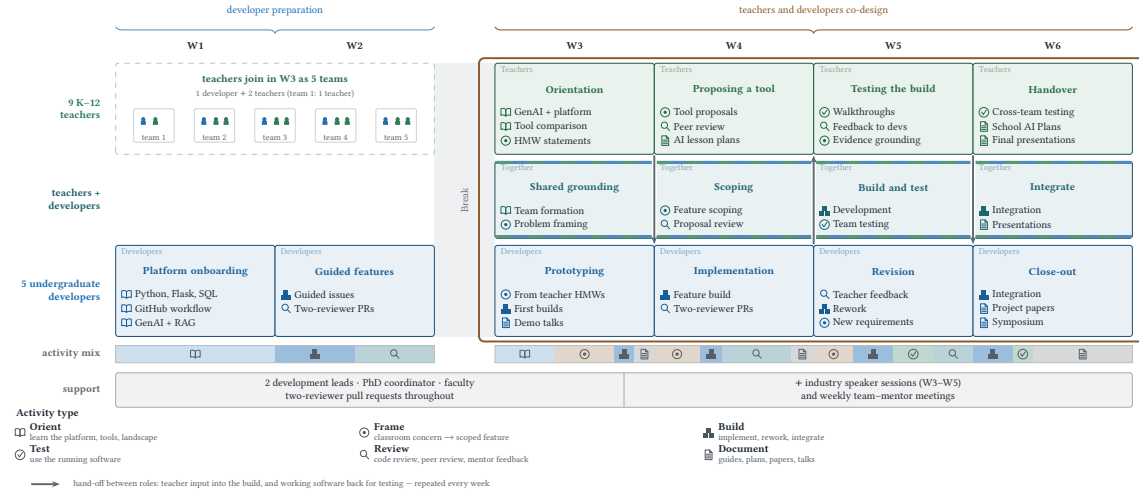


Fig. 2. ExLAIM's six active weeks. Weeks 1–2 focused on developer preparation in MerryQuery. After a one-week break, teachers joined for weeks 3–6, moving from classroom problem framing to recurring implementation, testing, feedback, and revision. Development leads and academic mentors supported the work throughout; industry speaker sessions took place in weeks 3–5 and weekly industry team mentoring in weeks 4–6. Alternating arrows between weeks 3–4, 4–5, and 5–6 represent the recurring weekly exchange of teacher input into development and working software returned for testing. The activity-mix strip summarizes listed activities by type and does not quantify sessions or time.

3.4 Program Activities

Weekdays included two one-hour morning working blocks, two afternoon blocks, and a half-hour end-of-day reflection on progress and plans. During co-design, teachers and developers worked together approximately 9 a.m.–3 p.m.; developers continued implementation until approximately 5 p.m. Team development occupied longer periods as the program progressed. Figure 2 summarizes activities across teacher, joint, and developer lanes. Its activity-mix strip represents listed activities, not time allocation or session counts.

3.4.1 Developer Preparation: Weeks 1–2. During week 1, developers set up MerryQuery, practiced GitHub collaboration, and learned its Python, Flask, and PostgreSQL components. ExLAIM also provided AI sessions covering GenAI foundations, prompt engineering, agents, and retrieval-augmented generation. Developers explored existing functionality, traced dependencies, and generated ideas for platform changes. In an activity adapted from the Nominal Group Technique [20], developers recorded ideas individually on sticky notes, then presented, discussed, and openly voted on each in turn, assigning it to one of four local priority labels inspired by MoSCoW prioritization [1]: *Very Important*, *Moderately Important*, *Not much important*, and *Toss it!* (Figure 3). Prioritized ideas were transferred to a shared GitHub Projects board for planning and tracking. During week 2, developers fixed bugs, completed guided changes, and implemented selected ideas tracked on GitHub Projects. They practiced task documentation, code review, and presentations. This produced a developer backlog and smaller improvements before teachers joined.

3.4.2 Teacher–Developer Co-Design: Weeks 3–6. The co-design phase began with teacher-oriented introductions to GenAI, prompt engineering, and MerryQuery. Facilitated discussions about AI in classrooms and workplaces gave teachers and undergraduate developers opportunities to share their perspectives, including where AI use might be useful

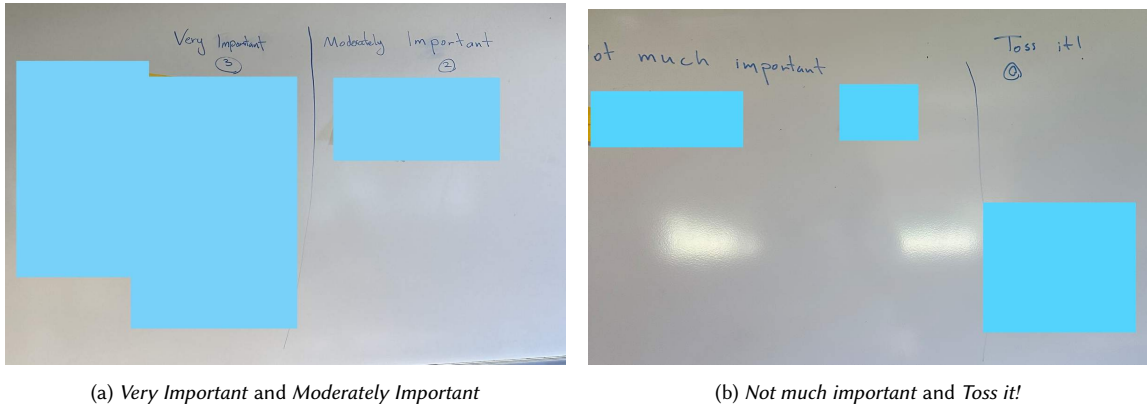


Fig. 3. The prioritization board at the close of the week-1 idea-generation activity. Developers wrote candidate platform changes on sticky notes, and the group placed each note under the priority heading that received the most votes. Panels (a) and (b) are the left and right halves of the same board; the initial letter of *Not much important* falls outside the frame of panel (b). Note text is redacted.

or inappropriate, and learn about one another’s roles and interests. Following team formation, participants identified classroom problems, developed proposals, and built and revised tools. The activities described below overlapped as teams moved between design and implementation.

Classroom problem framing and proposals. Teams worked through design-thinking activities, articulated classroom-focused problem statements, and examined educational AI systems such as MagicSchool and Khanmigo. They presented their problem statements for feedback before deciding what to pursue. As ideas developed, teams examined other educational technologies and searched for research and theoretical frameworks relevant to their proposed tools. Teams documented proposed features using a two-sided proposal form (Appendix D). The teacher side articulated the classroom design focus; the developer side described the technical approach, feasibility, and build plan.

Implementation, feedback, and revision. Developers built working versions of team features in MerryQuery, and teachers then interacted with them and provided feedback. Progress presentations and peer-review activities brought proposals and implementations to the whole program. Teachers also led a weekly *AI in the News* session, selecting and presenting developments in educational AI to the cohort.

Teams tested their own implementations throughout the phase, and during the final two weeks participated in a structured cross-team testing rotation in which members of other teams tested each tool using a common worksheet and reported feedback through peer-review discussions and presentation slides, providing further opportunities for revision. These activities complemented the code review and industry mentoring described in Section 3.3.

Documentation and plans for school use. The final two weeks also included work on evidence grounding, integration, documentation, and preparation for presentation. Developers completed integration work, project papers, and teacher-facing user guides. Teachers developed individual School AI Plans describing how they anticipated using AI in their educational settings.

The program concluded with presentations in which all five teams demonstrated their work and received feedback, and each developer presented a poster at a university-wide undergraduate research symposium. Appendix B, Table 3,

Table 1. The five teams' classroom concerns, tools, and selected changes during implementation.

Team # and tools	Classroom concern	Selected implementation changes
1: Team Mode	Help-seeking during independent work and opportunities for peer interaction	Replaced message selection with an editable summary; added teacher escalation and controls for group support
2: Course-grounded preparation and Collaborative Boards	Preparation using course materials and access to shared classroom activities	Added source selection; replaced manual board links with managed creation and access; made the AI Coach board-aware
3: Daily Activity	Daily communication and preparation of warm-ups and exit tickets	Added editable drafts, source selection, cognitive-demand settings, and required-response options
4: Course Materials, Feedback Generator, and Teaching Guidance	Material management and individualized feedback	Added asynchronous file processing; separated student submissions from course retrieval; presented feedback as an editable draft
5: Practice Sets and Quizzes	Timely visibility into students' difficulties with practice topics	Developed topic-level analytics and extended the work to quiz creation and teacher controls

lists ExLAIM materials and their contents to help readers assess this study and adapt them for future teacher–developer collaborations.

4 Contribution 2: Tracing Classroom Needs into AI-Supported Tools

Each team developed one primary tool during four weeks of co-design¹. We summarized the five developer-authored project papers, checked the descriptions against the teacher-facing guides, and returned each project description to its developer for revision and verification. The descriptions trace classroom needs, implementation changes, and final scope, as also summarized in Table 1.

4.1 Team 1: Team Mode

The first team began with concerns about students seeking help and working together. A teacher described students becoming stuck during independent work but hesitating to ask for assistance. The team also questioned how to support interaction among classmates when students increasingly worked individually with AI. These concerns led to **Team Mode**, which connected private AI-supported work to help from a teacher or peer. Figure 4 shows its teacher interface.

An early implementation asked students to select individual messages from their AI conversation when requesting help. The team replaced this with a single editable summary because selecting a message added work when a student was already struggling. Students reviewed and approved what they shared, while the full private conversation remained outside the request. Teacher review also prompted changes to peer support. Peer requests and peer rooms were made anonymous from the students' perspective while remaining identifiable to authorized staff, and the team added an escalation path so that an unanswered peer request could be redirected to a teacher. The industry mentor also reviewed Team Mode using demonstration data representing a full class.

As the tool expanded to teacher-created groups, teachers could monitor conversations, intervene, and determine how much assistance an AI facilitator could provide, and the implementation supported private instructional groups based on learning needs known to teachers. Group information was restricted to authorized staff, and the interface avoided displaying diagnoses, individualized education program (IEP) or 504-plan labels, or medical information. The

¹To maintain focus, we exclude non-primary tools and improvements.

