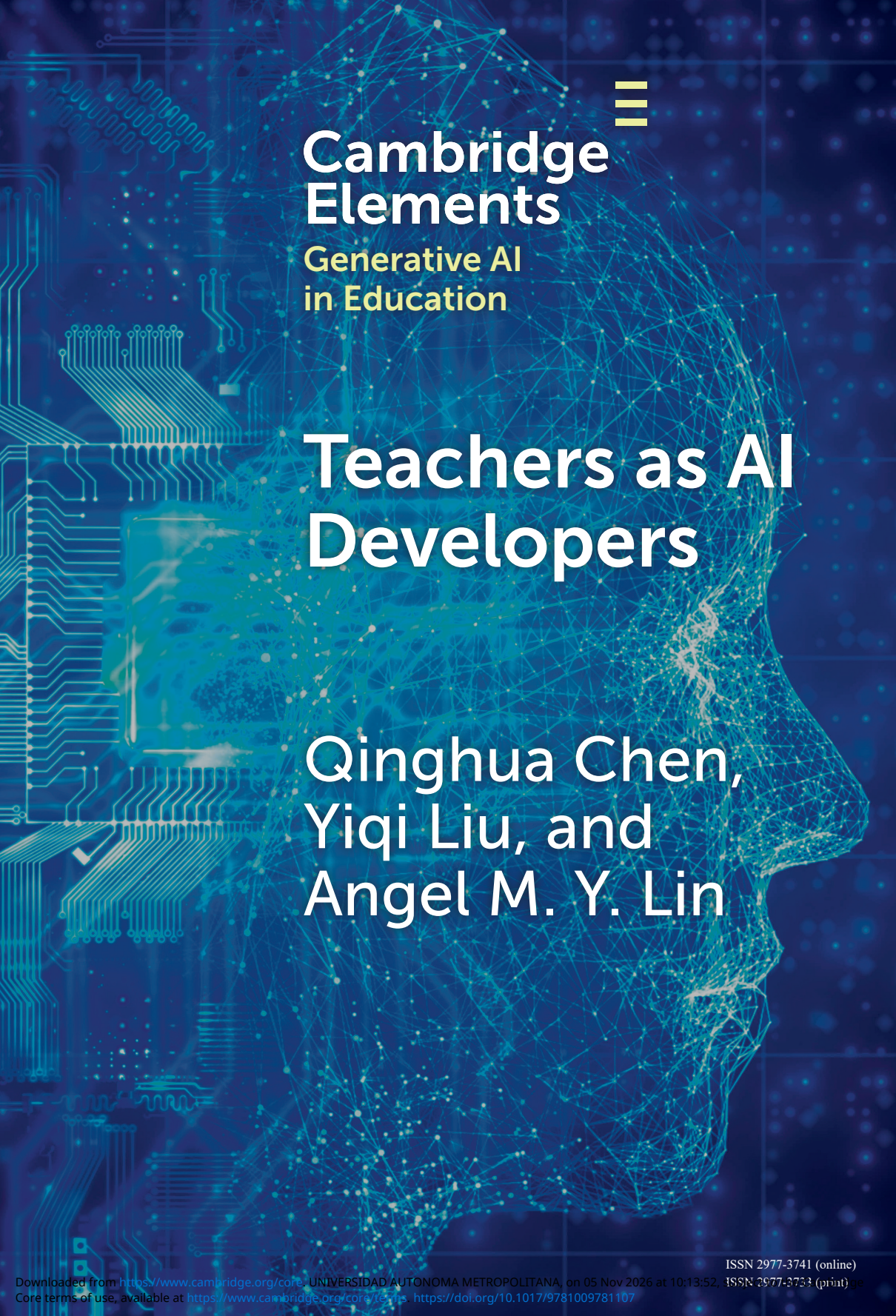


ISSN 2977-3741 (online)
ISSN 2977-3733 (print)

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Cambridge Elements

Elements in Generative AI in Education

edited by

Mark Warschauer

University of California, Irvine

Tamara Tate

University of California, Irvine

TEACHERS AS AI DEVELOPERS

Qinghua Chen

The Education University of Hong Kong

Yiqi Liu

The Education University of Hong Kong

Angel M. Y. Lin

The Education University of Hong Kong

CAMBRIDGE UNIVERSITY PRESS

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

CAMBRIDGE
UNIVERSITY PRESS

Shaftesbury Road, Cambridge CB2 8EA, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre,
New Delhi – 110025, India

Cambridge University Press is part of Cambridge University Press & Assessment,
a department of the University of Cambridge.

We share the University's mission to contribute to society through the pursuit of
education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781009781084

DOI: 10.1017/9781009781107

© Qinghua Chen, Yiqi Liu, and Angel M. Y. Lin 2026

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, with the exception of the Creative Commons version the link for which is provided below, no reproduction of any part may take place without the written permission of Cambridge University Press & Assessment.

An online version of this work is published at doi.org/10.1017/9781009781107 under a Creative Commons Open Access license CC-BY-NC-ND 4.0 which permits re-use, distribution and reproduction in any medium for non-commercial purposes providing appropriate credit to the original work is given. You may not distribute derivative works without permission. To view a copy of this license, visit https://creativecommons.org/licenses/by-nc-nd/4.0

When citing this work, please include a reference to the DOI 10.1017/9781009781107

First published 2026

A catalogue record for this publication is available from the British Library

A Cataloging-in-Publication data record for this Element is available from the Library of Congress

ISBN 978-1-009-78108-4 Hardback
ISBN 978-1-009-78109-1 Paperback
ISSN 2977-3741 (online)
ISSN 2977-3733 (print)

Cambridge University Press & Assessment has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

For EU product safety concerns, contact us at Calle de José Abascal, 56, 1°, 28003 Madrid, Spain, or email eugpsr@cambridge.org

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Teachers as AI Developers

Elements in Generative AI in Education

DOI: 10.1017/9781009781107
First published online: July 2026

Qinghua Chen
The Education University of Hong Kong

Yiqi Liu
The Education University of Hong Kong

Angel M. Y. Lin
The Education University of Hong Kong

Author for correspondence: Qinghua Chen, cqinghua@eduhk.hk

Abstract: The dominance of commercial AI in education has created a paradox: While these tools promise transformation, they perpetuate dependency, financial barriers, and environmental unsustainability. This Element challenges the "technical mystique," the misconception that AI development requires expertise beyond educators' reach, and demonstrates how teachers can become creators rather than consumers of educational technology. The authors chronicle their transformation from linguists and educators to AI developers without formal computer science training, illustrating how the technical barriers educators face are often more perceived than real. Through detailed case studies, including an AI-assisted dialogue system (ADS) for multilingual nurse–patient training and assessment, they demonstrate how teachers can leverage AI coding assistants, browser-based models, and open-source resources to develop pedagogically tailored applications. This title is also available as open access on Cambridge Core.

Keywords: teacher agency, technical mystique, sustainability, learning-through-practice, democratization of AI knowledge, AI literacy

© Qinghua Chen, Yiqi Liu, and Angel M. Y. Lin 2026

ISBNs: 9781009781084 (HB), 9781009781091 (PB), 9781009781107 (OC)
ISSNs: 2977-3741 (online), 2977-3733 (print)

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Contents

1 Introduction: Reclaiming Agency in Educational AI 1

2 Learning to Speak "AI": From Technical Mystique to Pedagogical Fluency 11

3 Navigating Three Types of Barriers: A Systematic Analysis 21

4 Building through Learning and Practice: The Journey from Conception to Working System 35

5 The Sustainability Imperative: Lightweight AI for Education 55

6 Opening a Crack 63

References 78

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

1 Introduction: Reclaiming Agency in Educational AI

This Element weaves together critical analysis of teacher agency, personal narrative of building AI tools, and practical technical guidance. This integration is deliberate, not accidental, because each audience reading this Element needs what the other brings. We ask all readers, including technologists, educational researchers, and teachers, to engage with all three threads, even the unfamiliar ones.

The title of this Element might give the impression of a straightforward how-to guide for teachers interested in AI development. While practical guidance is indeed one of our goals, this Element aims first to examine something more fundamental: what it means for teachers to become AI developers, why teachers might want to assume this role, and what purposes this pursuit can serve, especially when all technical work could seemingly be delegated to AI companies and technologists. In this opening section, we argue that becoming an AI developer offers teachers a pathway to reclaim agency in an educational landscape increasingly influenced, shaped, or colonized by big tech companies' version of AI, usually with very limited teacher input.

To understand what is at stake, consider the story of Yang, an English teacher whose career spans both the pedagogical innovations of the 1990s and the AI disruptions of the 2020s.

1.1 Yang in the 1990s: Claiming Pedagogical Agency

In the 1990s, Yang began his career as an English teacher at a Hong Kong primary school, bringing fresh ideas from his teacher education program and recent professional development workshops. During these workshops, researchers happened to introduce him to "whole body sense-making," a pedagogical approach that enhanced instructional clarity by combining verbal explanations with visual aids and purposeful body language in creating a better English learning environment. Yang was captivated by the strategy's potential and introduced it to his new school.

The principal was supportive and even asked Yang to lead professional development workshops for other English teachers. Yang found colleagues who shared his passion and wanted to form a teaching team around these ideas. But he also encountered resistance from older teachers who, while appreciative of his efforts, saw the approach as "for younger teachers." They cited physical limitations and worried they would be laughed at by students if they adopted Yang's gestural teaching style.

Despite this resistance, Yang persisted. He did not simply implement a method handed down by researchers: He adapted it to his context, built a community of practice around it, and claimed agency over how pedagogical

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

innovation would unfold in his classroom and school. This was teacher agency in action: the capacity to evaluate, adapt, and champion approaches that aligned with his professional judgment about what his students needed.

1.2 Yang in 2023: When Teacher Agency Meets AI

Fast forward to 2023. Yang had spent decades at the same primary school, beloved by students and respected by colleagues. Then one day, he noticed something troubling: Students who typically underperformed in his class were suddenly producing exceptionally good English writing. His first fear was that students were plagiarizing, copying from each other. The truth turned out to be both simpler and more disorienting. His students had discovered ChatGPT.

Those carefully designed assignments crafted from decades of teaching experience and refined through countless iterations to support student learning were being completed in seconds by an AI agent that required no thinking, no struggle, no learning process from the students at all. Everything Yang had built his career around seemed suddenly rendered helpless.

Around this time, Yang's school established a partnership with a university research team to integrate ChatGPT and other AI tools into teaching. Yang was asked to carve time from his already overloaded schedule to attend training sessions on "teaching English with ChatGPT." But in his mind, he already knew how to teach English well without it. This new thing called AI, which everyone now talked about breathlessly, seemed only to devalue his decades of accumulated expertise.

Yang experienced a discomfort eerily similar to what those older teachers must have felt in the 1990s when he had urged them to adopt whole-body sense-making. He recognized, with some irony, that he had long since stopped using those gestures himself. The passionate innovator of the 1990s had become the reluctant adopter of the 2020s.

1.3 The Question of Agency

After reading Yang's story, questions likely emerge. Will AI prove to be just another wave washing through education, like whole-body sense-making and countless other pedagogical innovations before it, exciting on arrival, forgotten within a decade? What transformed Yang from an innovative teacher eager to experiment with new ideas into someone troubled by his students' use of ChatGPT, viewing it as disruptive to his teaching and dismissive of his hard-won expertise?

The answer lies in a concept that forms the core of this Element: teacher agency. The integration of AI into education should not become yet another wave of technological innovation that strips teachers of agency, along with their

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

capacity for innovation and their professional passion. As scholars have long critiqued regarding university-led research interventions, when innovations are done *to* teachers rather than *by* or *with* them, agency erodes.

But here is the crucial difference between the 1990s and now: In the 1990s, Yang could claim agency through pedagogical means alone – by adapting teaching methods, building communities of practice, and advocating for approaches he believed in. In the 2020s, the innovations reshaping his classroom are not primarily pedagogical; they're technological. And unlike pedagogical methods that researchers present for teachers to evaluate and adapt, AI technologies often arrive as closed systems developed by distant tech companies, implemented through top-down mandates, leaving teachers as users rather than agents (International Task Force on Teachers for Education 2030, 2025).

The question Yang faces is not whether AI will influence his classroom; his students have already answered that. The question is whether he can reclaim the same kind of agency over AI that he once exercised over pedagogy.

A Moment to Reconnect 1

Innovation done *to* teachers versus *by* or *with* teachers transcends specific technologies or pedagogies. The enabling condition is important: Teachers have genuine choice and adaptation capacity. The constraining condition is when innovations arrive as closed systems with only top-down mandates; this usually erodes teacher agency.

1.4 How to Read This Element

If you are a technologist or AI developer working in education, you might be tempted to skip the theoretical sections about teacher agency and professional identity. Please don't. The teachers you are building for or, better yet, building with, experience technical work not as neutral tool use but as deeply connected to their professional identity, autonomy, and sense of meaning in their work. Understanding this is not peripheral to your technical practice; it is central to building educational AI that actually serves learning. The theoretical framing helps you see what is at stake for teachers when they engage with your "tools."

If you are an educational researcher or theorist, you might be tempted to skip the technical sections as implementation details beneath the level of critical analysis. Please don't. Theory that remains disconnected from the material practices of building technology risks becoming what teachers often experience as "discussion above the clouds," making it intellectually sophisticated but practically inert. The technical sections are not just examples illustrating theory;

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

they are where theoretical commitments meet real constraints, where abstract critical principles encounter actual decisions, where you discover what your critical analysis actually means when you have to make something work.

If you are a teacher considering whether to engage with AI development or questioning whether you should resist it entirely: You need both what the theorists bring (frameworks for understanding what's happening to your professional agency) and what the technologists bring (demystification of what's actually possible). You should not have to choose between critical consciousness and practical capability. You need both.

The integration itself is the argument. Teacher agency in the AI era cannot be reclaimed through theory alone; rather, you have to build something. But building without critical reflection on purposes and conditions risks reproducing exactly the problems we are trying to solve. And technical work disconnected from teacher identity and pedagogical purposes creates tools that teachers experience as alienating rather than empowering.

So we are asking everyone to stretch into uncomfortable territory. Theorists, engage with code. Technologists, engage with questions of meaning and identity. Teachers, engage with both, not because you must become computer scientists or critical theorists, but because the integration of these perspectives is how we create different futures for educational AI.