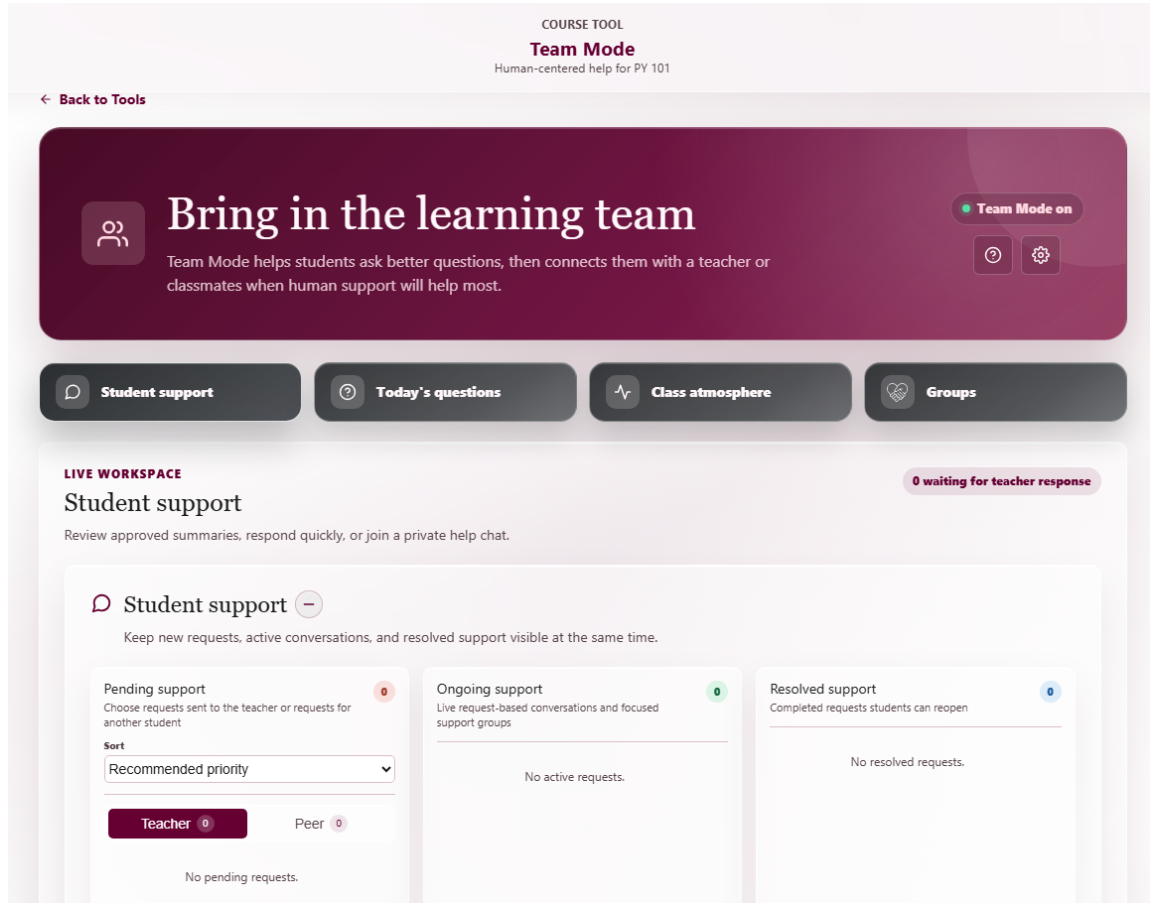


Fig. 4. The teacher interface for Team Mode.

design drew on the zone of proximal development [32], the ICAP framework [5], and structured peer tutoring [10]. Appendix C details the grouping strategies and teacher controls.

The concern about students working alone with AI thus became a series of decisions about what information could leave a private conversation, when help should be moved to another person, and how teachers could supervise that interaction. The developer's reflection similarly identified the difficulty of connecting pedagogical goals to privacy, permissions, database state, interface design, and testing.

4.2 Team 2: Course-Grounded Preparation and Collaborative Boards

The second team's teachers wanted to use MerryQuery for course preparation and live classroom collaboration. The team extended flashcard generation to allow teachers to select specific course files, developed lesson-planning support with citations, and built **Collaborative Boards** via a Miro integration. Teachers could create an activity in MerryQuery, distribute a short access code, and view a snapshot of participation without requiring students to create

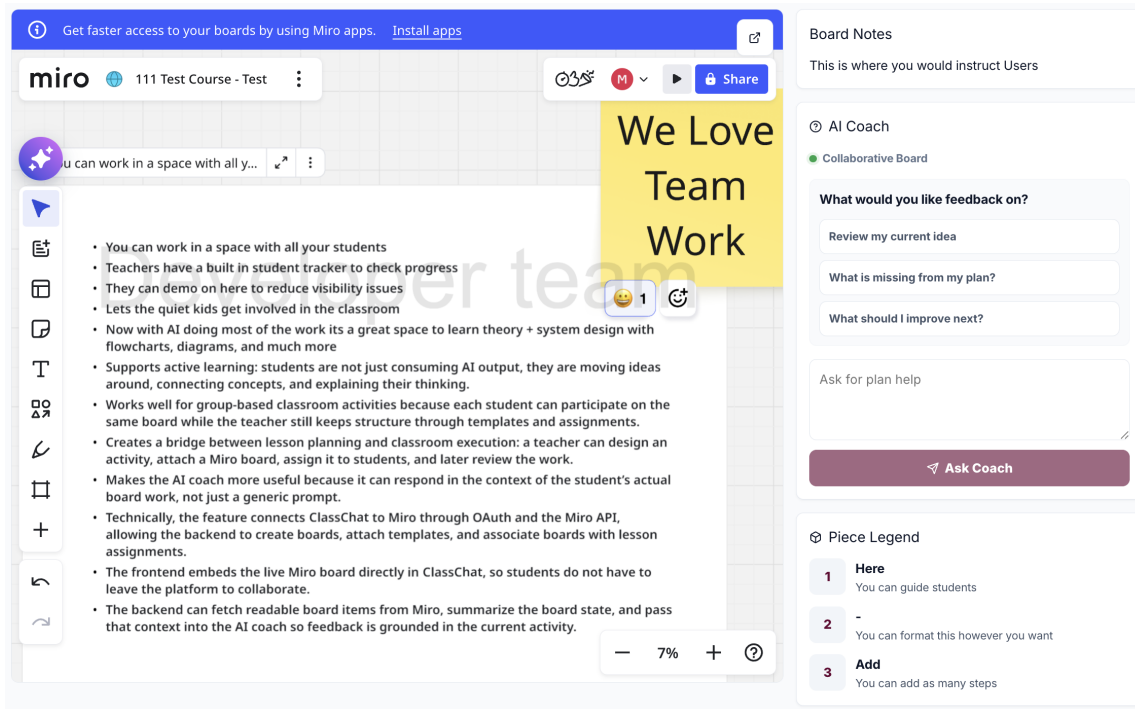


Fig. 5. The student view of a Collaborative Board, with the embedded Miro board beside the board-aware AI Coach.

Miro accounts. Across these tools, teachers' course materials and activity instructions helped define the context available to AI-supported assistance.

The board workflow changed substantially during implementation. The initial version relied on manually entered embed links, leaving part of the setup outside MerryQuery and sometimes directing students to Miro permission screens. The team replaced this with managed authorization through OAuth, server-side board creation, and visitor access. A teacher walkthrough of the full workflow prompted further changes: setup was centralized, privacy information was made visible, and students could enter through a short code without encountering a request-access screen. The AI Coach was also made board-aware, drawing on visible board items, saved instructions, and recent conversation context. Figure 5 shows the student view of a Collaborative Board.

The team deferred the automated deletion of remote Miro boards and identity matching between MerryQuery and Miro, which required further work on how resources and identities would be managed across the two systems, and student contribution counts were defined as partial snapshots of activity. The developer's reflection emphasized that permissions, citations, account access, source material, and teacher control shaped whether the generative functionality could fit a classroom workflow.

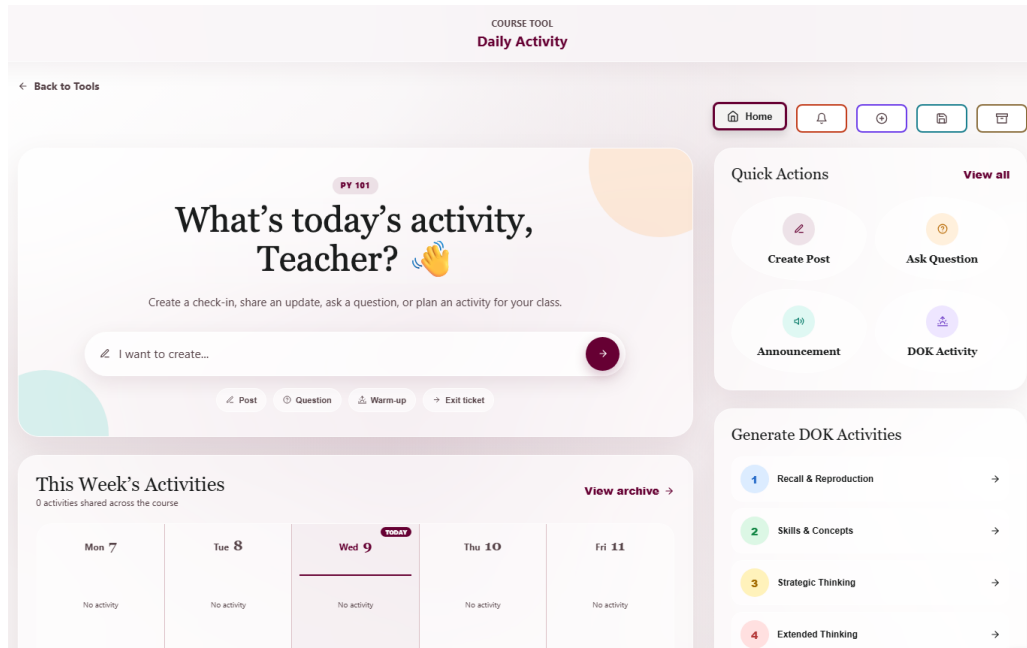


Fig. 6. The teacher interface for Daily Activity.

4.3 Team 3: Daily Activity

Team 3 combined one teacher's need for daily communication and an archive with another's request for customizable warm-ups and exit tickets. **Daily Activity** became a course landing page for announcements, questions, generated activities, saved drafts, and response review. Students saw new posts and recent items they had not viewed or answered.

Early versions supported questions and messages with reactions or written responses. Three testing sessions with the team's teachers and other teachers informed the landing page, required-response options, and organization of earlier material. Generation initially led directly to posting; revisions introduced drafts for editing, saving, and later publication. Teachers could select Depth of Knowledge (DOK) levels and source documents instead of using the full course collection, controlling both generation inputs and published outputs (Figure 6).

The team rejected automated generation and sharing to preserve final teacher approval. Scheduled posting was a separate, late request that went unimplemented due to time constraints. The developer's reflection described classroom concerns being reconsidered as engagement goals became decisions about what students would see and how teachers would prepare and review it.

4.4 Team 4: Course Materials, Feedback Generator, and Teaching Guidance

The fourth team's work addressed material management and individualized feedback. One teacher wanted a simpler way to manage course materials in MerryQuery. The other, who primarily taught project-based courses, described the time required to write detailed feedback for individual students. The team rebuilt **Course Materials** as a workspace with folders, search, drag-and-drop organization, file conversion, Google Drive integration, and visible processing states. It also developed a **Feedback Generator** that produced teacher-editable drafts for feedback based on student work

The screenshot displays the 'Feedback Generator' interface, a 'COURSE TOOL' for 'Draft rubric-based feedback for PY 101'. It features a 'Reset form' button in the top right. A green status bar indicates 'Feedback generated from the uploaded work and rubric.' The interface is divided into two main sections: 'INPUTS' and 'OUTPUT'.

INPUTS:

- Student work and rubric:** Includes instructions to upload student work and a rubric, and an optional assignment title field containing 'e.g. Short response reflection'.
- STUDENT WORK UPLOAD * Required:** A section for uploading the student submission, with a 'Select student work' button.

OUTPUT:

- Draft feedback:** Includes 'Edit' and 'Copy' buttons.
- Feedback Draft:** The main content area showing the generated feedback.

Overall rubric alignment:

The student work demonstrates a lack of completion and clarity in addressing the assignment requirements. While the student has attempted to engage with the problems presented, many responses are missing, which significantly impacts their overall performance against the rubric criteria.

Criterion-by-criterion notes:

Completion of Problems: The student has not provided outputs for several problems, including Problems 1, 3, 5, 7, 9, and 10. This lack of completion indicates insufficient effort, as

Fig. 7. The Feedback Generator displaying a draft based on a student submission.

and rubric information, and redesigned **Teaching Guidance** so that teachers could select part of an AI response and request suggestions for explaining it.

Implementation forced the team to distinguish information that initially looked similar at the interface level. Course-material uploads were first handled synchronously, but document conversion, OCR, chunking, embedding, and storage could take long enough to create browser or server timeouts. The team moved these operations to asynchronous processing and made the processing status visible. Student submissions uploaded for feedback were also separated from reusable course materials: a submission could inform a feedback draft without entering the retrieval context available across the course.

Teacher walkthroughs prompted clearer organization for growing file collections and presentation of generated feedback as an editable draft. The feedback and walkthroughs from each teacher during the cross-team testing rotation were used to inform further refinements to the interfaces and workflow. Figure 7 shows the Feedback Generator after draft generation.

Teachers wanted to inspect and revise a draft before sharing it, so the implementation produced an editable output that teachers could copy rather than send automatically. Teaching Guidance similarly remained attached to the text that a teacher deliberately selected, rather than acting independently throughout the student conversation. The developer's reflection emphasized learning to turn informal discussions and requirements into technical specifications and testable software. The broad goal of reducing teacher workload had therefore become much narrower in practice; the team had to decide how uploads should behave, what student information could persist, which outputs would remain editable, and where teacher review should remain in the workflow.

Fig. 8. The teacher interface for Practice Sets.

4.5 Team 5: Practice Sets and Quizzes

Team 5 sought timely visibility into students’ practice difficulties, replacing manual grading and tallying of paper exercises. In **Practice Sets**, teachers uploaded or generated questions, students completed them online, and results populated live topic-level analytics to inform instructional adjustments during or after class (Figure 8). The developer revised the tool through repeated demonstrations and teacher feedback.

A late request extended the work to **Quizzes**. Whereas practice questions could be extracted from course materials, quizzes required a new algorithm to generate original questions, a creation and configuration interface, monitoring during attempts, and teacher controls to lock a student out of an ongoing quiz. Monitoring was limited to binary flags for departures from the active quiz tab and screenshot attempts; it excluded keystroke logs, mouse tracking, and screen recordings. This extension added assessment administration to the original practice-analytics goal.

4.6 Patterns Across the Five Cases

The cases show how classroom concerns became decisions about interactions, information, and the scope of implementation. We identify three patterns across the trajectories. These are observations about the work completed during ExLAIM; classroom use and its consequences are beyond the scope of this account.

Working versions helped teams clarify how their tools should work. Broad classroom goals left practical questions unanswered until teams could try the software. In Team Mode, testing helped clarify what students would share when requesting help and when an unanswered request should reach a teacher. In Collaborative Boards, walkthroughs revealed difficulties students could encounter when accessing a board. In Daily Activity, revisions let teachers choose

which course materials informed generation and review drafts before publication. Feedback came from the teams’ teachers, members of other teams, and mentors, as described in Section 3.3.

The tools gave teachers and students decisions about what AI could use and share. Students approved the summary shared when requesting help through Team Mode. Teachers could review and edit generated activities and feedback before sharing them. Teachers also selected course files for generation, while board content and activity instructions guided the AI Coach. Student submissions uploaded for feedback were kept separate from the course materials available to the AI across the course. These features gave people the opportunity to direct AI assistance, although the study does not establish how they would use that assistance in classrooms.

Some features were deliberately excluded, while others remained unfinished. Daily Activity and the Feedback Generator retained teacher review rather than automatically sharing generated content, whereas Miro board deletion and identity matching remained unfinished for want of integration work, and Team 5 added quiz functionality following a late request. A missing capability could therefore reflect a design decision or unfinished work, and a late request could extend what a team built.

5 Contribution 3: Developer Experiences and Teachers’ Plans for Future AI Use

5.1 Method

We analyzed nine School AI Plans (Appendix E) and semi-structured interviews with all five developers (protocol in Appendix F). Plans were distributed in week 3 and completed by the program’s close; two researchers prepared the responses for analysis (Appendix G). The form invited teachers to use voice typing or AI tools when drafting their responses. Interviews took place two weeks afterward over video call, lasted 22–57 minutes ($M = 40$), and were recorded and transcribed. The protocol covered the developer’s principal tool, its development, cross-tool ideas, and reflection on co-design. Both corpora were de-identified and scanned for residual identifiers.

For RQ1, we only coded responses touching on research and tool comparisons, AI’s development role, unimplemented ideas, process effectiveness and resources, collaboration challenges, and advice. We also coded recurring questions outside the protocol about naming AI as a co-design participant and collaboration with other developers. For RQ2, we coded plan responses about intended AI content and timing, standards alignment, tools and fallbacks, permission status, teachers’ own AI use, career activities, and requested support, following the sections of the form (Appendix E).

An LLM instance extracted developer turns across the full transcripts, retaining verbatim text, timestamps, and provenance annotations. Every extracted turn was machine-checked against its source; two human researchers verified a 20% sample of turn–question assignments and coverage annotations against the transcripts. The 128 turns yielded 154 turn–question pairs. Coding used the full turn, with identical codes across its pairs; per-developer counts used unique turns. Appendix G details preparation, extraction annotations, and uncoded turns.

Thematic analysis. We used an LLM-led, codebook-based coding-reliability approach adapted from Dai et al. [7] and Akpınar et al. [2], informed by Saldaña’s first- and second-cycle coding and analytic memoing [26]. LLMs coded and refined interpretations; one researcher authored an exhaustive set of exemplars, which Dai et al. define as gold-standard examples of the codebook and theme structure the LLM is to produce.

First-cycle coding and refinement. For each corpus, a fresh LLM context produced a familiarization memo from response text alone. Claude Opus 5, at a reasoning effort held constant across coding instances, then used the memos,

research questions, and eight corpus-derived exemplars per corpus to code every response, recording a code, verbatim quotation, definition, method, and research-question tag for each unit. Codes were consolidated using derived inclusion and exclusion criteria and refined through written exchanges between two separately contextualized LLM coders, with unresolved disagreements referred to a fresh-context LLM adjudicator and refinement capped at six rounds. Final codebooks contained 73 plan codes and 56 interview codes. Appendix G details the exemplars, coding types, refinement protocol, programmatic checks, and intermediate counts.

Reliability assessment and human audits. Two additional LLM coders independently applied each finalized codebook as a second round of coding. We interpreted Cohen’s κ for the LLM coders [19]: $\kappa = 0.965$ for plans and $\kappa = 0.981$ for interviews, with all programmatically verified quoted spans matching their sources (Appendix G reports intervals and check counts). These statistics measure LLM agreement, not independent human validation. Two human researchers then re-audited approximately 20% of first-cycle coding rows for conceptual fit, quotation grounding, and theme assignment, and found no disagreement with the proposed codes.

Second-cycle pattern coding and thematic organization. ChatGPT Astra 6 (Extra High effort), chosen as a different LLM family from Claude Opus 5 to reduce overfitting to the codebook, organized and applied codes into categories and themes through second-cycle pattern coding, examining labels, definitions, quotations, memos, variations, and contrary cases. Each code received one primary category and theme. Two human researchers re-audited approximately 20% of code–pattern–theme tuples; following confirmability principles [11, 18], the audit included a researcher who had not authored exemplars and examined agreed decisions to identify possible shared LLM errors. The audit found no substantive problems; researchers made minor wording changes and resolved disagreements through discussion.

5.2 RQ1: Developers’ Accounts of the Teacher–Developer Co-Design Process

We organize developers’ accounts around classroom expertise (Section 5.2.1), AI-assisted development (Section 5.2.2), and the conditions that enabled or limited iteration (Section 5.2.3); theme statements correspond to the second-cycle themes in Appendix H. Each developer is identified by the tool their team built: D1-TeamMode, D2-Boards, D3-DailyActivity, D4-Feedback, and D5-Practice. Statements about teachers and peers are developers’ accounts.

5.2.1 Grounding design and revision in classroom expertise. Teacher expertise guides design through translation, negotiation, and repeated review. D3-DailyActivity described their tool as emerging from ideas teachers brought into the collaboration: “Pretty much everything that I implemented started as my teachers’ ideas first.” D5-Practice described asking teachers what they would like to have, then identifying a shared need across teachers’ subject classrooms; an initial idea for a dashboard connecting teachers and administrators was set aside as unlikely to be useful in everyday teaching. D1-TeamMode traced their own tool’s initial idea to the graduate lead on the research team and described their own experience and their teacher’s classroom needs as helping it grow.

Repeated review made classroom judgment part of the implementation. D5-Practice assigned teachers authority over design because they would use the product, and described developing a version, presenting it to teachers, and returning to change or add what they requested. For D4-Feedback, this meant working with actual student submissions with names removed and a rubric supplied by a teacher, then asking teachers to judge the resulting feedback:

“this is the feedback it generated, [...], is this accurate? [...] is this something that you would say?” (D4-Feedback)

When teachers found the feedback too strict, too lenient, or unlike what they would write, D4-Feedback revised the prompt and generated another version. D5-Practice described turning requests into a list of implementation tasks and explaining when a requested capability exceeded what could be built in the available time.

The review also exposed disagreements about appropriate classroom interaction. D1-TeamMode reported that teachers' concerns about distraction prompted the addition of a pause option, but teachers continued to worry that digital communication could displace in-person communication, whereas D1-TeamMode viewed the tool as supporting students who lacked confidence to speak. D4-Feedback described one teacher's feedback as highly critical and sometimes difficult to absorb, but felt it improved the work, and found that becoming familiar with teachers' philosophies helped subsequent conversations proceed more smoothly.

Design direction draws unevenly on research, comparison, and developer judgment. D4-Feedback used cognitive load theory to explain the motivation for reducing teachers' feedback workload; D1-TeamMode described literature as supporting a decision to continue with an idea; D3-DailyActivity recalled becoming excited about an initial implementation before undertaking sufficient research; and D5-Practice reported little personal literature review but identified a teacher's research contribution. D2-Boards described generating flashcards from teachers' coursework and repeatedly asking "why should someone use it", but felt that comparison with other platforms later encouraged improvements to familiar features without revealing a new problem to address.