The messiness of moving between theoretical argument, personal narrative, and technical guidance mirrors the messiness of actually doing this work. We have tried to make that messiness productive rather than apologizing for it. Stay with us through the sections that feel least familiar because that is often where the most important learning happens.

1.5 The Technical Mystique

One of the most significant factors preventing teachers from claiming agency over technological innovation is what we call the "technical mystique": the widespread belief that AI development requires specialized expertise far beyond educators' reach. This mystique creates a psychological barrier that relegates teachers to the role of consumers, and sometimes dummy ones, rather than creators of educational technology. It suggests that while teachers might adapt new classroom pedagogy (like whole-body sense-making), they cannot possibly engage with the technical substrates of AI systems.

This Element challenges that mystique directly. We argue that the technical barriers teachers face are often more perceived than real, and that educators can become *active creators of AI technology* (Zhai, 2025), developing solutions that are pedagogically tailored, ecologically sustainable, and responsive to their

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

students' actual needs. The path to reclaiming agency in the AI era *does not* require teachers to become computer scientists. It requires demystifying AI development enough that teachers can apply the same adaptive, innovative thinking they have always used in their pedagogical practice (Fengchun & Mutlu, 2024; International Task Force on Teachers for Education 2030, 2025).

Yang's story illustrates what happens when agency is lost. The sections that follow show how it can be reclaimed not by rejecting AI, but by teachers becoming developers of the AI tools that serve their educational communities.

1.6 Why Teachers as AI Developers?

Beyond dismantling the technical mystique, we must address a fundamental question: Why are teachers potentially invaluable contributors to AI in education? The answer lies in understanding what differentiates teacher-developers from commercial AI developers and why that difference matters.

Developers at commercial AI companies often pursue goals centered on scale and competitive advantage: training ever-larger models, topping benchmark leaderboards, and outperforming rival firms' products. These are legitimate business objectives, but they do not necessarily align with the nuanced, context-specific goals that drive teaching and learning in actual classrooms.

Teachers, by contrast, work toward classroom-based and community-specific goals that AI could help achieve but often at a scale and specificity that commercial products overlook. Consider Yang's situation: He finds himself increasingly exhausted by answering the same foundational questions from students, questions that are absolutely necessary for learning but repetitive. Could Yang delegate some of this repetitive work to an AI agent, freeing time for his own professional development or simply for rest and family, without positioning his own needs against his students' learning needs?

Of course, Yang could subscribe to ChatGPT or encourage his students to do so, enabling more self-directed learning. But this solution comes with impacts on student learning and significant financial and environmental costs that deserve scrutiny. Commercial AI services, trained by consuming enormous amounts of computational resources and energy, are undeniably powerful (Muldoon et al., 2024). Yet they may not be the most sustainable solution for teachers' everyday goals and tasks, particularly when similar levels of effectiveness could be achieved with only a fraction of the cost, energy consumption, and environmental footprint.

This realization points to a crucial possibility: Teachers can benefit from acquiring a certain amount of technological knowledge, not an overwhelming amount but just enough to utilize freely shared resources from technology

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

communities with similar goals and far smaller environmental footprints. The question then becomes: What does "just enough" knowledge look like?

1.7 Three Levels of Teacher Technical Proficiency

We propose a framework of three levels of technological knowledge and skills that can enable teachers to accomplish different kinds of tasks and reach varying degrees of autonomy in AI development. These levels are not rigid stages but rather overlapping capacities that teachers can develop based on their goals, interests, and contexts.

1.7.1 Level 1: AI Literacy – Understanding through Metaphor and Analogy

The first level focuses on developing a conceptual understanding of how AI systems work, often through accessible metaphors, analogies, and comparative examples. This level equips teachers to understand the scope of work involved in developing AI-related systems without subscribing to technical mystique discourses that position them at a disadvantage when negotiating with AI companies or technical experts.

In our earlier work, all three authors experienced scenarios where we discussed issues with technology experts, attempting to turn our pedagogical ideas and innovations into practice. We repeatedly encountered situations where technical experts wielded disproportionate leverage by deploying specialized jargon and inflating the complexity of certain development work. We are not suggesting that every technical expert behaves this way, nor that you will necessarily encounter such dynamics. Rather, we are arguing that having a fundamental understanding, including through metaphors, comparisons, and hands-on experience, is essential for teachers and researchers to form collaborative and productive relationships oriented toward shared goals.

At this level, teachers learn to ask informed questions: What does "training" a model actually mean? How do lightweight models differ from large language models (LLMs)? What makes one AI solution more resource intensive than another? This literacy doesn't require coding ability, but it does require enough conceptual grounding to evaluate claims, assess alternatives, and participate meaningfully in decisions about educational AI.

1.7.2 Level 2: AI-Assisted Development – Conversational Proficiency

The second level enables teachers to achieve sufficient technological proficiency to accomplish development work in partnership with AI coding assistants, with a minimal need to depend on external technical experts. At this level, teachers

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

understand fundamental programming concepts and structures, such as what a variable is, what a function does in Python, or how natural language processing pipelines typically work in practice. It is important to emphasize conversational proficiency requires not only understanding these concepts but also *their thematic relations* (Lemke, 1990). For example, when we refer to a website in a conversation with AI, it can misinterpret and assume we are referring to other websites online, rather than the one we are developing, and this can significantly complicate the situation. But if we can say front-end, GitHub, it would be better.

Importantly, teachers at this level don't necessarily write code from scratch or manually modify every line. Instead, their knowledge enables them to communicate more effectively with AI agents to realize their development goals. They can prompt an AI coding assistant with clear specifications, recognize when generated code aligns with their intentions, identify when something seems problematic, and iterate through refinements by asking informed follow-up questions.

This is the level where the learning-through-practice framework becomes most powerful. Teachers do not need formal computer science training; they need enough foundational understanding to engage in productive dialogue with AI for development. The AI handles the syntactic details, while the teacher provides the pedagogical vision and contextual judgment.

1.7.3 Level 3: Code-Informed Adaptation – Autonomous Refinement and Critical Awareness

The third level encompasses everything beyond conversational proficiency – the ability to directly read, understand, and modify code generated by AI for further development and refinement. If an AI assistant has written the code necessary for a classroom teaching system, a teacher at this level can communicate with the AI about debugging specific issues, modify portions of the code directly, or strategically replace sections with newly generated alternatives.

At this level, teachers become truly autonomous developers who can troubleshoot independently, combine code from multiple sources, and iterate rapidly without waiting for external assistance. They can open the hood, so to speak, and tinker with the engine rather than just describing the desired destination. Of course, interested teachers can progress far beyond this threshold, and we have found numerous inspiring cases in our research of teachers who have become sophisticated programmers and addressed technical and sociocultural issues. But this third level represents a meaningful milestone: the point at which teachers are no longer dependent on others to realize their technological visions for their classrooms (Zhai, 2025).

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

1.8 The Power of "Just Enough" Knowledge

Figure 1 demonstrates the three levels with an explanation of the transition phase. These levels are not rigid stages but overlapping capacities that teachers can develop based on their goals, interests, and contexts. Teachers may operate at



Figure 1 Three levels of teacher technical proficiency.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

different levels for different tasks, and progression is not necessarily linear. The framework serves to clarify what kinds of autonomy different levels of technical knowledge enable, helping teachers make informed decisions about how much technical proficiency they want to develop for their specific purposes.

These three levels are not meant to establish a hierarchy where Level 3 is inherently superior to Level 1. Different teachers with different goals, contexts, and interests will find different levels appropriate for their needs. A teacher who primarily wants to make informed decisions about adopting AI tools may thrive at Level 1. Another teacher who wants to create custom classroom applications may aim for Level 2. A teacher who delights in technological tinkering may pursue Level 3 and beyond.

What matters is that teachers have access to pathways for developing the level of technical capacity that serves *their* professional goals, not the goals imposed by commercial vendors or technical gatekeepers. The technical mystique would have teachers believe that meaningful engagement with AI requires expertise far beyond reach. Our framework suggests something different: that "just enough" knowledge, thoughtfully acquired and purposefully applied, can transform teachers from passive consumers into active agents of educational AI innovation.

In the sections that follow, we explore visions for teacher-led innovation and sustainable practice that we have recently come across in our research and collaboration with teachers.

1.9 AI Development as Familiar Pedagogical Practice

Teacher-led AI innovation, as we envision it, actually mirrors processes already deeply familiar to educators – the same exploratory, iterative approaches teachers use when adopting new pedagogical methods or designing curriculum. When teachers encounter a new teaching approach, they do not immediately implement a fully formed system. They go through stages: exploring possibilities, deconstructing what the method entails and what it could achieve, experimenting with adaptations, reflecting on outcomes, and gradually building toward confident implementation.

The process of teacher-led AI development, particularly when working with AI coding assistants, follows remarkably similar patterns:

- Exploration and discovery – teachers begin by exploring what is possible with current AI technologies. This stage involves curiosity-driven investigation: What kinds of educational tools can be built? What problems do existing solutions solve? What gaps remain? This exploration requires no coding ability but the same evaluative judgment teachers apply when assessing any new educational resource.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

- Educational design and conceptualization – the crucial work happens at the design stage, where teachers leverage their pedagogical expertise to envision what an AI tool should accomplish. What learning outcomes should it support? What kind of interactions will benefit students? How should it provide feedback? What contextual factors matter? This is fundamentally educational design work that teachers excel at rather than technical implementation.
- Partnership with AI coding assistants – here is where the paradigm shift occurs: Teachers do not need to write code from scratch. Instead, they communicate their educational vision to AI coding assistants, which handle the technical implementation. A teacher describes the desired functionality such as "I need a tool that helps students practice English conversation with cultural context sensitivity" and the AI assistant generates the code. The teacher's role is to provide clear specifications, evaluate whether the results match their pedagogical intentions, and iterate through refinements.
- Testing and pedagogical refinement – as with any educational innovation, the first version is never perfect. Teachers test AI-generated tools in their classrooms, observe how students interact with them, and identify what works and what needs adjustment. But critically, these adjustments are driven by pedagogical insights, not technical expertise. Teachers communicate needed changes to their AI coding assistant such as "students need more scaffolding at this stage" or "the feedback should be more encouraging" and the AI handles the technical modifications.
- Iteration based on educational insight – the iteration cycle continues, but it remains grounded in educational judgment rather than technical prowess. Teachers are not debugging code; they are refining educational experiences. The AI coding assistant serves as a responsive implementation partner that translates pedagogical vision into functional technology.
- Community and knowledge sharing – teachers can share not just finished tools but also design patterns, pedagogical approaches, and effective ways of communicating with AI assistants. A teacher who successfully created a vocabulary learning tool can document their design thinking and prompt strategies, enabling other teachers to adapt the approach to different subjects or contexts. The focus shifts from sharing code to sharing educational design wisdom.

1.10 From Consumers to Educational Designers

The vision we propose fundamentally repositions teachers as educational designers who leverage AI coding assistants as implementation partners. In this model:

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

- Teachers with Level 1 proficiency (AI literacy) understand enough about AI capabilities and limitations to envision appropriate applications, evaluate existing solutions, and articulate educational needs clearly.
- Teachers with Level 2 proficiency (AI-assisted development) can work fluently with AI coding assistants to develop custom educational tools, translating their pedagogical vision into functional applications through iterative dialogue with AI.
- Teachers with Level 3 proficiency (code-informed adaptation) can directly examine and modify AI-generated code when needed, enabling more sophisticated troubleshooting and refinement which remains optional rather than essential.

Importantly, the center of gravity stays with educational design throughout. The technical implementation becomes a translation process handled primarily by AI assistants, freeing teachers to focus on what they do best: designing meaningful learning experiences.

2 Learning to Speak "AI": From Technical Mystique to Pedagogical Fluency

2.1 Demystifying AI

The barriers teachers face in implementing AI may not lie in understanding the novelty or complexity of how AI works but rather in learning to navigate a techno-social space with conventions and abstractions that lack clear equivalents in everyday experience. Consider, for instance, the difference between a portable document format (PDF) document that allows you to copy and paste text versus one that does not. To our eyes, these two types of documents look nearly identical – we wouldn't usually notice any difference. Without knowing that text might be stored as images in a PDF file, we might develop other explanations for why we cannot simply copy and paste certain sections of text from a PDF file. And with advances in some PDF readers that now automatically perform OCR during copy–paste actions, the distinction blurs further. We are left puzzled when the copied text appears garbled or improperly formatted, not understanding why it works smoothly in some cases but not others. We then explore other possible solutions that are destined to fail, such as that the mouse might not be working properly or the PDF reader needs to be removed and reinstalled.

This is usually where frustration sets in and self-questioning begins: "Ah, I'm not a tech-savvy person" or "This isn't for me." Gradually, these small but repeated incidents, moments when we feel we cannot perceive what's actually

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

happening, accumulate into perceived barriers. The technical dimensions of work seem impossibly difficult, and we conclude that we should leave it to the technology experts. Yet these are not personal failings but rather just some gaps in our understanding of some invisible technical distinctions. Similar significant differences hiding behind seemingly identical things pervade development work. Here's another example: In Python programming, the number '5' (in quotes) and the number 5 (without quotes) may look nearly identical on screen, but they represent fundamentally different data types: '5' is text (a string), while the other (5) is a numeric value. Try to perform mathematical operations on '5' and you'll encounter cryptic error messages which, without this context, seem to confirm your worst fears about your technical inaptitude.