5.2.2 Using AI across development while retaining responsibility for its output. AI assists broadly while developers negotiate its usefulness, credit, and control. All five developers reported using AI, and their uses reached beyond writing code. D3-DailyActivity used an agent to explain an unfamiliar codebase and locate the files relevant to the proposed feature. D1-TeamMode used meeting notes to obtain a structured plan for responding to feedback. D5-Practice moved between teachers and Claude when elaborating requirements: the developer formatted ideas with AI, returned them to teachers for additions, and translated the resulting requests into technical tasks. D4-Feedback generated an initial rubric and example submissions at different quality levels to test the tool before obtaining teachers' student work examples. D5-Practice used AI to generate unit tests, inspect errors, and revise code before submitting changes for review.

Developers also marked where AI did not help or was set aside. D3-DailyActivity valued AI's implementation support but wanted more personal involvement in research ideation: "It was a good starting point, but it definitely was something that I wanted to do a little bit more myself." The same developer attributed the later addition of saved drafts to personally testing the tool and recognizing a classroom inconvenience, not to an AI suggestion. D2-Boards valued AI's help in understanding the codebase and establishing a starting implementation, but described substantial repair work:

"It was really good at, like, setting, like, a foundation in some of the code, but at the end of the day, like, I was always having to debug a lot." (D2-Boards)

D2-Boards added that a code change could appear correct while the application still failed because of database or runtime problems.

Control was exercised through review. D5-Practice read generated code and requested it in manageable portions before allowing generation to continue, a practice that "brings the developer in the loop", and reported catching AI mistakes before pushing changes. D2-Boards attributed additional debugging to peers' changes, some of which they believed were AI-generated and insufficiently reviewed. When asked whether AI belonged in the name of the co-design

approach, D4-Feedback emphasized its central place in the program, D5-Practice its contribution to translating technical mismatches, and D2-Boards its potential to attract participants.

5.2.3 Coordinating participation and working within the limits of delivery. Shared development exposes uneven coordination and support. D2-Boards described the process as effective with a qualification:

“I would say it was effective, except for teacher availability.” (D2-Boards)

D3-DailyActivity identified a specific scheduling mismatch: one teacher attended only in the mornings, while individual development sessions were held in the afternoons, making updates and feedback harder to obtain. D4-Feedback wanted more development time between meetings so that teachers could return to what had changed. D5-Practice reported no conflict with teachers.

Across teams, D3-DailyActivity recalled recognizing overlapping functionality and interface choices during cross-team testing, prompting a shift from thinking about one feature to considering the application as a whole. D1-TeamMode described difficulty incorporating peers’ work when changes were not available on shared branches. D5-Practice objected to changes to implemented code being made without sufficient explanation, clarifying that this concern involved other developers rather than the development leads or program staff; D2-Boards praised collaborators overall. Proposed improvements included earlier shared review, clearer contribution expectations, more predictable schedules, time for developers to explain their work to one another (D3-DailyActivity), and more independent development time (D5-Practice).

Support and resources were experienced unevenly. D1-TeamMode praised staff responsiveness while also describing uncertainty about who was in charge and difficulty obtaining face-to-face help on some days. D3-DailyActivity valued beginning with an existing application, and D4-Feedback found end-of-day recaps useful for obtaining input. D1-TeamMode and D5-Practice both wanted clearer, earlier communication about the schedule. D2-Boards and D5-Practice reported having the resources they needed; D1-TeamMode raised differences in access to AI subscriptions.

Feature progress coexists with unfinished work and limits to testing. Of ideas raised late in the program, D2-Boards explained, “They weren’t abandoned, necessarily, it was that they weren’t prioritized.” D3-DailyActivity had not added scheduled posts, D4-Feedback had not completed PDF export, and D5-Practice had not completed extracting information from images. The export request arrived late, and D5-Practice described development time competing with the paper and poster. D1-TeamMode held back proposed principal and school-district roles because permission and future security arrangements were unclear, and separately questioned the privacy implications of grouping students by support needs, a concern that no teacher feedback resolved. D3-DailyActivity reported refining the warm-up and exit-ticket generator’s prompt only a few times and wanting more opportunities to test its output, and D1-TeamMode questioned whether simulated student interactions could establish how real students would respond.

Positive retrospective appraisals coexist with qualified accounts of change. D2-Boards recalled, “I grew so much as just a developer, and being able to speak to, others and visualize ideas.” D4-Feedback felt teacher feedback improved the work, and D5-Practice believed repeated review saved time.

5.3 RQ2: Teachers’ Envisioned Integration of AI-Supported Tools and Practices

We present the nine School AI Plans through instructional purposes (Section 5.3.1), tool and institutional fit (Section 5.3.2), teachers’ professional learning (Section 5.3.3), and future partnerships (Section 5.3.4). Each plan is attributed to its author using the teacher labels of Table 2, so the tool that teacher’s team built stays visible; the plans record intentions, requests, and some existing practices.

5.3.1 Connecting responsible AI use to subject learning and classroom routines. AI learning is framed around informed, responsible student agency. T2-Boards wanted to move beyond impressions formed through social media, while T3-Boards questioned whether students already knew how to use AI effectively for learning. T6-Feedback proposed teaching generative AI, prompting, and evaluation through problem solving and debugging, with the goal that students would “use AI to support their thinking instead of simply getting answers”. T3-Boards planned for students to use AI to brainstorm algorithms, explain Java concepts, and compare solutions “while still writing and testing their own code”, and T4-DailyActivity planned to use AI coaching during Design Thinking activities while preserving students’ own ideas.

Plans differed in how students would learn about AI and how their participation would be bounded. T2-Boards proposed that students create bots using Gemini or ChatGPT and build simple models with Google Teachable Machine. T8-Practice emphasized prompting, bias, citation, and age-appropriate access for younger students, including protection of their personal information and the content they could access. T2-Boards wanted students to recognize AI’s limitations and the continuing importance of human judgment, while T4-DailyActivity described discouraging copied responses.

AI is fitted to existing curricula through bounded lessons and recurring use. T2-Boards planned approximately eight to ten lessons distributed through the semester, explaining:

“Students will keep coming back to AI as we learn programming, data, and problem-solving.” (T2-Boards)

The same plan placed a formal AI unit after foundational computing content. T3-Boards combined introductory AI literacy with use across the year. T4-DailyActivity planned a five- to six-lesson Design Thinking unit; T9-Practice placed AI literacy at the start of class within a maximum of two 90-minute periods; T6-Feedback and T5-DailyActivity had not settled the timing. T6-Feedback, T8-Practice, and T2-Boards explicitly connected AI content to standards or named objectives; T3-Boards and T7-Feedback supplied course standards or resources without an explicit AI mapping; T9-Practice located the proposed literacy lessons outside the course standards and referred instead to state AI guidance.

5.3.2 Selecting tools whose instructional value could be realized locally. Tool choices balance instructional value with fit and practical usability. T5-DailyActivity valued the tool their team built because “It allows me to use my own curriculum to introduce new materials and assess their understanding.” T4-DailyActivity envisioned warm-ups, discussions, and exit tickets based on uploaded materials, while T8-Practice wanted practice-response data to help decide whether to reteach a concept or move forward. T9-Practice connected Khanmigo to their existing use of Khan Academy, and T6-Feedback identified Code.org as a way to keep AI activities within a platform already used by students. T1-TeamMode described alternatives that could collectively support collaboration, coding, and review, but would require moving between multiple systems; T7-Feedback positioned MerryQuery as one part of a wider set of technologies; T8-Practice named MerryQuery itself when asked about alternatives. T9-Practice wanted the platform’s features debugged and fully developed before classroom integration, and T8-Practice asked when current bugs would be resolved.

Student access is negotiated through uneven institutional readiness. T6-Feedback described checking privacy and technology requirements with administration or IT; T8-Practice identified a district IT request process; and T1-TeamMode named device availability and internet access among possible restrictions. T3-Boards reported that Gemini was the district’s only approved student AI tool, even though other tools were not blocked on student laptops. T9-Practice described expectations for student AI literacy without a developed school- or district-level AI plan. T8-Practice distinguished anticipated teacher-side use from student-side use for which permission was still needed. T1-TeamMode described a curriculum team’s interest in becoming a pilot school but wrote, “I have not moved on to getting permission

to do so.” T7-Feedback described existing student use of AI alongside an uninitiated permission process. Requests for university help included addressing administrators and technology services, facilitating approval, and maintaining access.

5.3.3 Applying and extending teachers’ own AI learning. Teachers adapt AI know-how to their own work while learning remains ongoing. T6-Feedback described a prompt structure with several explicit components:

“I have learned to use clear prompts that give AI a role, task, context, and expected output.” (T6-Feedback)

T4-DailyActivity similarly specified the task, audience, learning objective, and desired output, and described supplying instructional resources to bring responses closer to the curriculum. Uses included lesson plans, rubrics, assessments, differentiated materials, and communications. T3-Boards also planned to use “interactive websites that I design and create using AI”. T2-Boards configured custom assistants for recurring needs: “These have helped me streamline my workflow and reduced the need for me to repeatedly explain my preferences.” T8-Practice’s examples extended beyond instruction to personal and time-management tasks. Checking practices varied: T3-Boards discussed matching AI tools to different purposes, T5-DailyActivity compared outputs across AI systems, and T7-Feedback’s formulation was to “check AI with other AI to make sure that it is correct”. T8-Practice connected learning about material development to the program’s professional development, T2-Boards credited the program alongside an AI fellowship and other training, and T6-Feedback already used AI for planning and wanted to continue learning as the technology changed.

5.3.4 Extending classroom AI through career connections and continued partnership. Career relevance is pursued through subject connections and contact beyond the classroom. T1-TeamMode linked AI-supported programming to software development, and T4-DailyActivity described how AI could be explored across business departments, including marketing, finance, and human resources. T2-Boards wanted students to “hear how AI is being used in real jobs, what skills employers are looking for, and what advice they have for students who are interested in technology careers”. T2-Boards wanted developers to discuss their experience building the app, and T1-TeamMode proposed a visiting developer setting a real-world assignment. Specificity varied: T1-TeamMode requested 20–30 minutes for each of three Python classes, T8-Practice identified a university visit with a proposed time window but no settled date, and T2-Boards imagined a company or research-lab visit with hands-on or shadowing activities. T9-Practice was unsure about career-preparation practices, suggested involving career and technical education colleagues, and prioritized helping teachers understand AI before expanding student use.

Continuing partnership is envisioned as both support and teacher contribution. T6-Feedback requested classroom-ready activities and research-based teaching strategies; T7-Feedback wanted an AI curriculum for grades 9–12; T5-DailyActivity asked whether technical help would be available through a ticketing system or Slack. T9-Practice had found supplied folders insufficient for developing a school protocol and sought help before the start-of-school planning deadline: “I really just need help thinking this through.” T3-Boards and T4-DailyActivity requested further professional development. Teachers also offered ways to remain involved. T3-Boards proposed student surveys and continued development collaboration. T4-DailyActivity linked a continuing consultant role to classroom trials: “I would appreciate opportunities for my students and me to test new AI tools as they are developed.” T5-DailyActivity wanted to introduce district colleagues to the program through a professional learning community meeting. T6-Feedback and T8-Practice tied offers to lead professional development to available time, and T7-Feedback described a possible topic without a firm commitment while requesting no assistance with their own teaching at that point.

6 Discussion

This section summarizes the organizers' reflections (Section 6.1), interprets findings across the three contributions (Section 6.2), and identifies arrangements to retain or revise (Section 6.3). These interpretations reflect our perspective as organizers and researchers; each is accompanied by evidence that qualifies its scope.

6.1 Organizers' Reflections

Teachers' needs became working GenAI tools. The organizers considered the central goal achieved: each team delivered functional, tested software that progressed through most stages of the Software Development Life Cycle (planning, requirements, design, development, and testing) while deployment and maintenance remained future work. The program structure (Section 3) supported the recurring translation of classroom needs into implementation. The two-sided proposal form (Appendix D) documented the classroom focus alongside the technical plan, stable teams sustained teacher-developer collaboration for four weeks, and the weekly exchange (Figure 2) returned working software after each round of teacher input. This resembles the facilitated, team-based process that Penuel et al. describe as co-design [24]. It also involves teachers in the evaluation, refinement, and production phase where Alfredo et al. found passive involvement more common than active involvement [3]. This assessment has limits. Teachers, other teams, and mentors tested the tools, but students did not; as part of an ongoing effort, we attempt to deploy MerryQuery in K–12 classrooms.

Working software shaped participants' experiences and plans. Two outcomes were secondary program objectives. Developers encountered professional practices (onboarding, task documentation, two-reviewer merges, and presentations to industry mentors) and described their development in these terms (Section 5.2.3). Teachers completed the program with software they had shaped and an understanding of its unfinished features, which the organizers associated with greater willingness to continue collaborating. The School AI Plans document this willingness alongside conditions: continued involvement depended on available time, while use depended on permission processes not yet initiated and district policies approving a single student AI tool (Sections 5.3.2 and 5.3.4). These conditions are consistent with Dangel et al.'s account of teachers negotiating GenAI use in relation to district policy and professional norms [8].

Rapid implementation heightened the importance of evaluation. The organizers expected that, without GenAI, teams would have produced wireframes rather than working software. They also recognized that rapid implementation could compromise quality if generated output was accepted without review. The program made GenAI available without prescribing its use (Section 3.1) and distributed evaluation across two-reviewer code review, active teacher testing, cross-team testing, and industry mentor review (Sections 3.3 and 3.4.2). Studies of AI-assisted development similarly identify a continuing need for supervision [4, 17]. Developers' reported limitations concerned testing rather than generation: limited prompt refinement and uncertainty about whether simulated student interactions could predict actual interactions (Section 5.2.3).

Developers delegated different parts of the work to AI. All developers used AI, but differed in the tasks they delegated and the extent of their review. Delegation concentrated on understanding an unfamiliar codebase, drafting plans and test materials, and producing initial implementations. Teachers and developers retained responsibility for deciding what tools should do, while developers integrating changes remained responsible for checking generated output (Section 5.2.2). The two-reviewer rule made review a shared responsibility, although developers reported debugging peers' untested generated code.

6.2 Interpreting Findings Across Contributions

Each contribution provides a descriptive account. Together, they support five interpretations that extend beyond the individual accounts.

Layered mentorship distributed expertise but may also have diffused authority. Five undergraduate developers received four layers of support: two recently graduated development leads responsible for architecture, review, and deployment; a PhD candidate who coordinated the program, contributed code and provided research support and mentorship; four faculty members, at least one present in every working block; and industry mentors who reviewed implementations weekly (Section 3.3). Developer accounts identify several contributions from this support. They described the prepared platform and two-reviewer process, both maintained by the development leads, as enabling conditions. Team Mode’s initial idea came from the graduate lead, while an industry mentor reviewed it using full-class demonstration data (Sections 5.2.1 and 4.1). Developers also valued staff responsiveness and end-of-day recaps (Section 5.2.3). The accounts identify challenges: one developer described difficulty obtaining face-to-face help on some days, and two requested clearer, earlier schedule communication (Section 5.2.3). This level of mentorship limits transferability (Section 7).

Teacher contributions emphasized review points rather than additional automation. Changes following teacher walk-throughs reveal a common pattern. Message selection became an editable summary, immediate posting after generation became a saved draft, generated feedback became a copyable draft rather than a sent message, and automated posting was declined (Section 4.6). In a setting where GenAI made most automation inexpensive to implement, teachers specified where human review should occur: who sees generated content and when. This interpretation emphasizes the placement of review within workflows. It builds on the customization priorities identified by Hou et al. [14] by tracing review points in implemented workflows, and situates teacher–AI complementarity [13] in concrete interface decisions.

Teachers’ needs oriented development toward workflow support, with GenAI in a supporting role. MerryQuery entered the program as a retrieval-grounded chat interface (Section 3.1). Several components developed around it resembled course-management functionality: a landing page for announcements and daily activities, a materials workspace with folders, search, and Google Drive integration, managed access to collaborative boards, help-request routing, and quiz administration with attempt monitoring (Table 1). Generation supported these workflows (drafting warm-ups, feedback, flashcards, and practice questions) rather than defining them. This direction reflected teachers’ stated needs: daily communication and an archive, simpler material management, timely visibility into practice results, and help-seeking during independent work (Sections 4.1–4.5). The plans similarly weighed the effort of switching platforms and positioned MerryQuery as complementary to existing systems (Section 5.3.2). One interpretation is that teachers valued consolidated classroom workflows, with GenAI as one capability among several. This would position learning-management platforms with generative features as relevant comparisons alongside chatbot-centered systems examined in recent work [14, 33]. Alternatively, this direction may reflect the platform’s starting point: question answering was already available, so teams developed missing capabilities around it.

Rapid implementation shifted constraints toward coordination and verification. Developers rarely identified implementation as their principal difficulty. They described limited teacher availability, afternoon sessions teachers could not attend, changes unavailable on shared branches, unexplained code changes by peers, and generated code pushed without local testing (Section 5.2.3). When features could be drafted within days, teacher feedback and code review became constrained by participants’ available time. Some challenges are common to student teams developing software with

external stakeholders [29]. GenAI may increase review demands, consistent with the integration challenges reported by Li et al. [17]. On this interpretation, the two-reviewer rule protected collaborators from inheriting unverified work rather than simply adding development overhead.

Initial classroom use may favor teacher-facing features, with institutional access requiring continued partnership. The plans suggest two adoption paths. Features operated by teachers alone (organizing course materials, generating warm-ups and flashcards, and drafting lesson plans) require no student accounts; teachers already described using other AI for planning (Section 5.3.3). Student-facing features (Team Mode, Collaborative Boards, and Practice Sets) depend on permissions most teachers were willing to request, a district’s approval of a single general-purpose tool, and device and network conditions (Section 5.3.2). Operator role alone does not determine these requirements. When tools process student work, as in the Feedback Generator and Team Mode’s monitoring, the privacy and technology review that teachers described discussing with administration or IT is likely to apply regardless of who operates them. Only uses without student data avoid this review (Sections 5.3.2 and 4.4). Teacher-operated features involving no student data are therefore likely candidates for initial classroom use. These plans suggest institutional barriers to classroom use, complementing Alfredo et al.’s concerns about limited deployment research [3].

6.3 Arrangements to Retain or Revise

The findings suggest six arrangements for similar collaborations. Retain review at each hand-off and specify review points as design decisions: proposal forms should identify where generated content receives human review before reaching students, reflecting teachers’ most consistent design contribution. Also specify how generated output will be tested with teachers (which materials, how many outputs, and when) given developers’ limited opportunities for prompt refinement (Section 5.2.3). Prioritize teacher participation and review capacity over implementation time: align teacher testing windows with developer sessions and scale code review to the rate of generation. Assign clear responsibility for daily priorities and scheduling within the mentoring structure to prevent distributed expertise from obscuring authority. Begin institutional approval processes in the first co-design week, classify features by whether they process student data, and plan initial classroom use around those that do not. Finally, retain perspectives close to students’ experiences until students can participate directly. Undergraduate developers identified a disagreement about digital versus in-person help (Sections 5.2.1 and 5.2.3).

7 Limitations and Future Work

This study has several limitations. First, ExLAIM involved one cohort of nine teachers and five undergraduate developers, one university program, and one platform. Its preparation phase, established codebase, and substantial mentorship bound transferability to settings with different resources. Second, testing involved participating teachers, cross-team reviewers, and industry mentors, but did not include actual students or systematic classroom evaluation. Future work should conduct longitudinal classroom pilots, and follow how technical support, maintenance, and design priorities evolve during use. Third, developers’ accounts of their contributions and their own and peers’ review practices were self-reported, while School AI Plans recorded intentions and reported practices rather than subsequent behavior or changes attributable to ExLAIM. Researchers’ involvement in the program may also have shaped responses and interpretations. Finally, the interview and plan analyses were LLM-led: researcher-authored exemplars influenced segmentation and coding, and LLM agreement may reflect shared interpretive tendencies. Programmatic quotation checks and the two human audits described in Section 5 reduce these risks, but errors outside the audited samples may

remain. As future work, we plan to rerun ExLAIM while incorporating experiences reflected in this paper, including additional time for development and testing, clearer coordination, and stronger opportunities for classroom evaluation with students. We will also examine how technical support, maintenance, institutional conditions, and design priorities evolve as the tools move from co-design into classroom use.

8 Conclusion

ExLAIM paired nine K–12 teachers with five undergraduate developers through two weeks of developer preparation and four weeks of co-design in MerryQuery. Developers described GenAI assistance alongside continuing human responsibility for interpretation, testing, and integration. The program carried classroom concerns into working software while making conditions for school use more specific. Teachers connected intended uses to existing curricula and identified needs for development, institutional permission, and continuing partnership mediated through the working software they helped develop. ExLAIM offers a situated model for HCI researchers and also institutions preparing future developers: teachers shape and evaluate working versions, developers respond with technical judgment, and planned partnerships work toward classroom evaluation and revision.