These invisible technical distinctions are not arbitrary quirks. They are the sediment of decades of specialized problem-solving, in which each solution is optimized for technical efficiency rather than intuitive understanding. For instance, PDF technology was developed in the early 1990s to ensure documents would display identically across different computers, even when recipients lacked the original fonts or software (hence the name PDF: portable document format). This "portability" was revolutionary for its time, but it created a fork in the road: PDFs could either prioritize text extractability (storing actual text data) or visual fidelity (essentially creating a picture of the page). When documents are scanned from paper or created by certain applications prioritizing appearance, they take the "image" path, looking perfectly readable to the human eye while being invisible to copy–paste functions. Similarly, today's AI systems emerged from a fundamental paradigm shift in how we approach language understanding. Earlier systems (from the 1960s through the 1990s) relied on "natural language processing" (NLP) linguists and programmers painstakingly coding grammatical rules and logical relationships. Modern generative AI (GenAI) took a radically different path: Rather than explicit rules, these systems learn statistical patterns from vast amounts of text, developing their own implicit "understanding" through exposure. Therefore, AI "understanding" is not exactly the same as human understanding but perhaps more in the sense of statistical pattern associations. Each of these technical evolutions solved real problems for their creators, but in doing so, they created new layers of abstraction: new "grammars" that diverged further from everyday intuition. The result is a techno-social landscape shaped by the accumulated decisions of specialists solving specialist problems, leaving non-specialists to navigate a space that was never designed to be compatible with their intuitions.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

2.2 From Mystification to Professional Disempowerment

Teachers encounter invisible technical distinctions repeatedly. For example, when a tool works perfectly in a tutorial but fails mysteriously in their hands (e.g., a PDF file does not release its text for the copy-and-paste function), the error message seems to blame them for something they do not understand. This is when the technical mystique becomes internalized. What begins as an external perception ("AI development requires specialized expertise") gradually transforms into a personal belief ("I lack the capacity for this kind of work").

This pattern resembles what researchers term learned helplessness, a phenomenon where repeated experiences of uncontrollable outcomes lead individuals to stop trying to exert control. First identified by Seligman and Maier (1967) in psychological research, the concept was subsequently applied to educational contexts by scholars such as Dweck and Repucci (1973) and Fincham et al. (1989), who demonstrated how this dynamic manifests in classroom settings where students develop stable beliefs about their inability to succeed despite having the capacity to do so. However, what teachers experience is more bounded and specific: a domain-specific erosion of what educational researchers call "technological self-efficacy" (Bandura, 1997; Wang et al., 2004), the belief in one's capability to successfully engage with technology. The paradox is striking: A teacher who confidently masters complex pedagogical frameworks, who navigates intricate classroom dynamics, who advocates effectively for students, may simultaneously hold a firm belief that "I am not a tech person." This narrowing of perceived capability does not reflect actual inability. Rather, it reflects the accumulated weight of frustrating navigation in techno-social spaces that were never designed to be intuitively navigable by nonspecialists.

The technical mystique thrives in this psychological space. Each unexplained failure becomes evidence that "I'm not a tech person," that "this is beyond me," that "I should leave this to the experts." What begins as a reasonable response to confusing feedback calcifies into a stable belief about personal capacity: "I *can't* understand what's happening." The distinction is crucial. The former is a gap in conversational fluency that can be developed; the latter is a perceived limitation in ability that becomes self-fulfilling, preventing the very learning-through-practice that could build fluency.

But here is what makes this particularly consequential in educational contexts: Teachers are not just individuals struggling with technology in private. They are professionals whose relationship with learning is constantly on display. When faced with a technical obstacle in the classroom, such as struggling to display a PowerPoint in full screen or being unable to connect to the

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

projector, the teacher's identity seems to demand transparency and honesty about their limitations. Many teachers respond by delegating these seemingly simple technological tasks to students who "know the tricks." While this solves the immediate problem, it comes at a hidden cost. The technological knowledge required becomes reframed as mere "tricks" that young people would "naturally" possess or dismissed as "just technical details" peripheral to the real pedagogical work that remains firmly in the teacher's agentic control.

Both patterns, attributing technical fluency to youth as an innate quality and minimizing technology as auxiliary to "real" teaching, create an artificial and unnecessary boundary around teacher agency. They prevent teachers from recognizing that technological capability could be as much within their professional domain as curriculum design or classroom management. The result is a self-imposed limit on what teachers believe they can control: Pedagogical goals remain theirs, but the technological means to achieve those goals are ceded to others, be it tech-savvy students, IT departments, or commercial platform providers. What begins with "Can someone help me get this in full screen?" can gradually expand into "I'll let the experts handle the AI tools."

It is important to point out that there is nothing inherently wrong with these choices. Every teacher must decide their own comfortable level of technological engagement, and delegating technical tasks can sometimes be a perfectly reasonable division of labor. However, if you find yourself sensing that these arrangements have become limiting, that is, if you have begun to wonder whether your pedagogical vision is constrained by what you believe you cannot do technologically, then it may be worth revisiting the three levels of AI engagement we discussed in the previous sections. The question is not whether you *should* want to move to a different level, but whether you *could* if you wanted to, and whether current beliefs about your capabilities are based on actual limitations or on the accumulated mystification we've been examining. Of course, the danger is that we simply replace one mystique with another – celebrating the "teacher-developer" as the new hero narrative. The point is not to romanticize coding or educational AI development but to keep it within professional reach, just as learning the assorted tricks for showing a PowerPoint in different rooms is within reach, rather than proof of some innate technical gift. The story that follows traces the thinking and planning process behind developing an AI-assisted dialogue system (ADS) for nursing education, from its initial conception through Version 1. Along the way, we will illustrate how we gradually learned the "grammar" of AI development – how we became more familiar with the techno-social space created by AI specialists and learned to articulate our pedagogical goals in ways that allowed AI agents to help us achieve them. The technical implementation details of this project and its

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

subsequent iterations (Versions 2, 3, and 4) can be found in Section 4. In this section, we focus on the bigger picture of the learning process itself: how we moved from feeling mystified by AI technology to reclaiming our agency as educators who could shape technology to serve our pedagogical vision.

2.3 Breaking Down Our Project: How an ADS for Nursing Education Was Built Using Basic Web Technologies and AI Libraries

Recently, our research team launched an innovative project: creating an ADS for English-medium nursing education. This system, developed for our nursing educator collaborators and their students, embodies our research-informed understanding of authentic nursing communication. Drawing on years of ethnographic research in content- and language-integrated learning within Hong Kong's English-medium nursing programs (e.g., Liu & Lin, 2025) and grounded in our theoretical framework of ecological languaging competencies, we conceptualize nurse–patient communication as situated within a holistic, ecological system. This approach reflects the plurilingual and pluricultural scenarios nurses encounter in everyday practice.

2.3.1 From Theory to Technical Requirements

Our ethnographic observations in Hong Kong hospitals revealed something crucial: Nurses and patients naturally translanguage between Cantonese, English, and Putonghua depending on the medical context and emotional register. A patient might describe symptoms in Cantonese, use English medical terms picked up from doctors, and then use Putonghua when speaking to Mainland relatives. Moreover, we observed that communication breakdowns rarely stemmed from simple language barriers: They emerged when nurses missed emotional cues embedded in how patients spoke – the hesitation cues that usually mask fear, the rushed speech betraying anxiety, the formal politeness concealing pain.

These insights shaped our technical requirements directly. We envisioned a system that could capture not only what student nurses said but also analyze the possible emotions implied when they spoke those words. We wanted students to experience situations where patients were in pain, dealing with anxiety due to illness, and unable to communicate their needs clearly, as well as situations requiring significant emotional labor from nurses (Hochschild, 1983), such as the management of feelings, to create publicly observable emotional displays in professional contexts. Our ideal training system needs to achieve two key goals: (1) Simulate realistic hospital

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

scenarios of nurse–patient interactions that honor this multilingual, emotionally complex reality, and (2) analyze and assess how well nursing students tackle these situations by recognizing patients' possible emotions (fear, nervousness, pain, anger) and responding accordingly while applying health science knowledge and healthcare principles. As critical applied linguists, we understood the importance of this approach for improving both nursing education outcomes and patient care. We hope that such a system could enhance nursing students' "on-the-ground" professional communication in intercultural contexts (Dai, 2024).

2.3.2 The Collaboration Challenge

However, we faced a significant challenge. Although at different career stages, we were all trained as linguists and social semioticians, not computer scientists. We had very limited knowledge of coding, data analytics, or AI system implementation. Our natural instinct was to collaborate with people who possess the technical expertise to help translate our theoretical innovation into a working system. We initially consulted internal information technology (IT) collaborators from our own institution, who were genuinely helpful but also fully occupied, as they were being pursued by many departments across the university. We then explored external collaboration with AI developers and startup companies in the region who might be interested in contributing to our research. Yet these paths proved to be even more difficult.

No one was at fault. Our IT colleagues were already deeply committed to their own priorities, and genuine collaboration would have required time to build a shared understanding. In this moment, the capacity simply was not there, which made it hard to fully translate our educational goals across disciplines. Our IT colleagues were well-meaning and eager to help; however, given their training, the theoretical principles underpinning our educational approach were not always readily appreciated in ways that would support the project's success.

2.3.3 The Decision to Learn

We were left with only one option: We had to start from scratch and attempt the development work ourselves, preparing ourselves for substantial learning in the process. Learning to do the development work ourselves became the most viable option, not because it was the easiest path but because it was the only one that could give us agency to realize our pedagogical vision. We did not find this decision to be empowering at first. It felt daunting, perhaps even reckless, but we moved on to this path.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

2.4 The Learning Journey Begins

It would be tedious and unhelpful to paste our entire chat record with various AI systems here. Instead, we have analyzed our conversations and identified the key learning breakthroughs: the moments when mysterious technical concepts suddenly clicked into place.

We communicated our theoretical perspective and our vision for the system to GenAI, asking it to suggest possible pathways we could pursue to achieve our development goals step by step. During this process, we also made clear what we did *not* want to use, such as solutions that would lock us into expensive commercial platforms or require server infrastructures. Through these initial conversations, we encountered many technical terms we were not familiar with, but the journey began.

2.4.1 Learning Milestone 1: Understanding the Front-End and Back-End Architecture

Our first major conceptual breakthrough came when we learned that any web-based system has two fundamental parts: a "front-end" and a "back-end."

A Moment to Reconnect 2

Before we dive into specific technical concepts, pause for a moment. Think about how you organize your classroom materials – perhaps you have what students see (your displayed slides, handouts, the learning activities) and what students don't see (your lesson plans, assessment rubrics, your notes about individual student needs). You're already managing a "front-end" that students interact with and a "back-end" that structures their experience. The technical concepts we're about to explore aren't foreign territory – they're ways of naming organizational patterns you already understand from your teaching practice.

The "front-end" was immediately familiar, which is the webpage interface where users interact with the system. As educators, we have been using front-ends for years: the online forms we fill out, the learning management systems, such as Moodle, where we post course outlines and assignments, and the web-based tools we use daily. The system we were building would have a front-end just like any other webpage (like the one we have eventually developed: https://nursing4.netlify.app). In terms of our "grammar" metaphor, learning the term "front-end" was like learning vocabulary in a new language when suddenly a concept we intuitively understood had a name.

The "back-end" was less intuitive but became clearer through analogy. If the front-end is like a restaurant's dining room where customers interact with staff,

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

the back-end is the kitchen where the actual cooking work happens. In our system, the back-end would be where the AI processing happens, where student responses were analyzed, and where the logic for generating appropriate patient reactions lived.

2.4.2 Learning Milestone 2: Navigating Deployment Decisions

Understanding front-end and back-end architecture led to our second major learning challenge: deployment decisions. We needed to make several interconnected choices: Should the system be available via the internet or only within a local network? Should we install it on a local computer, say the office computer of one of our team members, or host it "in the cloud"?

At first, these felt like overwhelming technical questions. But through discussions with AI and within our team, we realized these were not really technical questions at all. In fact, they were pedagogical questions in technical clothing. We were really asking: Who needs access? How many simultaneous users are there? What is our budget? What technical support can we rely on? These were questions we could answer based on our educational expertise.

Making the system available online would certainly increase accessibility, but it would add many layers of network complexity: connectivity issues, security concerns, and data privacy considerations for patient simulations. For local deployment (developing the system on our own computers), we would need hardware capable of running lightweight LLMs offline, which provides more control over data and incurs fewer ongoing costs.

This was our second breakthrough: realizing that choosing between technical approaches is not about identifying the "correct" answer but about understanding trade-offs and making decisions that align with our priorities. After several rounds of discussion, we decided on a two-step approach that would let us learn progressively:

Step 1: Create a cloud-based system for easier development and rapid iteration, aiming for a minimum viable product (MVP) that can:

- process Cantonese, Putonghua, and English (reflecting the plurilingual reality in our communities)
- respond in natural human voice to trainees' speech (creating more realistic interactions)
- work on both desktop and mobile environments (maximizing accessibility)

Step 2: After achieving Step 1 goals, purchase the necessary hardware for local deployment to facilitate these additional educational goals:

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

- manage at least twenty-five concurrent users within a local network (supporting classroom use)
- upgrade the AI component to other open-source models, preferably pretrained for medical discourse (improving accuracy)

This two-step process greatly reduced the initially daunting amount of work, helping us navigate the technical complexities one layer at a time rather than all at once. More importantly, it gave us permission to start imperfectly, to build something that works adequately first, and then improve it. This was perhaps our most important psychological breakthrough: Perfectionism is the enemy of our progress.

2.4.3 Where We Stand Now

By the time of this writing, we are concluding Step 1. We currently have an online system capable of handling clinical nursing scenarios, including taking patient information, checking for wounds, cleaning wounds, comforting patients when necessary, and providing important post-treatment instructions. It is not perfect as there are still issues: for example, speech latencies to reduce, emotional responses that could be more nuanced, edge cases that could cause unexpected behavior. But it works. We are soon to have our nursing educator collaborators piloting it with their students. And we have built it ourselves.

2.4.4 Learning Milestone 3: Understanding APIs and Hidden Complexity

Our third major learning breakthrough, perhaps the most humbling one, came when we discovered APIs (application programming interfaces). This concept revealed just how much complexity can hide in the back-end.

We learned that we could use APIs to access services we do not have to run ourselves. For example, if we want our nursing ADS to recognize trainees' voices, we do not need to build a voice recognition system from scratch. We can use an API to send the voice to a service provider like Azure (Microsoft's voice engine), and the voice recognition system would return the result as text. This seemed magical at first: We can leverage sophisticated technology just by learning how to "call" it through an API.

In Version 1 of our system, we used Azure's voice recognition service, which did an excellent job understanding and transcribing speech mixing Cantonese and English. It also generated high-quality voice output for the patient's words. We felt triumphant as we would build a multilingual, voice-interactive system! Then we tested it with actual users, and the euphoria evaporated. The system felt

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

sluggish, unnatural. Each interaction took four to five seconds to complete. Students would speak, then wait in awkward silence before the "patient" responded. The system worked, but it did not work well enough.

2.4.5 Learning Milestone 4: Iterating through Problems

This is when we confronted what might be the most important learning milestone: understanding that development is iterative, that "working" is not the same as "working well," and that problems are not failures but information to facilitate further improvement.

We traced through what was happening in each interaction and discovered the hidden complexity:

> System records trainee's voice → Voice sent to Azure via API →Azure processes voice and generates text → System receives output via API → System processes output via AI (Transformers.js) → Text generated by AI sent to Azure via API → Azure generates response voice → Voice played through system

Eight steps. Multiple round trips across the internet. No wonder there was a delay. The problem was not with our code; it was with our architecture. We needed a fundamentally different approach. It is important at this point not to return to thoughts like "I'm not a tech person" and "I should hand this off to 'real' developers." But try to internalize a different mindset: Problems have solutions, and we can find them through systematic experimentation and learning.

We experimented with multiple approaches: running some AI processing locally in the browser, using different voice models, restructuring the conversation flow. Some approaches failed. Others improved things marginally. Gradually, through iteration, we found certain combinations that worked better. We will leave the technical details of these solutions to Section 4, but what matters here is the learning that has happened: We have developed enough conversational fluency in the techno-social space to *diagnose* the problem, *research* potential solutions, *experiment* with approaches, and *evaluate* results. This was our professional identity shift in miniature: from feeling helpless in the face of technical problems to feeling capable of working through them.

2.4.6 From Experience to Framework

At this point, if the preceding text still sounds somewhat dense or unfamiliar to you as a teacher, that's perfectly fine and actually important to acknowledge. We are not suggesting that reading about our experience will instantly make you fluent in technical development. What we are suggesting is that these concepts

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

are *learnable through practice*, that the barriers are lower than they appear, and that the learning process, while challenging, is not fundamentally different from other professional learning we have undertaken to become teachers.

In reflecting on our journey, we have identified three types of barriers that we had to overcome, and that we believe other educators will encounter: conceptual barriers (understanding unfamiliar abstractions like APIs and deployment architectures), decision-making barriers (choosing between technical approaches when you don't yet understand all the trade-offs), and iterative learning barriers (knowing how to experiment, encounter problems, and persist without giving up or concluding "this isn't for me"). Let us examine each of these barriers more systematically in the next section, using our experience as concrete illustrations of how they can be navigated.

3 Navigating Three Types of Barriers: A Systematic Analysis

In this section we shall describe three distinct but interrelated barriers that we have encountered on our journey from language teachers to AI developers and share our experiences in overcoming them. Understanding these barriers and recognizing them as navigable rather than insurmountable is crucial for other educators considering similar paths. What made these barriers particularly challenging was not their inherent difficulty but rather that they required us to reconfigure our understandings derived from everyday life experience to fit the logic of the techno-social space. We will discuss these barriers and how they affected our journey of educational AI development, as well as some of the strategies we found useful in overcoming them.

3.1 Barrier 1: Conceptual Barriers – Learning to Think in Technical Abstractions

Conceptual barriers emerge when technical concepts do not map neatly onto anything in our everyday experience. These are not "hard" barriers in the sense of requiring advanced mathematics or computer science degrees, but they require learning the community of practice, including thinking and talking about issues in the techno-social space. As pointed out by Lemke (1990), it requires not only understanding individual concepts but also grasping the thematic relations between these concepts.

For example, after we explained our needs to our AI coding assistant, we got an AI-coded HTML (HyperText Markup Language) file. This file, containing both content and layout information and widely used in coding webpages, in fact needs to be "deployed" through services like Netlify, which uses its server space to publish the website on the internet without

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

requiring us to manage server-side issues (web servers, networks, etc.). The word "deploy" already contains many important steps that are not known to many educators. Recall our earlier discussion of the techno-social space: These are the "levels of abstraction" that are hard to find concrete equivalents in our everyday life. The challenge is not cognitive complexity; it is conceptual reorganization and compression, especially regarding the thematic patterns involved.

We further categorize these conceptual barriers into three domains: web-related concepts, programming-related concepts, and AI model–related concepts. We offer key examples from each rather than exhaustive lists.

3.1.1 Web-Related Conceptual Barriers

Consider the concept of "hosting." In everyday life, hosting can mean having people over to your home. But in web development, "hosting" refers to where your website's files physically exist on a server computer somewhere in the world (typically in a data center). When we read the AI output saying we had to decide where to "host" our application, we initially did not quite understand that "host" in this context actually means finding server space online where all the files related to our application are stored and where code (or "instructions") is executed. The term was familiar, but its technical meaning required reorganizing different thematic patterns: from hosting your family members (providing accommodation to family members) to hosting your web application (renting server space for storing website-related files and allowing users to access them there). Further, there is crucial distinction between client-side and server-side processing. In web development, these terms describe *where* computation happens, on the user's device (client-side) or on a remote computer (server-side), usually in a data center. This distinction became critical when we discovered that some operations we want cannot be performed in the user's browser for security reasons, so they have to run on the "server-side." We gradually picked up these concepts but the breakthrough came when we realized these were not arbitrary technical jargon meant to exclude outsiders. They were precise terms that the web development community needed to communicate efficiently. Once we stopped feeling intimidated by the unfamiliar vocabulary and started treating it like learning technical terms in any new field, the concepts became approachable.

Below is a list of other web-related concepts we found useful to understand. If you are interested, AI can explain these terms in detail, but we encourage you to specifically ask AI to explain the thematic relationships between related concepts. We found this a rewarding way to learn through exploration:

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

- **Front-end vs. back-end** – the visible interface that users interact with versus the behind-the-scenes logic and data processing
- **HTML, CSS, and JavaScript** – the three core technologies for webpages (structure, styling, and interactivity)
- **Domain name** – the web address that users type (like "nursingtrainer.com")
- **HTTPS vs. HTTP** – secure versus nonsecure web connections
- **Browser** – the application (Chrome, Safari, Firefox) that interprets and displays web code
- **Responsive design** – making websites work well on different screen sizes (desktop, tablet, mobile)
- **CDN (content delivery network)** – a system that delivers your website files quickly to users worldwide
- **Build process** – converting your development code into optimized files ready for deployment
- **Local development** – testing your website on your own computer before making it public

But of course, without the relevant hands-on experience of using these concepts, they can still sound raw and dry, so please do not get overwhelmed if you cannot fully understand each one of the above; this applies to the other two lists that follow (programming-related conceptual barriers and AI model–related conceptual barriers). We will continue with this journey first so as to open future possibilities.

3.1.2 Programming-Related Conceptual Barriers

Programming introduced a different kind of conceptual challenge: learning to think with extreme literal precision in ways that initially felt alien. This is not to say as linguists we are not aiming for precision in our language; rather, we think that human language allows a certain open-endedness instead of fitting words or expressions in exact form. This, we argue, also allows creativity and innovation in expressing the nuances of human lived experiences. Yet, the programming language does not seem to have this function.

Specifically, the most fundamental shift for educators was understanding that computers are extraordinarily literal. In human communication, we rely heavily on shared context and implicit understanding. If I ask a colleague to "Please review this draft," they understand I mean the document we have been discussing along the way, that "review" means providing substantive feedback, and that I would like it within a reasonable timeframe. A computer, by contrast, needs explicit instructions for every single step. The concept that caused us the most confusion was the distinction between text and numbers – our earlier example

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

of '5' versus 5. This became viscerally real when we tried to calculate average pain scores from our simulated patients. We collected scores from students, tried to average them, and got bizarre results. The system was treating them as text and concatenating them ('5' + '3' = '53') rather than adding them mathematically (5 + 3 = 8). The data looked identical on screen but was fundamentally different to the computer. What made this particularly confusing was that in human communication, knowing when to treat numbers as words and when to perform calculations on them is largely intuitive. We do not consciously think "I need to convert this to a numeric type before adding" as we simply understand from the context what operation makes sense. Hours of frustrated debugging finally led to understanding: to the computer, these are completely different types of information, even though to our eyes they appear the same. Luckily, some coding applications, such as VS Code, are now able to detect these issues and mark them out easily.

Another major conceptual shift involves asynchronous operations and turn-taking. In everyday face-to-face conversation, turn-taking rules are assumed and usually agreed upon: I ask you a question, you answer, then I respond. We use social contextual cues such as pauses, eye contact, and intonation to signal when we have finished speaking and when should be the other person's turn. But when we sent a voice recording to Azure for transcription, we discovered that the system had no inherent understanding of conversational turns. It could not automatically know when the user had finished talking and when it should process and respond to students' talk. In our initial deployment, the system would keep "listening" and "responding" simultaneously without understanding the boundaries between speaking turns. This caused the system to break down: It might start responding while the user was still talking, or keep listening indefinitely while the user waited for a response. We had to explicitly design a mechanism to signal turn completion. We first tried using a three-second pause to indicate that the user had finished speaking. But this created new problems: Users would assume the system had not heard them and would start repeating themselves before the three seconds elapsed, causing the system and user to talk over one another again. In the end, we found a better but still not ideal solution: adding a button that users could click to signal "I'm finished speaking." This felt less natural than voice-only interaction, but it provided clear, unambiguous turn boundaries.

This experience required us to think in terms of parallel timelines and explicit signaling rather than assuming the linear sequences and implicit social cues we are accustomed to in everyday conversation. What seems like a simple technical problem – voice interaction – is actually a complex challenge of translating human conversational norms into explicit computational rules.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Other key programming concepts we encountered include:

- **Variables** – containers that hold information (but only one value at a time)
- **Functions** – reusable blocks of code that perform specific tasks
- **Loops** – ways to repeat operations multiple times
- **Conditional statements (if/then)** – how programs make decisions
- **Arrays and objects** – different ways to organize multiple pieces of related information
- **Error messages** – feedback from the computer about what went wrong (often cryptic but learnable)
- **Debugging** – the process of finding and fixing problems in code
- **Libraries and frameworks** – pre-written code you can use to avoid building everything from scratch

3.1.3 AI Model–Related Conceptual Barriers

AI models introduce perhaps the most counterintuitive concepts because they violate common assumptions about how computers typically work. The first major concepts were context windows and memory. We initially assumed that our patient simulator would naturally "remember" information the trainee had already provided just like remembering a patient's name because this is how systems like ChatGPT appear to work from a user's perspective. But we discovered that AI models do not actually have memory in the way humans do. Each interaction is independent unless you explicitly include previous context. This explained why our chatbot simulating a patient would sometimes inexplicably regress to earlier stages of the interaction, such as asking for identification information again while wound cleaning was being performed, for instance. The revelation: We need to include possibly the entire conversation history in each new interaction with the AI model. This was like having a conversation with someone who had amnesia and needed you to recap everything that had been said so far before each new exchange. Understanding this fundamentally changed how we structured our code: We had to maintain and continuously update a conversation history that got passed to the AI model with every interaction.

The second major shift was understanding that AI models work through probability, not rules. In everyday thinking, we expect computers to follow precise rules: if condition A, then do B. But AI LLMs generate responses based on statistical patterns rather than following programmed instructions. This meant we could not simply tell the model "always respond with empathy

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

when the patient expresses pain." Instead, we had to craft prompts that made empathetic responses more statistically likely.

This shift from rule-based to probability-based thinking required substantial conceptual adjustment and explains why the AI's responses were sometimes unpredictable, occasionally generating responses that did not quite fit the scenarios we designed. This unpredictability has also shaped our future development directions. We opened a parallel route using pre-recorded voice responses with a rule-based decision tree, which would eliminate AI unpredictability entirely. We planned to compare these two approaches to determine which works better for our pedagogical purposes: the more natural but occasionally unpredictable AI-generated responses or the more consistent but potentially less adaptive pre-recorded responses.

A third crucial concept was understanding model size and capability trade-offs. We learned that AI models come in vastly different sizes, and size correlates with not only capability but also computational resource requirements. A larger model might better understand nuanced medical terminology but require expensive cloud processing, while a smaller model could run directly in a web browser but might struggle with specialized vocabulary. Understanding that this is a trade-off landscape rather than a simple "better versus worse" has helped us make informed decisions about which models to use for different purposes. Different use cases might call for different model choices, and we do not need to find the "best" model but rather the most appropriate model for each specific context.

Additional AI concepts worth understanding:

- **Tokens** – the units that AI models use to process text (roughly similar to words and subwords, but not exactly – typically three to four characters per token)
- **Temperature** – a setting that controls how creative or predictable the AI's responses are
- **System prompts** – background instructions that shape the AI's overall behavior
- **Hallucinations** – when AI models generate plausible-sounding but incorrect information
- **Fine-tuning** – adapting a pre-trained model for specific purposes
- **Open-source vs. proprietary models** – models freely available versus those requiring payment or licenses
- **Local vs. cloud-based models** – models that run on your device versus those accessed through the internet

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

A Moment to Reconnect 3

If those lists felt overwhelming, here's what matters: You didn't need to understand all those terms to teach successfully yesterday, and you don't need to understand them all today. But knowing that they exist and knowing there's vocabulary for these technical distinctions changes something important. When you encounter a problem, you now know there are nameable concepts that might help you solve it. You're not lost in undifferentiated technical fog; you're navigating terrain that has landmarks, even if you don't know all their names yet. That shift from "I have no idea what's happening" to "I don't understand this specific thing, but I can learn about it" is the shift from mystification to agency.

The conceptual barriers across all three domains share some common features: They require reorganizing the thematic patterns of everyday language for techno-social space; they make distinctions less invisible to human perception but crucial to technical functioning; and they organize information according to computational logic which can sometimes be sharply different or verbose when compared with human perception and intuition. But critically, none of these concepts require mathematical sophistication or innate "tech talent." They require us to just hang in there, get repeated exposure, and be willing to revise and diversify our everyday assumptions for the purpose of learning, a skill that educators already possess from learning any new disciplinary frameworks.