Ethics and Privacy Statement

This study was reviewed by the authors' institutional review board (IRB) and determined to be exempt.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. DUE-2526340. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

- [1] Agile Business Consortium. 2014. MoSCoW Prioritisation. The DSDM Agile Project Framework Handbook. Accessed 11 September 2026. <https://www.agilebusiness.org/resource/the-dsdm-agile-project-framework/>
- [2] Nil-Jana Akpinar, Sandeep Avula, Chia-Jung Lee, Brandon Dang, Kaza Razat, and Vanessa Murdock. 2026. LLM or Human? Perceptions of Trust and Quality in Research Summaries. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. ACM, Barcelona Spain, 1–25. doi:10.1145/3772318.3793386
- [3] Riordan Alfredo, Vanessa Echeverria, Yueqiao Jin, Lixiang Yan, Zachari Swiecki, Dragan Gašević, and Roberto Martinez-Maldonado. 2024. Human-centred learning analytics and AI in education: A systematic literature review. *Computers and Education: Artificial Intelligence* 6 (June 2024), 100215. doi:10.1016/j.caeai.2024.100215
- [4] Valerie Chen, Ameet Talwalkar, Robert Brennan, and Graham Neubig. 2026. Code with Me or for Me? How Increasing AI Automation Transforms Developer Workflows. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3772318.3790850
- [5] Michelene T. H. Chi and Ruth Wylie. 2014. The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes. *Educational Psychologist* 49, 4 (2014), 219–243. doi:10.1080/00461520.2014.965823
- [6] Rebecca Cober, Esther Tan, Jim Slotta, Hyo-Jeong So, and Karen D. Könings. 2015. Teachers as participatory designers: two case studies with technology-enhanced learning environments. *Instructional Science* 43, 2 (March 2015), 203–228. doi:10.1007/s11251-014-9339-0
- [7] Shih-Chieh Dai, Aiping Xiong, and Lun-Wei Ku. 2023. LLM-in-the-loop: Leveraging Large Language Model for Thematic Analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, Singapore, 9993–10001. doi:10.18653/v1/2023.findings-emnlp.669
- [8] Aayushi Dangol, Smriti Kotiyal, Robert Wolfe, Alex J. Bowers, Antonio Vigil, Jason Yip, Julie A. Kientz, Suleman Shahid, Tom Yeh, Vincent Cho, and Katie Davis. 2026. Relief or Displacement? How Teachers Are Negotiating Generative AI's Role in Their Professional Practice. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3772318.3791904

- [9] Allison Druin. 1999. Cooperative Inquiry: Developing New Technologies for Children with Children. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 592–599. doi:10.1145/302979.303166
- [10] Douglas Fuchs, Lynn S. Fuchs, Patricia G. Mathes, and Deborah C. Simmons. 1997. Peer-Assisted Learning Strategies: Making Classrooms More Responsive to Diversity. *American Educational Research Journal* 34, 1 (1997), 174–206. doi:10.3102/00028312034001174
- [11] Egon G. Guba. 1981. ERIC/ECTJ Annual Review Paper: Criteria for Assessing the Trustworthiness of Naturalistic Inquiries. *Educational Communication and Technology Journal* 29, 2 (1981), 75–91. doi:10.1007/BF02766777
- [12] Kenneth Holstein, Gena Hong, Mera Tegene, Bruce M. McLaren, and Vincent Aleven. 2018. The Classroom as a Dashboard: Co-Designing Wearable Cognitive Augmentation for K–12 Teachers. In *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*. Association for Computing Machinery, New York, NY, USA, 79–88. doi:10.1145/3170358.3170377
- [13] Kenneth Holstein, Bruce M. McLaren, and Vincent Aleven. 2019. Co-Designing a Real-Time Classroom Orchestration Tool to Support Teacher–AI Complementarity. *Journal of Learning Analytics* 6, 2 (2019), 27–52. doi:10.18608/jla.2019.62.3
- [14] Irene Hou, Zeyu Xiong, Philip J. Guo, and April Yi Wang. 2026. “Bespoke Bots”: Diverse Instructor Needs for Customizing Generative AI Classroom Chatbots. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3772318.3791491
- [15] Nicole M. Hutchins, Shan Zhang, Joanne R. Barrett, and Maya Israel. 2025. Empowering Educators in AI: Insights from Co-Designing an AI Microcredential with and for K-12 Educators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 29137–29144. doi:10.1609/aaai.v39i28.35186
- [16] Sam Lau, Kianoosh Boroojeni, Harry Keeling, and Jenn Marroquin. 2026. Barriers that Programming Instructors Face While Performing Emergency Pedagogical Design to Shape Student-AI Interactions with Generative AI Tools. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3772318.3790682
- [17] Jie Li, Youyang Hou, Laura Lin, Ruihao Zhu, Hancheng Cao, and Abdallah El Ali. 2026. Vibe Coding in Product Teams: Reconfiguring AI-Assisted Workflows, Prototyping, and Collaboration. In *Proceedings of the 5th Annual Symposium on Human-Computer Interaction for Work*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3808045.3808062
- [18] Yvonna S. Lincoln and Egon G. Guba. 1986. But Is It Rigorous? Trustworthiness and Authenticity in Naturalistic Evaluation. In *Naturalistic Evaluation*, David D. Williams (Ed.). Number 30 in New Directions for Program Evaluation. Jossey-Bass, San Francisco, CA, 73–84. doi:10.1002/ev.1427
- [19] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW, Article 72 (2019), 23 pages. doi:10.1145/3359174
- [20] Sara S. McMillan, Michelle King, and Mary P. Tully. 2016. How to use the nominal group and Delphi techniques. *International Journal of Clinical Pharmacy* 38 (2016), 655–662. doi:10.1007/s11096-016-0257-x
- [21] Line Have Musaeus, Marianne Graves Petersen, and Clemens Nylandsted Klokmose. 2024. Bringing Teachers and Researchers Together through Participatory Design and Cooperative Prototyping in Computing Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. Association for Computing Machinery, New York, NY, USA, 902–908. doi:10.1145/3626252.3630796
- [22] Rebecca Nicholson, Tom Bartindale, Ahmed Kharrufa, David Kirk, and Caroline Walker-Gleaves. 2022. Participatory Design Goes to School: Co-Teaching as a Form of Co-Design for Educational Technology. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3491102.3517667
- [23] Santiago Ojeda-Ramirez, Eva Durall Gazulla, and Kylie Peppler. 2026. Emergent Technology, Emergent Critique: Students and Teachers Developing Critical AI Literacy Through Participatory Design Around Generative AI. In *Proceedings of the 25th Annual ACM Interaction Design and Children Conference*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3773077.3812170
- [24] William R. Penuel, Jeremy Roschelle, and Nicole Shechtman. 2007. Designing Formative Assessment Software with Teachers: An Analysis of the Co-Design Process. *Research and Practice in Technology Enhanced Learning* 2, 1 (2007), 51–74. doi:10.1142/S1793206807000300
- [25] Prerna Ravi, John Masla, Gisella Kakoti, Grace C. Lin, Emma Anderson, Matt Taylor, Anastasia K. Ostrowski, Cynthia Breazeal, Eric Klopfer, and Hal Abelson. 2025. Co-designing Large Language Model Tools for Project-Based Learning with K12 Educators. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–25. doi:10.1145/3706598.3713971
- [26] Johnny Saldaña. 2021. *The Coding Manual for Qualitative Researchers* (4th ed.). SAGE, London, UK.
- [27] Ismaila Temitayo Sanusi, Deepti Tagare, and Fred Martin. 2026. A Research Course to Develop AI Tools for K–12 Learning. In *Proceedings of the 57th ACM Technical Symposium on Computer Science Education V. 1*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3770762.3772631
- [28] Douglas Schuler and Aki Namioka (Eds.). 1993. *Participatory Design: Principles and Practices*. Lawrence Erlbaum Associates, Hillsdale, NJ, USA.
- [29] Jan-Philipp Steghöfer, Håkan Burden, Regina Hebig, Gül Çalıkli, Robert Feldt, Imed Hammouda, Jennifer Horkoff, Eric Knauss, and Grischa Liebel. 2018. Involving External Stakeholders in Project Courses. *ACM Transactions on Computing Education* 18, 2 (2018), 8:1–8:32. doi:10.1145/3152098
- [30] Mei Tan, Hansol Lee, Dakuo Wang, and Hari Subramonyam. 2024. Is a Seat at the Table Enough? Engaging Teachers and Students in Dataset Specification for ML in Education. *Proceedings of the ACM on Human-Computer Interaction* 8, CSCW1 (April 2024), 1–32. doi:10.1145/3637358
- [31] Saara Tenhunen, Tomi Männistö, Matti Luukkainen, and Petri Ihantola. 2023. A Systematic Literature Review of Capstone Courses in Software Engineering. *Information and Software Technology* 159 (2023), 107191. doi:10.1016/j.infsof.2023.107191
- [32] Lev S. Vygotsky. 1978. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, MA.

- [33] Minju Yoo, Hyounghook Jin, and Juho Kim. 2025. How Do Teachers Create Pedagogical Chatbots?: Current Practices and Challenges. Version Number: 2. doi:10.48550/ARXIV.2503.00967
- [34] Dora Zhao, Hannah Cha, Michael J Ryan, Angelina Wang, Rachel Baker-Ramos, Evyn-Bree Helekahi-Kaiwi, Rebecca Diego, Josiah Hester, and Diyi Yang. 2026. Whose Knowledge Counts? Co-Designing Community-Centered AI Auditing Tools with Educators in Hawai‘i. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3772318.3790958

A Participant Backgrounds

Table 2 summarizes the backgrounds of the five undergraduate developers (panel a) and the nine K–12 teachers (panel b), ordered within each panel by the team in which the participant worked.

Table 2. Participant backgrounds. Developers and teachers are reported in separate panels because the attributes recorded for the two roles differ. Each team paired one undergraduate developer with one or two K–12 teachers; each label’s suffix names the tool that team built. Generative AI use was recorded for developers only, and years of teaching for teachers only.

(a) Undergraduate developers						
Developer	Team	Standing	Age	Major	Prior K–12 experience	Generative AI use
D1-TeamMode	1	Senior	25–34	Computer Programming	None	Daily; ChatGPT, Claude, Copilot
D2-Boards	2	Junior	18–24	Computer Science	None	Weekly; ChatGPT, Claude
D3-DailyActivity	3	Junior	18–24	Computer Science	None	Weekly; ChatGPT, Claude, Gemini
D4-Feedback	4	Senior	18–24	Computer Science	K–12 teaching and study	Weekly; ChatGPT, Claude, Gemini, Copilot
D5-Practice	5	Sophomore	18–24	Computer Science	K–12 teaching	Daily; Claude
(b) K–12 teachers						
Teacher	Team	Grades taught	Years teaching	Courses taught		
T1-TeamMode	1	9–12	5	Python I and II, TED-technology engineering design, Intro to CS		
T2-Boards	2	9–12	21	Adobe Visual Design, Intro Computer Science		
T3-Boards	2	9–12	11	3D Modeling, Game Art and Design, Advanced Game Art and Design		
T4-DailyActivity	3	9–12	16	Virtual Enterprise Honors, Adobe Visual Design, Entrepreneurship 1		
T5-DailyActivity	3	9–12	7	AP Computer Science Principles		
T6-Feedback	4	9–12	21	Honors Microsoft Excel, Computer Science I and II		
T7-Feedback	4	9–12	11	Microsoft Word and PowerPoint, Microsoft Excel, Computer Science I Honors		
T8-Practice	5	6–8	22	Career Exploration, Digital Literacy, Computer Science Discoveries I, Artificial Intelligence		
T9-Practice	5	9–12	Not reported	IB Math Applications and Interpretations, Honors Math 3		

B Materials used in ExLAIM

Table 3 lists the study materials, and what each contains. All were produced during the program except the interviews, conducted two weeks afterward.

Table 3. Program materials and their contents.