3.2 Barrier 2: Decision-Making Barriers – Navigating Trade-offs with Incomplete Understanding

If conceptual barriers involve learning new ways of thinking, decision-making barriers involve *acting* on incomplete understanding and making consequential choices before fully grasping all their implications and being ready to accept painful but sometimes necessary setbacks as lessons learned. This is, in many ways, the most psychologically difficult barrier we faced because it can easily leave us disheartened and tempted to surrender to the discourse of "this is not for me." As linguists and educators, we are comfortable in our own field and are able to articulate precisely what we want to achieve, such as describing in rich detail the kind of nurse–patient interaction we want to simulate, including the emotional nuances we want students to navigate. But translating these clear educational visions into technical implementations requires constant compromise, and we often cannot fully understand the trade-offs we are making until afterwards.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

3.2.1 The Disheartening Reality of Technical Constraints

The most emotionally challenging aspect was encountering situations where the system simply would not work the way we had envisioned, not because our pedagogical reasoning is flawed but because of technical limitations we did not anticipate. This may sound counterintuitive because AI now seems to many as being very capable; however, if we zoom in closely on our journey, we get to know that tiny technical blockages can have the potential to hold back a great program. Consider the decision whether to use Microsoft's voice service Azure for voice interaction. Pedagogically, we knew that natural conversational flow is crucial for authentic nurse–patient simulation. Real nursing interactions happen in real time, with immediate responses to patient distress, natural turn-taking, and appropriate pauses. We communicated this clearly to ChatGPT to give us advice on system structure: "We need natural conversation with minimal delay." But when we tested the system, the four- to five-second delay between trainee speech and patient response was pedagogically unacceptable. The problem is not that we implemented it incorrectly, but that the system has to send data back and forth to cloud services multiple times for each interaction, and this inherently takes time.

We then faced a painful decision matrix with no obvious, correct answer:

- Keep the high-quality voice services but accept delays that undermine pedagogical authenticity?
- Switch to lower-quality robotic sounds but faster alternatives that process speech and sound locally in the user browser?
- Abandon voice interaction entirely, sacrificing the vocal emotional cues central to our pedagogical framework?
- Try a combined (remote and local) sound processing approach but with no guarantee that it would work better?

This was not a technical decision that we were unqualified to make; instead, it was a *pedagogical* decision meshed with technical ones. But we could not evaluate the trade-offs without understanding the technical constraints, which can take weeks to fully grasp. We would like to emphasize that although by consulting with technical experts, we may have been able to avoid some of the so-called failures we experienced at several of our initial iterations, we would also have lost the learning opportunities we now see as necessary for teachers learning to become AI developers. So, rather than calling them failures, we would like to accept them as necessary experiences in climbing the learning curve, and it is important to just bear with the uncertainty during the learning process.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

3.2.2 The Psychology of Uncertain Decisions

What made decision-making barriers particularly difficult was the psychological weight of uncertainty. In our usual scholarly work, we can research thoroughly before committing to positions. Decisions then feel grounded in expertise. But in technical development, we feel we are constantly making decisions at the edge of our understanding:

- "Should we store data in this way or that way?" (We barely understood the options, let alone which was better.)
- "Should we add security features now or later?" (We didn't fully grasp what they entailed or why they mattered.)
- "Should we use this framework or that one?" (We'd just learned that these were different approaches but we could not evaluate their relative merits.)

Each decision point triggered internal doubt: "Are we missing something crucial? Is there an obviously better approach that any experienced developer would choose? Are we making just another beginner's mistake that will cause problems later?" The discourse of the technical community sometimes amplified this anxiety. Online forums are full of confident pronouncements: "Never use X" or "Always choose Y." But these often reflect specific contexts or personal preferences, something we cannot easily discern as beginners. We read conflicting advice with no reliable way to evaluate which specific guidance applied to our situation. These moments can again be disheartening. We have clear pedagogical goals and we knew exactly what educational experience we wanted to create. But the technical reality kept forcing compromises. There were moments of real frustration: "If I have to trade away another feature that is pedagogically important because of technical limitations, what is the point? Are we just building a poor version of what we imagined?"

3.2.3 Learning to Decide through Experimentation

Our breakthrough came when we reframed decision-making from "choosing the correct approach" to "choosing an approach to test." We began asking different questions:

- Instead of "What's the best solution?" we asked "Which solution can we implement first to test our basic ideas?"
- Instead of "Is this the right architecture?" we asked "Is this good enough to validate our pedagogical assumptions?"
- Instead of seeking comprehensive understanding before acting, we developed hypotheses and tested them.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

This required accepting several uncomfortable truths:

- Some decisions would need to be revisited – we might build something or implement a technical realization that later needed to be rebuilt differently. This felt like failure initially, but we came to understand it as a normal development process and, more importantly, not only inevitable but also a necessary learning process.
- "Good enough for now" was legitimate – perfect implementation is not required for a prototype that can teach us what we really need.
- We could act on partial information – we did not need to fully understand every implication before making a decision. We needed just enough understanding to make an informed initial decision and be prepared to adjust along the way.
- Mistakes were information, not failures – when a technical decision led to problems, we gained crucial understanding of why that approach did not work, which informed better decisions going forward.

The Azure latency problem exemplifies this evolution. Initially, we felt defeated: We had spent weeks implementing voice interaction only to discover it was too slow to simulate interactions. But gradually we recognized this as valuable information: We now understand that cloud-based voice processing introduces delays, which means we need to explore browser-based alternatives. This knowledge, taught by hands-on experience and exploration, fundamentally shaped our subsequent decisions. We learned to think back and forth, question ourselves constructively ("Are we missing something important?"), and ultimately recognize that sometimes we have to try something to find out. The learning process is not optional but an intrinsic and crucial part of the work, and a route of retaining our agency.

3.3 Barrier 3: Iterative Learning Barriers – Navigating Steep Learning Curves and Cascading Complexity

The third type of barrier emerged from the recursive nature of technical development: Solving one problem reliably surfaces new problems requiring new learning. Just when we felt we had achieved some solid understanding, implementation would reveal new gaps in our knowledge, often with sharp learning curves that felt overwhelming. This iterative learning process has a particular emotional texture: Each time we thought we were "almost done," we would discover another layer of complexity requiring substantial new learning. The finish line kept receding.

3.3.1 When One Solution Creates New Learning Requirements

Consider what happened when we needed to deploy our code to make it accessible online. We had been developing locally using VS Code (Microsoft's open-source

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

development environment), testing in our browsers. This worked fine initially. Then we learned we needed GitHub – a platform for storing and managing code. Okay, we created an account. But then GitHub required learning Git commands and concepts: how to "commit" changes, how to "push" code to the repository. These seemed straightforward in tutorials until we actually tried using them and encountered "merge conflicts" when our simultaneous edits created incompatible versions. Resolving these conflicts required understanding how GitHub tracks changes and how to manually reconcile differences. Then we needed to actually deploy the code from GitHub to create a functioning website. This required learning about deployment services, build processes, and configurations we had not anticipated. Each step revealed new concepts we needed to understand. None were individually overwhelming, but their cascade was emotionally challenging.

3.3.2 The Sharp Learning Curve of Security

Another steep learning curve emerged around security, something we had barely considered initially but which became impossible to ignore. Our first harsh lesson: We would put our Azure access key directly in our code, but within days of deploying, Azure warned us the key had been exposed publicly and disabled it. Anyone could have viewed our code, copied the key, and used our Azure account (and our money) for their purposes. This triggered urgent learning about web security: why front-end code is inherently public, how to protect sensitive information, and what security measures were necessary versus optional. These were not concepts that we could defer learning; they required immediate understanding and action.

3.3.3 The Emotional Weight of Learning Cascades and Coping Strategies

What made the iterative learning barriers particularly challenging was their unpredictability. When learning a language, you know roughly what territory to cover: vocabulary, grammar, pronunciation, genres, registers. The scope feels bounded. But in technical development, you cannot see the full scope of required learning upfront. Each implementation decision surfaced new learning requirements that we could not have anticipated. This created several psychological challenges:

- Imposter syndrome intensification – just when we would feel competent in one area, a new area of ignorance would reveal itself. This fed the "I'm not a tech person" narrative; if we were technical experts, would we not have known about these things from the start? However, in fact, it is not uncommon even for tech experts to learn along the way.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

- Planning frustration – we couldn't accurately estimate how long anything would take because we couldn't foresee what additional learning each task would require.
- Energy depletion – the iterative nature meant sustaining learning efforts over months, with new concepts constantly emerging. This was intellectually exhausting in ways that felt different from our usual teaching and research work.

Reflecting on our journey, we found several strategies worked well to manage cascading complexity:

- Just-in-time learning – rather than trying to learn everything upfront, we learned specific things as we needed them. When we encountered merge conflicts, we learned about conflict resolution, but not yet all of Git's advanced features. This kept learning focused and purposeful.
- Celebrating micro-successes – when implementation surfaced a new learning requirement, we reframed this as progress rather than setback. "We now know we need to learn about this" represents an advancement in understanding.
- Time-boxing learning – we gave ourselves permission to learn imperfectly. "I'll spend two hours understanding the basics" felt more sustainable than "I'll master this completely."
- Accepting ongoing learning as normal – perhaps most importantly, we stopped expecting to reach a state where learning would stop. Experienced developers, we realized, also constantly learn new things. The difference was not that they knew everything but that they were comfortable with ongoing learning as part of the process.

The iterative learning barrier did not disappear, but it became less emotionally destabilizing. We learned that the appearance of new learning needs is a sign of progress, not incompetence. Each layer of complexity we navigated built confidence for the next.

3.4 Synthesis: How the Three Barriers Interact

These three barriers – conceptual, decision-making, and iterative learning – do not occur sequentially but are intertwined throughout the development journey. Conceptual barriers make decision-making harder (how can you choose between approaches you do not fully understand?). Decision-making creates iterative learning needs (implementing a decision reveals usually new concepts to master). Iterative learning surfaces conceptual gaps (trying to use GitHub revealed we did not understand "version control" as a concept).

Understanding these barriers as distinct but interrelated helped us recognize what kind of difficulty we were facing at any moment. Feeling overwhelmed by technical

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

terminology? That's a conceptual barrier and we have strategies for that. Paralyzed by choosing between two implementation approaches? That's a decision-making barrier that we can make a best pick for now and adjust if needed. Frustrated that "one more thing" kept emerging? That's an iterative learning barrier and we can time-box our learning and accept good-enough understanding for now.

Most importantly, recognizing these as navigable barriers, as challenges that could be worked through with strategies and persistence, rather than evidence of fundamental unsuitability for technical work, was itself a crucial conceptual shift.

Key Technical Concepts: A Quick Reference

This box summarizes the most important technical concepts discussed in this section. You do not need to memorize these – treat this as a reference that you can consult when you encounter these terms in your own development work.

Web Development Concepts

- **Front-end** – the webpage interface where users interact with your system (like our nursing simulator interface at nursing3.netlify.app)
- **Back-end** – the behind-the-scenes processing and logic (like a restaurant kitchen versus dining room)
- **Hosting** – where your website's files are physically stored on a server computer, usually in a data center (requires reorganizing the everyday meaning of "hosting guests")
- **Deployment** – the process of making your application accessible on the internet through services like Netlify
- **Client-side vs. server-side** – whether computation happens on the user's device or on a remote server – critical for understanding security constraints and performance
- **HTTPS** – secure web connections required for certain operations like accessing device microphones

Programming Concepts

- **Variables** – containers that hold information, but only one value at a time (unlike physical boxes that can hold multiple items)
- **Data types** – categories of information (text versus numbers) that look identical to humans but are fundamentally different to computers – the source of our '5' versus 5 confusion
- **Asynchronous operations** – tasks that can happen simultaneously or out of sequence, requiring explicit signals for turn-taking (like our three-second pause or button-click solution for voice interactions)

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

(cont.)

- **Functions** – reusable blocks of code that perform specific tasks
- **Debugging** – the process of finding and fixing problems in code – often revealed invisible distinctions like text-versus-number issues

AI Model Concepts

- **Context window and memory** – AI models do not inherently remember previous interactions; you must explicitly include conversation history with each new prompt (explaining why our patient "forgot" information)
- **Tokens** – the units AI models use to process text, roughly three to four characters each (not exactly words, which affects how quickly multilingual text consumes the context window)
- **Probability-based generation** – AI models predict likely responses based on statistical patterns rather than following fixed rules – the source of both naturalness and unpredictability
- **System prompts** – background instructions that shape the AI model's overall behavior (like our attempts to make empathetic responses more likely)
- **Model size and capability trade-offs** – larger models offer better understanding but require more computational resources – a trade-off landscape rather than simple "better versus worse"
- **Hallucinations** – when AI models generate plausible-sounding but fabricated information (a concern for medical accuracy in our simulations)

Development Tools and Practices

- **API (Application Programming Interface)** – a way to access external services in your application (like using Azure's voice recognition), but which can introduce latency through multiple round-trip communications
- **GitHub/Git** – tools for tracking changes to code and collaborating – required learning about commits, merges, and conflicts
- **Environment variables** – a secure way to store sensitive information (like API keys) separate from publicly visible code; we learned this after our Azure key was exposed
- **Version control** – tracking changes so you can revert to earlier versions when experiments do not work out
- **Build processes** – converting development code into optimized files ready for deployment

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

4 Building through Learning and Practice: The Journey from Conception to Working System

When we first conceived of building an ADS for nurse–patient communication training and assessment that would incorporate our theoretical and methodological perspectives, and when we decided to undertake this development work ourselves, we faced one of the most challenging questions: Where do we actually begin? Even though we already knew that AI could probably assist us in achieving this goal, we were not certain how to begin given our very limited knowledge of coding, programming, and AI training.

We want to make sure that our account of this journey is read not only as about the technicalities of building a system but also, and more importantly, as about teacher agency. Through engaging in dialogue with AI as their coding assistants, teachers will be reading and modifying code rather than merely using finished software. Teachers will make technical decisions grounded in pedagogical purposes, rather than implementing another educational initiative. This section chronicles that transformation not as a linear progression of technical mastery but as a messy, iterative process of discovery, where our growing understanding reshaped what we were trying to build and, ultimately, transformed us from doing implementation to innovation. We are not claiming that we have discovered and implemented the optimal technical or system structure that can serve our purposes, but we believe that reaching this point makes a significant difference, not only in the fact that we now have a working system to test and improve further but also in our growing knowledge and skills as teacher-developers, which we believe will benefit our future projects immensely.

The development trajectory we describe here will likely not perfectly match your own journey when you build educational AI because there will be considerable differences in teaching contexts, along with students' needs and the difficulty level of technical challenges. But the underlying pattern, including the rhythm of reaching the limits of what AI can do for us, being forced to understand enough to modify code ourselves, and discovering new possibilities that reshaped our vision, may feel remarkably familiar. This is the lived experience of teacher-led development, where we grow into capabilities we did not know we possessed.

4.1 AI Created a System in One Click

Our first version emerged from enthusiastic naivety. We knew almost nothing about web development, but we knew that Claude and ChatGPT could write code. So we did what felt natural: We described what we wanted in plain language. "Create a web-based system where nursing students can speak to a simulated Cantonese-speaking patient. The system should recognize student speech, determine

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

appropriate patient responses, and speak those responses back." Within minutes, AI generated a complete HTML file, claiming it contained the functions we required, including interface, speech recognition, conversation logic, and audio synthesis, all bundled together. It felt almost magical when we opened the file in a browser, clicked "Start," spoke to it in Cantonese, and heard the simulated patient respond. The voice sounded robotic, but it worked! The technical mystique we discussed in Section 1 seemed suddenly dissolved. If AI can generate working code from plain descriptions, perhaps the barriers to teacher-led development are lower than we imagined?

This initial success was intoxicating but misleading. Yes, AI created a functioning prototype. But along the robotic-sounding voice made it more like a toy than pedagogically useful. In addition, AI made countless decisions that we did not understand, such as which service to use for speech recognition, how to structure the conversation logic and flow, when to trigger responses, and how to handle errors.

A Moment to Reconnect 4

You're about to see actual code. Your first instinct might be "this confirms I can't do this – it looks like gibberish." But remember: The first time you saw a musical score, or a chemical equation, or academic citation formatting, it probably looked like gibberish too. The difference is that you weren't expected to read music or chemistry notation or APA style without guidance. Code is the same; it is a notation system with rules and patterns. You don't need to understand every character to grasp what a section of code accomplishes. Think of it like reading a recipe in a language you're learning: Even if you don't know every word, you can often figure out "this section is about mixing ingredients" or "this part involves baking." That's the level of reading we're aiming for: structural comprehension, not fluency.

The code (see Figure 2) looked like an alien language:

```
```javascript
const recognition = new webkitSpeechRecognition();
recognition.continuous = true;
recognition.onresult = function(event) {
    const transcript = event.results[event.results.length - 1][0].transcript;
    processStudentInput(transcript);
};
```
```

Figure 2 Code sample 1.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

We can now read this code after approximately six months of working on and further developing the system. We can now understand that it creates a speech recognition object (const recognition), tells it to listen continuously through the microphone (recognition.continuous = true), and processes recognized speech through a function called "processStudentInput." But at the time, it was opaque. We had a working system that we did not understand, and therefore could not modify or maintain ourselves. At this stage, if we wanted to make any modifications or changes, we had to ask AI to write the complete code again and again.

4.1.1 When AI Assistance Hits Its Limits

The first real test came when we wanted to change the voice system and replace the robotic voice. We had learned that Microsoft Azure provides a very natural-sounding Cantonese speech engine, and the pricing fell within our budget. We set up our team's Azure service, obtained the API key, and asked AI to replace the voice engine in the system. Then the problems arose.

This was not an easy process. When incorporating the Azure speech engine, AI constantly reached its length limit (around 700 lines of code), breaking the code it intended to write into two chunks of messages. Because of this limit, the ending of the first chunk typically terminated halfway through a complete line of code. Since even a tiny imperfection in the code, such as a missing semicolon or closing bracket, would cause the application to stop working, we were soon forced to learn "just enough" knowledge about the structure of the HTML code so that we could restitch the two chunks together. Moreover, we found it debilitating to keep regenerating the entire file every time we wanted just a small change, as we had to work very hard to restitch the chunks together. Readers can see how many commits we made on our GitHub back then. We identified the need to understand the file's structure well enough to modify it ourselves to make small changes. We also needed to give AI more precise instructions about which specific sections to modify, and to know where to replace the code with the new chunks AI wrote for us. This realization forced our first major growth step: We needed to learn how to read code.

4.1.2 Learning to See Structure through Working with AI

We asked AI: "Explain the structure of this HTML file. Which parts do what?" This meta-question, asking about code structure rather than asking for code modifications, proved to be transformative. AI explained:

- The `<style>` section contains visual appearance rules (CSS).
- The `<body>` section contains the page structure users see (HTML).
- The `<script>` section contains the behavior and logic (JavaScript). Within the script, there are variable declarations, function definitions, and event listeners.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

This explanation gave us a mental map, although specific terminology such as "event listeners" remained opaque. The file was no longer an undifferentiated mass of mysterious text as it had distinct regions with different purposes. Although we still did not understand what CSS or JavaScript actually means, we developed a relatively vague idea that the CSS seemed to be about how the system would appear in a browser. We could see elements related to color and numbers indicating the width and height of buttons. In the JavaScript sections, we encountered elements that seemed to be performing the logic of the system. We found the annotations written by AI to be useful; they provided clues about what each chunk of code would do, and the code then became at least partially readable to us.

For example (see Figure 3), the annotation (the sentence following "//") tells us that this chunk will perform some initialization of the page. Guessing from sentences like "Checking authentication . . ." we could assume that it represents a process of checking whether the user has logged into the system, potentially preventing users from accessing the system when they are not logged in. Because of this insight, we develop a new workflow: Instead of asking AI to regenerate everything, we started copying a specific function and asking, "Explain what this function does line by line." For example, in relation to the code in Figure 4, AI

```
// INITIALIZE PAGE
window.addEventListener('DOMContentLoaded', async () => {
console.log('🔐 Checking authentication...');
const loggedIn = await authFunctions.isLoggedIn();
if (!loggedIn) {
        console.log('⚠ User not authenticated');
        alert('請先登入才能進行測驗！ \nPlease login first to access the pretest!');
        window.location.href = 'login.html';
        return;
}
```

Figure 3 Code sample 2.

```
```javascript
function processStudentInput(transcript) {
        const lowerTranscript = transcript.toLowerCase();
        // Find matching response based on keywords...
}
```
```

Figure 4 Code sample 3.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

responded: "This function takes the student's spoken words (transcript), converts them to lowercase for easier matching, then searches through predefined responses for keywords . . . " Understanding one function at a time makes the code gradually less alien. In addition, we gradually acquired the ability to partially understand the code (although we are still not able to master the "grammar" and begin writing it ourselves independently).

This iterative learning with AI as a tutor rather than merely a code generator changed our relationship to the system. We were no longer only passive recipients of generated code; we became people who could understand, modify, and maintain code to the extent *just enough* for us to fix issues or add new features to our system. Still beginners, certainly, but beginners who can read and understand some part of the hidden processes rather than merely hoping the magic would work.

4.1.3 What We Actually Learned from Version 1 (AI's One-Click Creation)

Version 1's most important lesson was not about speech recognition or conversation logic, although we began to comprehend these components. It was about the nature of working with AI as our coding and learning assistant: AI excels at generating complete solutions, but if we had struggled with the ability to fix and update selected chunks of AI's existing work, our further development work would have been constrained by AI's current length capability. This means teacher-developers need sufficient understanding to guide AI assistance precisely ("modify this specific function") rather than vaguely ("make it better") and to collaborate with AI to undertake more complex tasks than a single HTML file can contain.

Looking back, this was the technical mystique operating in reverse. We had assumed the hard part was getting *any* code to work – that if AI could generate something functional, we had crossed the main barrier. But functionality without understanding just created a different kind of helplessness. The system worked to some degree, but we could not shape it, could not make it better, could not make it *ours*.

We also discovered that testing reveals limitations that no amount of code reading can anticipate. When we asked our colleagues to help test Version 1, the speech recognition struggled with background noise and typically lagged behind, making the conversation feel very unnatural. When they spoke naturally rather than carefully, keyword matching failed, as the mechanism of word matching does not allow much variation in wording. The patient's responses sounded robotic, which made the whole practice seem unserious. These were

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

not coding errors; they were fundamental limitations of the browser-based approach (where everything happens in the user's web browser) that AI had chosen for us in the one-click creation, without our awareness of this decision or possible alternatives.

Understanding these limitations required another shift in how we worked with AI. Instead of asking "why does this not work?" we began to ask "what are the technical constraints of using browser-based speech recognition?" This led us to prompt AI to explain additional concepts like client-side (the browser-based approach) versus server-side processing, API rate limits, and the trade-offs between different speech technologies. We have learned to have rather technical conversations with AI where we gained conceptual understanding, not merely solutions we did not understand. This additionally gave us intuition for diagnosing future issues. When something went wrong, we developed the intuition for what could possibly have malfunctioned. Most crucially, the one-click version and the learning that occurred alongside implementing and testing taught us that becoming a teacher-developer is not about mastering programming languages. It is more about developing sufficient technical literacy to identify what needs changing, to communicate those changes precisely to AI assistants, and to verify that changes work as intended. You do not need to write code from scratch when AI can generate it. But you do need to understand code structure well enough to guide AI's modifications and catch its errors. This realization led us toward Version 2, where we deliberately simplified to gain control.

4.2 Version 2: Strategic Retreat and Deeper Understanding of Our Options

After the one-click version's keyword-matching struggles and the limitations of the voice engine that generated robotic-sounding patient responses, we made what colleagues initially perceived as an admission of defeat: Version 2 abandoned real-time conversation entirely. Instead of attempting to understand student speech and generate responses through voice engines, we created a library of pre-recorded audio files, each accompanied by descriptive metadata (see Table 1). Based on this metadata, the system would simply select and play one of the pre-recorded patient voice files to the student.

This may sound like a retreat in terms of technical complexity, from having AI generate responses to having AI choose from pre-recorded sound files. However, although it may be a retreat in technology applied, it may well be an advance in pedagogical fitness. We were not yet able to remove the limitation of the open-source lightweight LLMs we were using, but we could control something immediately important: the authenticity of the patient's voice. One of

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Table 1 Metadata of the pre-recordings (a sample)

Dialogue stage	Main emotion	Emotion intensity	Secondary emotion	Language characteristics	Applicable scenarios	Not applicable scenarios	Review status
Initial contact	Nervousness	1- Mild	Nervousness	Mainly Cantonese	Stranger entering	Program already started	Pending annotation
Identity confirmation	Cooperative	2- Moderate	Cooperative	Mixed with English	Identity confirmation needed	Patient already familiar	Annotated
Procedure explanation	Pain	3- Strong	Pain	Mixed with Mandarin	During procedure explanation	No pain state	Pending review
Procedure in progress	Curious	-	Curious	Slower speech rate	Starting operation	No explanation needed	Reviewed
Procedure completion	Confused	-	Confused	Normal speech rate	Operation in progress	Operation completed	Needs revision
-	Grateful	-	Grateful	Faster speech rate	Feeling pain	-	-
-	Reassured	-	Reassured	Hesitation and pauses	Needs comfort	-	-
-	Caring	-	Caring	Rising intonation	Procedure completed	-	-
-	Questioning	-	Questioning	Stable intonation	Expressing gratitude	-	-
-	Seeking help	-	Seeking help	-	Concerned about follow-up	-	-
-	-	-	None	-	-	-	-

our team members who is a fluent Cantonese speaker volunteered to record patient responses with authentic emotion, pacing, and Hong Kong dialect variations. The recordings we have sound genuinely like an elderly lady from Hong Kong in ways that browser speech synthesis can never produce. With the metadata stored in an Excel file that labeled each voice recording, we were able to launch our Version 2, which simultaneously solved the problem of lagging time caused by the duration needed to generate text and synthesize speech. In the process of developing Version 2, we also accidentally learned another useful skill that greatly improved our workflow in collaborating with other team members. As mentioned, we pre-recorded patient voice files and simultaneously needed to label each recording with the metadata in Table 1. It would have been a huge burden adding to the already heavy workload of our usual teaching and research work, so we have learned Python (a programming language widely used in AI development and machine learning) to the extent that we could use a Python program written by AI to help us streamline the process of managing the metadata for each recording. We used the same Python program to help us generate an Excel sheet in which our collaborator could easily choose from the predetermined choices (like the choices in Table 1) built into the Excel file to help us manage the metadata of the recordings. The Python program was also able to help us process the metadata and then create JSON (JavaScript Object Notation), which could store the metadata in our system and make it ready for use. Consequently, the workflow became:

AI creates a Python program
↓
The Python program creates an easy-to-use Excel sheet
↓
The team works in the Excel sheet to add metadata to the patient's voice recording
↓
The Python program parses the Excel sheet and generates a JSON file for the system

In the above workflow, the Excel sheet became an important intermediate communicator between the team members who conducted the engineering work and the team members who did the recording and possessed better knowledge about the information of each recording and its intended purposes.

4.2.1 The Emotional Architecture of Teacher-Led Development

Version 2's pre-recorded approach yielded an unexpected technical benefit: it dramatically reduced latency. The conversation no longer suffered from the

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

awkward pauses required by text generation and voice synthesis. Students could speak, and the simulated patient responded almost immediately, creating a rhythm that feels genuinely conversational. This seemingly simple improvement, which reduced the response time from several seconds to nearly instant, transformed the practice experience from something artificial into something that could sustain the suspension of disbelief necessary for meaningful learning.

But alongside these technical improvements, we discovered something we did not anticipate in any of our planning: an emotional attachment to the system that grew with each problem that was solved. This is difficult to articulate precisely, but each bug we fixed, each feature we successfully added, seemed to weave another thread connecting us to what we were building. It is not merely that we are proud of our work, though we are. It is something deeper, more visceral. The system carries the weight of our learning, our struggle, our persistence through moments of frustration and confusion. This attachment proved to be a double-edged sword. On one the hand, it became a source of profound motivation. Since we invested so much, we were not going to easily abandon the work whenever we encountered new obstacles. Each success, whether solving a stubborn bug or finally grasping a concept that had eluded us, generated a genuine sense of achievement that propelled us forward. We were not merely building a system; we were building our capacity as teacher-developers, and that transformation felt meaningful in ways that transcended the immediate project. The system carries our learning, our struggle, our persistence through confusion. We were not just building a tool, we were becoming different kinds of teachers, ones who could shape technology rather than only use it.