Material	Contents
Program materials	Weekly schedules, activity handouts, presentation slides
Feature proposals, five teams	Two-sided form. Side A (teacher): classroom design focus, evidence grounding, options considered. Side B (developer): technical approach, feasibility, build plan, test plan. Two template versions were used; Appendix D reproduces version 2.0
Team project papers, five	Developer-authored account of the tool: what was built, what changed, what was left out, and a reflection on the work
Teacher-facing user guides, one per tool	Documentation of the completed tool, with screenshots
Cross-team testing worksheets	Structured notes from members of other teams testing each tool during the final two weeks
Developer interviews, five	Semi-structured, one per developer, 22–57 minutes ($M = 40$); origin of the idea, how the tool evolved, working with teachers, use of GenAI tooling, reflection
School AI Plans, nine teachers	Structured form given in week 3 and completed by the end of the program: planned AI content and timing, standards alignment, tools students will use, fallback if MerryQuery is unavailable, IT permission status, support requested

C Team Mode: Grouping and Teacher Controls

Instructional groups and strategies. Team Mode supported private instructional groups based on students’ learning needs, as known to teachers. Teachers could associate groups with strategies such as chunked directions, visual examples, additional processing time, understanding checks, small teacher-led groups, and extension activities. They could keep instructional groups together, mix students across them, or adjust the live groups manually. Group information was restricted to authorized staff, and the interface avoided displaying diagnoses, individualized education program (IEP) or 504-plan labels, or medical information.

Teacher oversight and support. Teachers could view pending and ongoing help requests, configure when peer support was available, and create groups for regular collaboration, structured support, and extension activities. Within collaborative spaces, teachers could monitor conversations, intervene, and determine how much assistance an AI facilitator could provide. Students could also submit check-ins, while teachers could review the class atmosphere and common questions to identify individual or group support needs. Peer requests and peer rooms remained identifiable to authorized staff despite being anonymous from students’ perspective.

D Feature Proposal Form

Teams documented proposed features using the two-sided form below (version 2.0). The form was a companion to the AI-Integrated Daily Planning Template. Its introduction explained that a MerryQuery feature did not have to target one class or audience, so Side A let the team define the feature on its own terms before tying it to a specific lesson. Side B was completed by the team’s assigned MerryQuery developer and returned to the team, so that the “MerryQuery Feature/Configuration Used” and “RAG Source Materials” fields in the Daily Planning Template reflected what was actually being built. The header recorded the team number, teachers, CS undergraduate, graduate mentor, industry mentor, and assigned developer. The form’s Week 1–4 labels refer to the four teacher–developer weeks, corresponding to ExLAIM active weeks 3–6.

Side A: Team (completed in Week 2). A1, teacher fields. Classroom- and curriculum-specific fields completed by the teachers.

- (1) **Classroom Design Focus Connection.** Paste your Week 1 Classroom Design Focus statement: “In my [class/unit], I want MerryQuery to [feature], so that [outcome].”
- (2) **First Lesson to Pilot This Feature.** Instructional lesson number and topic from your Daily Planning Template.
- (3) **RAG Source Materials.** What teacher-uploaded materials should ground this feature’s responses (textbook chapters, standards, past assignments, rubrics)?
- (4) **Desired Teacher/Student Experience.** What should a teacher or student see, or be able to do, when this feature works?
- (5) **Evidence Grounding.** What research, evidence-based practice, or expert source supports this feature? Select at least one item from the Week 3 Evidence Toolkit, or another peer-reviewed or practitioner source, and explain in one sentence how the evidence supports this design choice.

A2, developer fields. Design-framing fields completed together, informed by A1.

- (6) **Feature Purpose (General).** This feature does not need to target one class or audience. What broader need across the teachers’ work does it address?
- (7) **Idea Options Considered.** Per your Week 2 three-options activity: what other MerryQuery features or configurations did the team generate before narrowing to this one? Note the alternatives and why this one won out on buildability and teacher fit (Design Thinking Playbook, Ch. 1.7–1.8).
- (8) **Feature Summary/User Story.** As a _____, I want _____, so that _____.
- (9) **Week 4 Success Criteria.** How will you know it worked when you test and implement it? What goes on your presentation/poster?

Side B: Developer (completed before the Week 3 build began).

- (1) **Feasibility.** Build as specified, build with modification, or needs discussion with the team; explain.
- (2) **Technical Approach.** Closest existing MerryQuery capability, or what needs to be newly built. How does it plug into the RAG pipeline?
- (3) **Inputs and Outputs/Data Flow.**
- (4) **Constraints and Dependencies.**
- (5) **Week 3 Build Plan.** Milestones and target completion date, so the team can plan Week 4 testing.
- (6) **Code and Workflow.** GitHub repository link, with a README covering setup, how to run it, and any configuration needed. Note how you are tracking build progress (GitHub Issues, Trello, Notion, etc.) so the team can follow along between check-ins.
- (7) **How to Test This (Week 4).** How will the team access and try the feature for Week 4 testing? Include a test/staging link or access steps, plus two or three sample scenarios to try.
- (8) **Questions Back to the Team.**
- (9) **Sign-off.** Developer lead and date returned to the team.

E School AI Plan Form

Teachers received the following form in week 3 and could complete it until the end of the program. It asked teachers to describe their plan to introduce AI concepts, careers, tools, and speakers, and how NC State could help with this plan

Manuscript submitted to ACM

during the school year, and it invited them to use voice typing or other AI tools to help write their plans. The form began with the teacher's name, email, school, class titles and grade levels, and number of students.

AI Concepts for Students.

- Q1.1** What AI concept or topic do you want students to learn? Please reference the Content Learning Objective and AI Literacy Objective you are planning to teach based on your lessons here.
- Q1.2** How many lessons do you expect to develop? Approximately how much time will these lessons take?
- Q1.3** What standards will this align with? List the standard codes and paste a link to the standards document (state, district, or school).
- Q1.4** When will you teach this? Check all that apply: August through May, or not sure yet.
- Q1.5** Upload your lesson plan to the shared drive with your name in the file name, then paste the link below.
- Q1.6** Is this lesson plan the same as your Daily Planning Template (Part 1 of the Team Workbook), or a separate document?

AI Tools.

- Q2.1** What tools will students use? For each tool, please answer why they will use that tool and what the tool will be used for. Teachers could answer in prose or in a table listing each tool, what students will do with it, and why this tool.
- Q2.2** What existing or accessible tools could you use if MerryQuery is unavailable or not feasible in your school context? Please link the daily planning document from your team proposal as a reference.
- Q2.3** Please specify the process you need to follow to implement AI tool usage for your students, or any restrictions that prevent it.
- Q2.4** Where in the process of IT/school permissions are you? Options: haven't started; know who to ask, haven't asked yet; request submitted, waiting; approved; blocked or denied; no approval needed; other.

AI in My Teaching.

- Q3.1** What kinds of prompts and techniques have you learned to use AI to assist in your work as a teacher?

AI Careers.

- Q3.2** What practices do you envision yourself beginning that could help students prepare for careers in AI?
- Q4.1** Which of the following do you plan to do this year? Check all that apply: invite guest speakers; plan field trips; implement AI in the News activities; career exploration activity or project; none of the above; other.
- Q4.2** If you want our help with arranging speakers, please specify how long you would like a speaker to talk, the desired speaker (if one of our summer speakers or project team) or industry/field, and any specific questions or topics they should address.
- Q4.3** If you checked field trips, where would the field trip be, or what kind of site? Approximately how long would it last?

Support from the University.

- Q5.1** How would you like the NC State team to help support student learning about AI concepts?
- Q5.2** How would you like the NC State team to help support AI/technology use for your classroom(s)?
- Q5.3** How would you like the NC State team to help support you in leveraging AI for your teaching?

Q5.4 How would you like the NC State team to help support student learning about AI careers?

Q5.5 We were considering having monthly teacher-led meetings this school year. Would you be interested in leading a session? If so, what would be a topic you could lead?

Final Thoughts.

Q6.1 Is there anything else you wish to plan, share, or request?

F Developer Interview Protocol

The semi-structured protocol was planned for about 30 minutes, with probes used as needed. Section timings below are the planned allocations.

Section 1. System and Feature Walkthrough (8–10 minutes). A tool-oriented walkthrough in which developers used screenshots or the live system.

- (1) Could you walk us through the most important tool that you made for MerryQuery? Using a demonstration with the tool, please answer the following:
 - What are the core features of your tool?
 - What does your tool do?
 - What problem was it designed to solve?
 - Which features of your tool do you think are the most important?
 - Were there features added later in development for this tool?

Section 2. Feature Development Process (12–15 minutes). Asked for each major feature, or the most important ones.

- (2) Where did the idea for this tool come from? How did the literature review on educational theories backing your tool go? How about the comparison with other tools? *Probes:* a teacher; the research team; you as the developer; other software platforms or literature review; AI-generated suggestions; multiple sources.
- (3) How did the tool evolve throughout development? *Probes:* What was the original idea? How did it change? What feedback influenced those changes? What technical or pedagogical tradeoffs had to be made?
- (4) What role did AI play in developing this tool? *Probes:* brainstorming; generating examples; writing prompts; prototyping; debugging; other.

Section 3. Looking Across Tools (7 minutes).

- (5) Were there ideas or features that didn't make it into the final system? *Probes:* Why were they abandoned? Were there ideas that changed substantially before implementation?

Section 4. Reflection (3–5 minutes).

- (6) Looking back, was this co-design process effective? Was the schedule good enough? Were you provided with sufficient equipment?
- (7) What were the biggest challenges in collaborating with teachers and researchers while developing the tool?
- (8) If another team wanted to use this teacher-developer-AI co-design approach, what advice would you give them?

G Additional Details of the LLM-Led Analysis

Data preparation. Interview transcripts were de-identified before analysis. Residual identifiers in the plans—the partner university, two districts, a city, three first names, two dates, and a document URL—were replaced with stable tokens before coding. A subsequent scan of both corpora found no remaining names, schools, districts, or cities. Four plan cells misaligned during export were realigned and logged; 12 empty cells were excluded, leaving 172 responses.

Extraction provenance and uncoded turns. Extracted interview turns were annotated as directly prompted, asked out of sequence, raised by the developer, incidental to another question, responses to clarification, or misdirected (answering about a different referent than asked). These annotations preserved how statements arose for consideration during coding. Eleven turns received no code because they contained only a bare confirmation, a clarification request, or an unrecoverable fragment.

Exemplar construction. A researcher authored eight exemplars per corpus before the analysis runs. All exemplar responses were corpus cells, and every quoted span was programmatically verified against its source. Interview exemplars demonstrated every coding-schema field, In Vivo and process coding, and a response warranting no code. Plan exemplars demonstrated the quotation–definition–code format across polished prose, voice-typed run-ons, and fragments, but included no tag fields, In Vivo code, or zero-code response and omitted the checkbox register. The exemplars demonstrated format, unitizing, and granularity rather than a closed code list.

First-cycle coding and refinement procedure. Familiarization memos were produced without questions, research questions, or exemplars. Exemplars illustrated format and granularity without restricting available codes. Plans used provisional, structural, descriptive, and In Vivo coding; interviews used descriptive, process, and In Vivo coding. After the first round of coding, programmatic checks tested whether citations used during consolidation were accurate and matched corpus cells verbatim. During refinement, the two LLM coders retained their own contexts and exchanged round files in writing: Coder A proposed codes and definition changes, and Coder B accepted, rejected, or partially accepted each with a rationale and could propose revisions. A round comprised one exchange between coders; convergence required a full round without proposals from either coder. Disagreements the two coders did not resolve were referred to a fresh-context LLM adjudicator, and refinement was capped at six rounds.