On the other hand, this investment sometimes brought moments of exhaustion and doubt too. During particularly challenging phases, when progress felt glacial and solutions remained elusive, a nagging question still surfaced: Could an expert have accomplished all of this in an afternoon? What is the point of our weeks or months of effort if a professional developer can replicate our work in a fraction of the time? In these moments, the very attachment that sustained us also made us acutely aware of our inefficiency, our circuitous learning path, our amateur status. We came to recognize this emotional turbulence as an essential stage of the teacher-led development process. It is not a weakness to overcome but a natural feature of the landscape. The attachment we feel is not irrational sentimentality; it is the inevitable result of transformative learning. When you struggle to understand something, when you persist through confusion and eventual breakthrough, you do not simply acquire knowledge; you also forge a relationship with what you are creating. The system became a testament to our growth, a tangible record of capabilities we developed. This emotional dimension continued to shape our development work through subsequent versions.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

4.3 Version 3: When Growth Demands Structure and Pedagogical Fit

Version 2 worked well for individual practice sessions, but accumulating use revealed a fundamental limitation: This was still a system that served individual practice in private. It did not yet possess the functionality to record or assess the practice. We tested the system within our research team and with a few other colleagues to gain feedback from them. After consulting nursing educators, we realized that the nursing program wanted to expand beyond a single wound-care scenario. Our collaborators envisioned different patient personas, such as an anxious young mother, a resistant elderly man, a confused patient with dementia, and each requiring different communication approaches. In addition, to ensure that nursing students are equipped with sufficient knowledge to carry out medical procedures such as wound cleaning, our collaborators wished to explore installing pre-assessments before the training conversation. They also wanted usage analytics: Which students practiced most? Which conversation stages caused difficulty? How long did typical sessions last?

After some discussion within our team, we concluded that Version 2's single-file architecture could not accommodate this complexity. We brought our question to AI again and were told that our single file was already very complex, which could potentially cause issues. We could not simply keep adding features to one HTML file as it was already unwieldy. We needed multiple pages with different purposes: a landing page explaining the system, a registration page for student accounts, pre-test and post-test pages, separate training interfaces for different scenarios. We needed these pages to share information and coordinate their behavior. We then followed AI's suggestion to embark on a new journey of modularizing the system. This was somewhat daunting at first because, as mentioned, we were already familiar with the single-file approach and were required to step into new territory again.

4.3.1 Discovering Modularization through Necessity

This need forced us to understand web application architecture, not from textbooks or online courses, but from practical and pedagogical necessity. We asked AI questions like: "How do professional web applications organize multiple related pages?" and "What is a better way to share data between different pages in our situation?" AI introduced us to new concepts: routing (how the system determines which page to show), state management (how to preserve information as users navigate), and modular design (separating different concerns into manageable pieces). We learned these relatively fast because they were not abstract computer science concepts; instead, they were solutions to our specific, tangible problem of expansion at that time.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

We learned that instead of one massive HTML file, we could create separate files for different pages:

- `version3_home.html` – Landing page with system introduction
- `version3_register.html` – Student account creation
- `version3_pretest.html` – Initial assessment
- `version3_training.html` – Main simulation interface
- `version3_result.html` – Display the results of the assessment and training

Each file focuses on its specific purpose without becoming overwhelmingly complex. But this created a new challenge: How do these separate pages share information? When a student completes registration, how does the pre-test page know their identity? When they finish practice sessions, how does the post-test page retrieve their performance history?

4.3.2 Encountering the Database Concept

We were faced with the question of data storage and management. In the one-click version, AI had made the decision for us without our knowledge: storing data in the browser's local storage (a way to save data on the user's device). We then did not have access to these data as they were not stored on our server. We soon discovered that this is not what we wanted, as local storage revealed its limitations quickly. If a student practices on one computer, then switches to another, their progress would disappear because local storage only stores data on one device. If an instructor wants to review student performance, they would have to examine each student's individual computer. If we want aggregate analytics across the entire nursing cohort, we would have to manually collect data from dozens of machines, and this is, of course, not pedagogically feasible. We needed server-based storage, a database that lives online where any computer can access it, where instructors can review all student data, and where analytics can compute across the entire user base. This realization led us to learn about databases, not from a "Databases 101" course but from asking very specific questions for our project: "How can our system have a function of accounts and then store and retrieve user data?" "How can we have a mechanism in which the student's performance in the pre-test and the training session can be recorded in the back-end?"

4.3.3 Learning Database Concepts through Supabase

AI recommended Supabase, a service providing database capabilities accessible through JavaScript, designed specifically for situations like ours where non-professional developers need database functionality without managing

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

servers. We asked AI to explain the overall concept of databases conceptually before showing us code: "Explain how databases organize information differently than simply saving files." We developed a relatively vague idea about databases, and then we embarked on our journey.

Implementing this required learning new concepts: SQL (Structured Query Language, though Supabase abstracted away most of it), table schemas (defining what columns exist and what type of data each holds), relationships (how tables connect), and authentication (verifying student identity to prevent unauthorized access). We in fact only half understand these concepts. However, we followed our AI coding assistant's instructions and used SQL code to create our database, and now the structure appears as shown in Figure 5.

The database schema that emerged represented our growing understanding of the relationships between different types of information: user profiles connected to training sessions, training sessions linked to specific patient

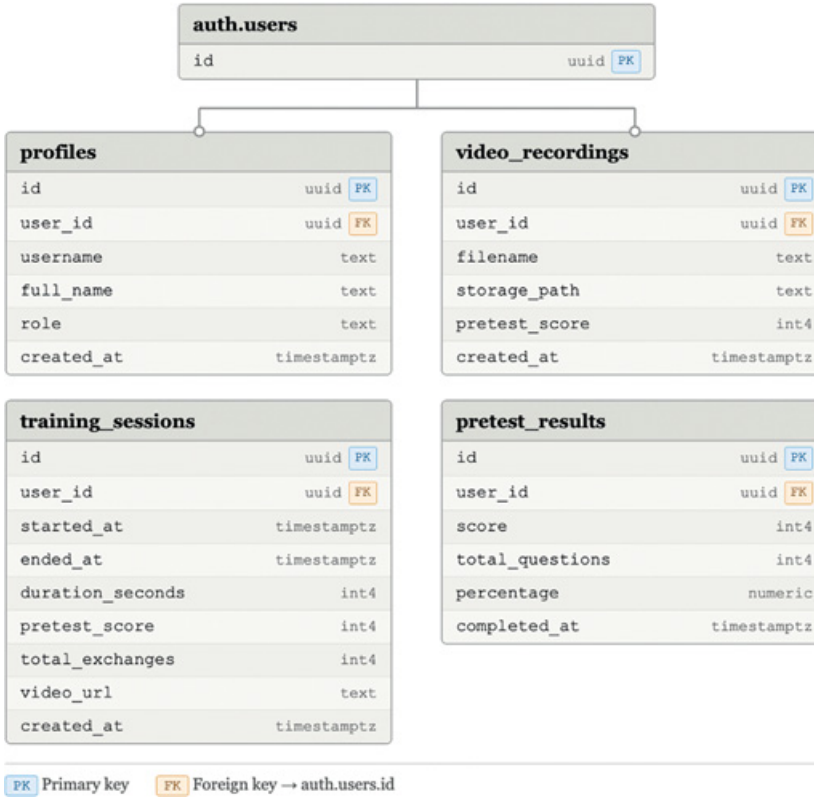


Figure 5 Database schemata.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

profiles, video recordings tied to individual users, and pre-test results tracking student progress over time. We did not design this structure ourselves from scratch; we only described to AI what information we needed to track and why, and it generated the code for us to set up this database architecture. But we learned to read these diagrams, to understand what the connecting lines meant, and to recognize when the structure aligned with our pedagogical goals. This is another instance of developing "just enough" knowledge, not database administration expertise but sufficient literacy to verify that the technical implementation matched our educational intentions. Working with AI on this implementation taught us something crucial about learning technical concepts through AI assistance: The most valuable questions were not "write me database code" but rather "explain the concept of database tables" followed by "show me a simple example" followed by "now how would this work for my specific case?" This progression from concept to example to application proved more educationally sound than jumping straight to complete implementation.

4.3.4 Understanding Modular Architecture

With our partial knowledge of databases, we grew more confident about the task of modularizing the system in Version 3. Our home page needs to redirect students to registration or login pages. The pre-test page needs to know who is taking the test to record the results properly and store it in the database. The training page needs to retrieve the student's practice history. The result page needs to showcase the results by retrieving relevant data from the database. This coordination between modules required learning about URL parameters (passing information through web addresses), session management (keeping track of who is logged in), and API patterns (how different parts of an application request and share data). Again, we learned this by collaborating with AI, alongside our growing knowledge and understanding of the technicalities. AI could explain the concept, show a simple example when necessary, and then help us implement it for our specific needs. This *tutorial-on-demand approach* meant we learned exactly what we needed immediately, when we needed it, in contexts that made sense for our project.

It needs to be pointed out that we did not ask AI to explain everything. In fact, we only asked AI to explain parts of the code when it became necessary for us to understand the code content in order to modify it or to make a decision about whether this mechanism fit our pedagogical purpose. We, in fact, also simply copy, pasted, and tested AI-generated code in our system if it worked without

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

issue. We believe this strategy made the learning-through-doing approach more sustainable, as we were not overwhelmed by the amount of learning required, nor were we merely copying and pasting while missing opportunities to learn at all. We believe there is indeed a fine balance here that worked because we also had the practical goal of launching the system for use.

4.3.5 Version 3's True Achievement

Version 3's real accomplishment was not only the multi-page structure or database integration; it was our transformation into developers who began to understand system architecture. We learned to think about applications not as monolithic code blocks but as coordinated modules, each handling specific responsibilities, communicating through defined channels. This architectural thinking became foundational for Version 4, where we needed to function not only with coordinating multiple pages but multiple external services, from LLMs and voice synthesis APIs to database backends, to make it work. We could not manage that complexity without first understanding modularity through Version 3's more manageable scope.

We cannot emphasize enough in our Version 3 development story that teacher-developers do not need to master every technical concept before starting to build. We learn what we need as we encounter genuine problems. The question shifts from "What do I need to learn about web development?" to "What specific problem am I trying to solve, and what concepts or distinctions do I need to know in order to address that problem?" This need-driven learning, supported by AI assistance and tutoring, makes technical literacy accessible for teachers like us. We are not here to argue that systematic programming education is not a valuable approach, but we would like to point out that our learning-by-doing approach should work better with the everyday workflow and duties of teachers like us. More importantly, this journey itself is an education for teachers, as we discussed in Section 1, and would fall into the discussed Level 3 proficiency.

A Moment to Reconnect 5

While specific AI models and platforms will continue to evolve, the core principles underlying the technical infrastructure, such as API integration, client-server architecture, and database design, will tend to persist. Understanding these fundamentals, especially the techno-social space it creates, enables teachers to adapt to new technologies with confidence and ease.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

4.4 Version 4: Orchestrating Intelligence

Version 4 represents our most ambitious leap: integrating lightweight LLMs that can process student communication and determine contextually appropriate patient responses, combined with professional voice synthesis and cloud-based data storage. But the technical complexity is not what made Version 4 significant. What mattered was how building it again transformed our understanding of what teacher-developers can do. We did not implement Version 4 by suddenly mastering AI systems, database management, and service orchestration. We implemented it by applying everything we had learned about working with AI coding assistants, combined with a growing comfort with technical concepts and a willingness to learn iteratively through failure.

4.4.1 Choosing "Just Enough" AI: Transformer.js over Commercial Solutions

The breakthrough for Version 4 came when we understood a crucial distinction: We did not need the most powerful AI available; instead, we needed an AI solution that is powerful enough just for our pedagogical purposes. Commercial AI services from companies like Anthropic, OpenAI, or Google are undeniably impressive, but they come with significant costs: subscription fees that scale with usage, substantial energy consumption, and environmental footprints that trouble us. Moreover, these powerful models are, in many ways, overpowered for our specific need, which is determining which pre-recorded patient response to play based on student input.

This realization led us to discover Transformer.js, a library that runs smaller AI models directly in the browser. Instead of sending student speech to commercial AI servers through API, waiting for responses, and managing API costs and environmental impact, we could use a lightweight model that ran locally to analyze student input and select the appropriate pre-recorded response. The AI did not need to generate novel responses; it needed to understand intent well enough to choose from our carefully crafted library of authentic patient voices. This approach aligns perfectly with the principle we articulated in earlier sections: Teachers do not need to adopt the most technically sophisticated solution; they need to find the solution that *serves their educational goals sustainably.* Transformer.js and its supporting models are not as powerful as GPT-4 or Claude, but it is powerful enough. It can process Cantonese, English, and Putonghua. It can recognize conversational intent. It can make decisions quickly without server round trips. And crucially, it does all this at a fraction of the cost and environmental impact.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

4.4.2 Understanding Browser-Based AI through Need

Understanding how Transformer.js works conceptually proved challenging initially. We asked our AI coding assistant: "Explain what Transformer.js is and how it differs from cloud-based AI APIs." The explanation that emerged was illuminating: While commercial AI services run massive models on powerful servers in data centers, Transformer.js runs smaller, optimized models directly in the user's browser. The trade-off is capability for efficiency. These supported models are less capable than their cloud-based counterparts but require no server communication, no API fees, and minimal energy consumption. This conceptual understanding helped us grasp several key points. First, browser-based AI greatly reduces latency. Second, it enhances privacy as student speech never leaves their device. Third, it democratizes access because students and our research team do not need accounts with commercial AI providers. Fourth, it aligns with sustainability because the environmental cost is dramatically lower than repeatedly querying large cloud models. Understanding these trade-offs allowed us to ask better implementation questions: "How do I load the Transformer.js supported model efficiently?" rather than merely "Write me code to use Transformer.js." The difference proved crucial as one question builds understanding of what is happening in our system, while the other just obtained code we cannot maintain or explain.

4.4.3 Learning through Model Integration Challenges

Version 4's development process looked nothing like Version 1's. Instead of asking AI to "create a nursing training system," we worked incrementally, integrating one component at a time, testing each integration and understanding how it works before moving to the next one.

First, we integrated Transformer.js for conversation intelligence. We asked our AI assistant how to load and use the model. It explained the pattern: Load the model when the page starts (which takes a few seconds), process student input through the model to understand intent, receive a classification or embedding that helps determine the appropriate response, then select and play the matching pre-recorded audio file. We implemented this step by step, testing with simple inputs first ("Hello" should trigger the greeting response) before attempting complex conversational scenarios.