Intermediate coding outputs. First-cycle coding produced 383 applications across 255 plan codes and 494 applications across 337 interview codes. Most codes were new rather than exemplar-derived: 231 for plans and 317 for interviews. Consolidation yielded 78 plan codes and 57 interview codes, each provisionally grouped into 11 categories, before refinement, which left 73 plan codes and 56 interview codes in the final codebooks (Section 5.1).

Reliability details. The two verification coders, unaware of each other, used the same LLM and reasoning-effort setting as first-cycle coding and received only the goal, codebook, exemplars, and corpus. Prevalence and bootstrap intervals were clustered by plan response or interview turn. For plans, code-level $\kappa = 0.965$ (95% CI [0.948, 0.980]); all 882 programmatically verified quoted spans matched the source. For interviews, $\kappa = 0.981$ (95% CI [0.969, 0.992]) across 154 turn–question rows and 8,624 cells; all 2,204 quotation and provenance checks passed. The human audit of first-cycle output examined unit–code–definition–quotation–inclusion–criteria rows.

Reliability computation. Agreement was scored per unit $\times$ code cell, each cell recording whether a coder applied that code to that unit: 12,556 cells for plans and 8,624 for interviews. Because each unit carried only a few of the 73 or 56

codes, the cell matrix is sparse, and κ is prevalence-sensitive. Where either coder applied a code, both did in 93% of plan cells (340 of 364) and 97% of interview cells (359 of 372).

Codebook-integrity checks. A mechanical gate checked for broken, relative, or one-sided cross-references and references to files unavailable to coders. It was introduced during adjudication in the plan analysis. For interviews, it was expanded to the nine defect classes cataloged during that analysis and run after consolidation and every revision. Both gates were validated against known-bad codebooks. These checks assessed codebook integrity; the main-text reliability assessment and human audits addressed coding consistency and conceptual grounding.

H Full Thematic Hierarchies

Tables 4–7 report the complete three-level hierarchies produced by the second-cycle pattern coding described in Section 5.1 and Appendix G. Each first-cycle code was assigned one primary category and one theme; categories are nested within themes, and every applied code appears exactly once. Code labels are reproduced as applied, with the platform name normalized to MerryQuery and spelling standardized to U.S. English. Illustrative excerpts are omitted here; representative quotations appear in Sections 5.2 and 5.3.

H.1 Teachers’ School AI Plans

Table 4. Thematic hierarchy for the nine School AI Plans (RQ2): seven themes, 22 categories, and 73 codes. Themes are given with their second-cycle pattern code; codes are listed in full under the category to which each was primarily assigned. Part 1 of 2 (themes S1–S4).

Category	Codes
S1 CULTIVATING RESPONSIBLE STUDENT AGENCY	AI learning is framed around informed, responsible student agency. (3 categories, 9 codes)
S1.1 Developing AI understanding and participation	Students' current AI use as the starting problem; Intended AI literacy content; Students building or configuring AI artifacts
S1.2 Exercising critical and ethical judgment	Critical evaluation of AI output; Ethics, responsibility and legality as content
S1.3 Preserving student thinking, authorship, and appropriate access	AI as thinking support, not answer source; Student originality and academic integrity alongside AI use; Bounding students to teacher-approved sources; Protective stance toward younger students' AI access
S2 FITTING AI INTO THE CURRICULUM	AI is fitted to existing curricula through bounded lessons and recurring use. (4 categories, 13 codes)
S2.1 Anchoring AI in disciplinary learning	Subject content as the lesson's object; AI supporting subject-area tasks
S2.2 Negotiating curricular alignment	AI content mapped to named standards; Course standards supplied without AI mapping; AI content outside existing course standards
S2.3 Allocating and sequencing instructional time	Scope set by existing course structure; Planned lesson count and instructional time; Calendar placement of the AI teaching; AI teaching front-loaded at the start; AI teaching sequenced after foundational content; Timing not yet decided
S2.4 Making AI recur across classroom practice	AI integrated across the course, not a standalone unit; Tool as a routine daily fixture of the class
S3 ASSEMBLING A WORKABLE TOOL REPERTOIRE	Tool choices balance instructional value with fit and practical usability. (3 categories, 10 codes)
S3.1 Specifying the platform repertoire and alternatives	Named MerryQuery features planned; Named non-MerryQuery tool planned; Approved general-purpose AI for hints and feedback; Fallback tools named; MerryQuery as complement, not replacement
S3.2 Seeking continuity and curriculum fit	Warrant: continuity with what is already in place; Tool grounded in the teacher's own materials
S3.3 Weighing instructional benefit and usability	Warrant: effect on learning or teaching; Consolidation versus platform-switching friction; Adoption conditional on the prototype maturing
S4 NEGOTIATING INSTITUTIONAL ACCESS	Student access is negotiated through uneven institutional readiness. (3 categories, 10 codes)
S4.1 Encountering institutional rules and material conditions	School or district approval required; District approval constrains tool choice; Privacy, regulatory and infrastructure constraints; Absence of institutional AI policy
S4.2 Navigating different and sometimes ambiguous access states	Permission process not yet initiated; Permission request submitted and pending; Student access unresolved or deferred; Informal institutional interest in piloting; No institutional barrier reported at the classroom level
S4.3 Seeking mediation of institutional access	University team asked to broker institutional access

Table 5. Thematic hierarchy for the nine School AI Plans (RQ2), continued. Part 2 of 2 (themes S5–S7). See Table 4 for the full legend.

Category	Codes
S5 ADAPTING AI KNOW-HOW TO TEACHER WORK Teachers adapt AI know-how to their own work while learning remains ongoing. (3 categories, 8 codes)	
S5.1 Developing a practical AI repertoire	Prompting technique learned; Verification and tool-matching technique learned
S5.2 Applying AI to instructional production and personal workflow	AI for teacher planning and materials; Building custom AI assistants for teacher work; The teacher's non-instructional AI use
S5.3 Locating and continuing professional learning	Learning credited to the program alone; Learning credited to the program alongside other training; Intent to keep learning as AI changes
S6 CONNECTING CLASSROOM AI TO FUTURE PATHWAYS Career relevance is pursued through subject connections and contact beyond the classroom. (3 categories, 10 codes)	
S6.1 Making future pathways and workplace relevance explicit	AI skills linked to future careers; Careers framed through the teacher's subject area; Speaker topics: AI at work and employability; No career-preparation practice envisioned yet
S6.2 Arranging contact with professions and further study	Career and outreach activities selected; Speakers and professional contacts requested; University campus visit planned; Industry site visit planned
S6.3 Turning outreach intentions into workable activities	Speaker format specified; Field trip logistics and open questions
S7 SUSTAINING RECIPROCAL PROFESSIONAL PARTNERSHIP Continuing partnership is envisioned as both support and teacher contribution. (3 categories, 13 codes)	
S7.1 Articulating differentiated support needs and limits	Lesson and curriculum development support requested; Implementation and technical support requested; Broader AI support beyond the platform wanted; Materials already supplied by the program; Support needed before a school-calendar deadline; Nothing requested of the university team
S7.2 Maintaining professional relationships and shared learning	Ongoing PD and teacher networking requested; Program people requested as speakers or visitors; Partnership continuing beyond the program
S7.3 Contributing through leadership, dissemination, and co-development	Offering to lead teacher PD; School-level AI policy leadership; Continued role in tool development offered; Spreading the program to colleagues

H.2 Developer Interviews

Table 6. Thematic hierarchy for the five developer interviews (RQ1): six themes, 23 categories, and 56 codes. Theme statements are the second-cycle pattern statements; codes are listed in full under the category to which each was primarily assigned. Codes shown in quotation marks are In Vivo codes taken from a developer’s own wording. Part 1 of 2 (themes I1–I3).

Category	Codes
I1 Design direction draws unevenly on research, comparison and developer judgment. <i>(3 categories, 7 codes)</i>	
I1.1 Engagement with literature	Literature review limited or skipped; Literature review carried out
I1.2 Comparison, differentiation and their limits	Comparison against existing tools; Gap analysis and differentiation from existing tools; Limits of tool comparison; “why should someone use it”
I1.3 Developer purpose and scope	Advice on the developer’s own approach
I2 Teacher expertise guides design through translation, negotiation and repeated review. <i>(5 categories, 15 codes)</i>	
I2.1 Classroom needs and prepared ideas	Feature idea originated with teachers; Teachers’ preparedness and specificity
I2.2 Translation and negotiation of requirements	Eliciting and translating what teachers wanted; Divergent priorities between co-teachers; Gap between what teachers wanted and what could be built
I2.3 Teacher authority over design	Designing for the teacher as the user; Professional practice invoked as the standard; “I’m ChatGPT itself”
I2.4 Participation and iterative acceptance	Extent of teacher involvement across the project; Build–show–revise loop with teachers; Teachers’ materials and judgment as the acceptance test
I2.5 Rapport, resistance and demanding feedback	Rapport and mutual understanding with teachers; Teacher resistance to the tool; “Mindsets”; Teacher feedback hard to absorb
I3 AI assists broadly while developers negotiate its usefulness, credit and control. <i>(4 categories, 9 codes)</i>	
I3.1 Assistance across development tasks	AI used to produce non-code material; AI used to build and debug
I3.2 Uneven usefulness and human-originated decisions	Assessment of what AI did well and badly; AI ruled out for part of the work
I3.3 Review and control of generated code	Controlling AI output before use; “brings the developer in the loop”; “if you’re not in control of AI, AI is going to be in control of you”
I3.4 Recognition and framing of AI	“Claude Code was, like, my best friend”; Naming AI in the co-design label

Table 7. Thematic hierarchy for the five developer interviews (RQ1), continued. Part 2 of 2 (themes I4–I6). See Table 6 for the full legend.

Category	Codes
I4 Feature progress coexists with unfinished work and limits to testing. <i>(3 categories, 6 codes)</i>	
I4.1 Feature trajectories and unfinished ambitions	Feature and product history; Feature wanted but not delivered
I4.2 Stated constraints on delivery	Scope, feasibility, law or policy as the stated reason; Platform and infrastructure constraints; Time as the stated reason
I4.3 Limits of testing and refinement	Not enough testing or iteration
I5 Shared development exposes uneven coordination and support. <i>(5 categories, 13 codes)</i>	
I5.1 Staff access and coordination	Staff support: access, responsiveness and ideas; Staff coordination and availability problems
I5.2 Work arrangements and participation	Program conditions that helped; Schedule allocation and interruption; Teacher availability and scheduling mismatch
I5.3 Material provision and equity	Resource and compensation inequity; Equipment and resources reported sufficient
I5.4 Visibility, integration and peer relations	Work not shared or visible between developers; Feature overlap and late cross-developer review; Peers' code practices created work for others; Peers judged positively
I5.5 Program redesign and accountability	Proposals for restructuring the program; Proposals for developer accountability and shared time
I6 Positive retrospective appraisals coexist with qualified accounts of change. <i>(3 categories, 6 codes)</i>	
I6.1 Appraisals of collaboration and program experience	Co-design judged effective; Program judged to have gone smoothly; No challenges reported
I6.2 Reported growth, delivery and uptake	Outcomes and growth reported from the process
I6.3 Qualifications to retrospective accounts	Developer's recall limited; Interview instrument commented on