Each integration taught us new concepts. For Transformer.js integration, we learned about model loading and initialization (preparing the AI before use), embeddings (numerical representations of text that capture meaning), similarity matching (finding which pre-recorded response best matches the student's intent), and async operations (since model processing takes time). The crucial

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

learning happened not only when things worked but also when they failed. The model sometimes took too long to load, and we learned about lazy loading and showing appropriate loading indicators. Intent matching would occasionally select wrong responses, and we learned about confidence thresholds and fallback strategies. Audio playback would sometimes stutter, and we learned about audio buffering and preloading.

Each failure prompted questions to our AI assistant: "Why would the model take so long to load?" or "How can I improve intent matching accuracy?" The AI would explain the underlying concept; we would adjust our implementation and test again. This iterative debugging cycle built understanding that passive acceptance of working code never could.

4.4.4 Discovering Architecture through Orchestration

The most profound learning from Version 4 came from needing to coordinate multiple components. A single student utterance required:

1. Capturing audio from the microphone
2. Sending it to Azure for speech-to-text conversion
3. Retrieving the transcribed text
4. Processing the text through the Transformer.js library to understand intent
5. Matching the understood intent to our response library (done by an open-source LLM)
6. Selecting the appropriate pre-recorded patient audio (done by an open-source LLM)
7. Playing the selected audio file
8. Saving the interaction to Supabase (both video and chat history)
9. Updating the displayed conversation history

Coordinating these steps required understanding not just each individual component but how they fit together, what order operations must occur in, how to handle timing when components take varying amounts of time to respond, and what to do when any step fails.

A Moment to Reconnect 6

This coordination of multiple components might sound overwhelming: audio capture, speech-to-text, AI processing, response selection, database storage. But consider what you coordinate every day in your classroom: student attention levels, pacing across different ability levels, curriculum requirements, assessment schedules, behavioral management, individual learning plans, emotional support needs. You're constantly orchestrating multiple complex systems that interact in real time, making split-second

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

(cont.)

decisions about which needs take priority. The technical orchestration we're describing isn't more complex than what you do – it's just differently complex. Your capacity to manage classroom complexity doesn't directly transfer to technical complexity, but the same underlying skill – holding multiple moving parts in your mind while keeping sight of the larger goal – absolutely transfers.

We learned about this coordination not from studying "distributed systems architecture" but from the practical problem of making multiple components work together. We would ask our AI assistant questions like: "After getting the speech-to-text result, how do I pass it to Transformer.js?" or "What happens if Azure is slow to respond and how do I prevent the interface from freezing?" These concrete questions led to explanations of concepts like Promises (representing future results), async/await patterns (readable ways to handle asynchronous operations), and error handling (managing failures gracefully). The coordination code became complex, but we understood it well enough as we had built it incrementally, learning each piece before adding the next. When problems arose (they always do), we could identify which component was failing, isolate that part of the code, and troubleshoot it without needing to understand the entire system simultaneously.

4.4.5 Learning from Production Challenges

Deploying Version 4 for actual nursing student use revealed challenges we never encountered in development. Since many students could use the system simultaneously, could our database handle concurrent access properly? Students practice in noisy environments – how might background noise affect speech recognition? How will students with varying internet connection speeds and device capabilities affect the Transformer.js model's performance, especially on older computers? These are not problems we could anticipate or avoid. They emerged from real use. But crucially, we developed enough technical literacy to troubleshoot them. When database conflicts occurred, we ask our AI assistant: "How do I handle concurrent database writes?" When the model loaded slowly on some devices, we could ask: "How can I optimize Transformer.js loading for lower-end devices?" We learned how to learn: identifying problems, formulating technical questions, implementing solutions, testing results. We also discovered the importance of concepts like caching (storing the loaded model so it does not need to reload), progressive enhancement (ensuring the system works even if

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

some features fail), and performance monitoring (tracking system responsiveness). We did not learn these proactively; we encountered problems that required them, then asked our AI assistant: "Why are some students experiencing slow model initialization?" The assistant would explain solutions as part of the answer; we would implement the suggestion, understanding it through use.

4.5 The Sustainability Learning Moment

One crucial insight came from comparing our approach to what we may have built with commercial AI APIs. A colleague asked: "How much would this cost and what would be the environmental impact if you used ChatGPT or Claude instead of Transformer.js?" We did not calculate this precisely, but the comparison is stark. With commercial APIs, every student utterance would trigger a cloud AI query – dozens or hundreds per practice session, multiplied by all students in a nursing cohort, repeated throughout the semester. The monetary cost would accumulate quickly, potentially making the system financially unsustainable for educational institutions. The environmental cost, including the energy consumed by data center servers processing millions of queries, would be substantial. With Transformer.js, the model loads once per session, then processes all interactions locally with much less energy cost beyond what the device already consumes. No ongoing fees. No server infrastructure. No data center energy consumption. The trade-off is accepting that our intent matching is not as sophisticated as GPT-4 or Claude, but it is sophisticated enough. This crystallized an insight that became central to our understanding of teacher-lead development: The goal is not to build the most technically impressive system but to build a system that achieves educational goals sustainably. Sometimes "just enough" AI, running locally, serves students better than "state-of-the-art" AI that brings unsustainable costs and environmental impacts.

4.6 The Pattern of Teacher-Developer Growth

Our development trajectory followed a pattern that may be instructive for other teachers considering similar journeys:

Phase 1: Enthusiastic Dependency (Version 1)
We relied entirely on AI to generate code, treating it as magic that we did not need to understand. This worked initially but left us unable to modify or maintain what we built. The limits of this approach emerged quickly, as when we needed changes, we had to regenerate everything, losing customizations and creating new problems.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Phase 2: Forced Literacy and Awakening (Version 1 continued)
When regeneration became unsustainable, we were forced to learn code structure, not through systematic study but through necessity. We learned to ask AI explanatory questions rather than merely requesting code solutions. We developed the ability to read code, understand its organization, and identify what needed change. This shift from "give me code" to "help me understand this code" proved transformational. More importantly, we began to awaken from our total dependency on AI, realizing that understanding the code structure is essential not just for maintenance but for reclaiming our agency in the development process.

Phase 3: Strategic Negotiation and Regaining Agency (Version 2)
This phase marked our emergence as genuine partners in dialogue with AI rather than passive recipients of its output. We learned to navigate complexity by discussing options with AI, negotiating technical trade-offs, and making informed decisions about what we actually needed. We were no longer asking "Can you make this work?" but rather "What are the trade-offs between these approaches?" and "Which solution best serves our pedagogical needs?" This phase represents a fundamental shift in agency: We had moved from dependency to partnership, from accepting AI's decisions to actively shaping them through informed negotiation.

Phase 4: Architectural Agency and Informed Decision-Making (Version 3)
Growth in actual use forced us to think beyond single features toward system architecture, but more significantly, we now possessed the agency to make these architectural decisions ourselves. We learned about modularity, data persistence, and coordinated workflows, not merely as technical concepts but as choices we were able to evaluate and select. We were no longer merely implementing what AI suggested as solutions; we were directing the path and direction of the development process, using our own knowledge, evaluating trade-offs, and choosing paths forward. This represents genuine teacher agency in technical development, as we owned the vision and the decisions, while AI served as our knowledgeable consultant.

Phase 5: Autonomous Orchestration and Sustained Agency (Version 4)
Our most ambitious version required coordinating multiple external services and components, and by this phase, we exercised full agency over the development process. We made agentic choices, such as selecting Transformer.js over commercial AI APIs, based on our evaluation of pedagogical needs, sustainability concerns, and cost considerations. We negotiated with AI not as

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

supplicants hoping for solutions but as informed developers who understand enough to assess proposals critically, identify potential issues, and insist on alternatives when necessary. We became comfortable with just enough knowledge, knowing enough to be effective without knowing everything. Our agency no longer depended on comprehensive technical mastery but on our capacity to learn, question, decide, and direct.

5 The Sustainability Imperative: Lightweight AI for Education

Having chronicled our development journey from Version 1 through Version 4, we now turn to an equally fundamental dimension of teacher-led AI development: sustainability. While we have woven environmental considerations throughout our narrative of technical growth, we believe the sustainability imperative deserves focused attention. The choices teachers make about how to build educational AI systems carry environmental consequences that extend far beyond individual classrooms, and these choices cannot be adequately addressed as mere footnotes to technical decisions.

The environmental footprint of AI development is substantial and growing. Recent studies have revealed costs that remain largely invisible to end users. Each query to ChatGPT consumes approximately 0.3–0.4 watt-hours of electricity and emits around 0.15 grams of CO2, which is four to five times more than a standard Google search (Luccioni et al., 2024). Training GPT-3, which has 175 billion parameters, consumed 1,287 megawatt hours of electricity and generated 552 tons of carbon dioxide equivalent, the equivalent of 123 gasoline-powered passenger vehicles driven for one year (Patterson et al., 2021). Even the simple act of conversing with AI systems carries hidden costs: A typical exchange of 20–50 questions consumes enough water for data center cooling to fill a 500 ml bottle (Li et al., 2023). These figures, while individually modest, accumulate dramatically when multiplied across millions of users and the countless development conversations that teacher-developers conduct while building their systems.

Yet this creates an uncomfortable paradox. Without AI coding assistants, teachers like us cannot realistically build the educational systems we envision. But relying heavily on AI assistance to generate code and brainstorm solutions consumes substantial energy, potentially undermining the very sustainability principles we wish to uphold. This paradox demands that we ask a different first question: Rather than immediately turning to AI to build everything from scratch, what existing resources may we draw upon that would dramatically reduce our carbon footprint? What already exists that we could adapt rather than create a new?

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

5.1 The Open-Source Community: A Foundation for Sustainable Development

The open-source community represents one of the most significant resources for reducing the carbon footprint of educational AI development. Platforms like GitHub and Hugging Face have become vast repositories of freely available code, models, and educational resources that teachers can adapt rather than building from scratch. This ecosystem fundamentally changes the sustainability calculus of teacher-led development.

Hugging Face, often referred to as "the GitHub of machine learning," hosts a great number of AI models from leading providers through a unified platform. More importantly, it provides many useful pre-converted models optimized for browser-based inference, educational toolkits translated into different languages, and countless examples of educational applications that teachers can study, adapt, and deploy. We have found this platform to be very useful for us, as it makes powerful AI accessible to educators, researchers, and developers regardless of their institutional resources or technical background.

Consider a few concrete examples of how existing open-source resources can dramatically reduce both development effort needed and environmental impact. A teacher wanting to build a language-learning application might discover an existing speech recognition model on Hugging Face that was trained on multilingual data. Rather than training a new model from scratch, the teacher can download this pre-trained model and adapt it to their specific pedagogical needs. Perhaps they need to change the user interface to align with their school's branding, adjust the feedback mechanism to be more encouraging for younger learners, or modify the assessment rubric to match curriculum requirements. These modifications require minimal AI assistance compared to building the entire system from scratch.

Similarly, GitHub hosts countless educational AI applications with open-source licenses. A teacher may find a basic chatbot tutorial application and realize it contains all the functionality they need for their classroom. With AI as a coding assistant to help make targeted modifications – such as changing color schemes, adding a database to track student progress, or adjusting conversation flows to match specific learning objectives – the teacher can create a pedagogically appropriate tool while consuming only a fraction of the energy that building from scratch would require. This approach aligns perfectly with the learning-through-practice framework we have advocated throughout this Element. Teachers do not need to understand every line of code in an open-source project to adapt it successfully. They need sufficient literacy to identify which components handle which functions, to articulate what modifications

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

would serve their pedagogical goals, and to work with AI assistants to implement those changes. The open-source community effectively provides the scaffolding that makes teacher-led development both feasible and sustainable.

5.2 The Importance of Continuous Learning with AI Coding Assistants

The relationship between technical learning and sustainability may not seem obvious at first, but it proved crucial in our development journey. The more we learned about code structure, programming concepts, and system architecture, the less we depended on AI to generate large amounts of code, and consequently, the lower our environmental footprint became. This connection between learning and sustainability represents one of the most compelling arguments for teachers investing effort in developing technical literacy.

Consider our early development phase with Version 1, when we treated AI as a magical code generator. Each time we wanted to modify something, we would ask AI to regenerate the entire HTML file, often 700 lines or more. Because AI frequently hits length limits, breaking code into incomplete chunks, we asked for multiple regenerations before achieving working code. Each regeneration consumes computational resources, including energy to run the AI model, cooling water for data center infrastructure, and electricity for data transmission. Multiplied across dozens or hundreds of modification cycles, this approach creates a substantial carbon footprint. As our technical literacy grew, our interaction pattern with AI fundamentally changed. By Version 3, when we understood code structure well enough to identify specific functions or sections needing modification, we asked AI to generate only those targeted pieces, such as 20–30 lines instead of 700. We then copied those lines into the appropriate location ourselves, tested the modification, and iterated if necessary. This reduced the computational burden on AI systems by an order of magnitude. Moreover, because we could read the generated code and identify errors ourselves, we also needed fewer iteration cycles, further reducing energy consumption. The environmental benefit compounds when we consider maintenance and updates. A system built without understanding requires AI regeneration for even trivial changes. But a system whose structure we comprehend can be maintained with minimal AI assistance. For example, when our nursing simulation system needed a small adjustment to the user interface, changing button text from "Start" to "Begin Practice," we made this change ourselves in seconds without AI.

Beyond the direct environmental benefits, this learning process connects deeply to the teacher identity and agency themes we have explored throughout this Element. As we developed technical literacy, we experienced the same kind

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

of professional growth that occurred when we first learned new pedagogical approaches. The difference is that this growth simultaneously enhanced our capability and reduced our environmental impact. Each concept mastered represents both professional development and environmental responsibility. This dual benefit creates a virtuous cycle that reinforces sustainable practices. When teachers recognize that learning to modify code themselves not only increases their autonomy but also reduces energy consumption, the motivation to develop technical skills strengthens. When they experience the satisfaction of making changes quickly and efficiently without depending on extensive AI generation, they develop appreciation for the elegance of targeted modifications over wholesale regeneration. The environmental imperative thus becomes woven into the learning journey itself, rather than being an external constraint or sacrifice. This learning-driven approach to sustainability challenges the false dichotomy between technical sophistication and environmental responsibility. One might assume that more advanced development requires more AI assistance and therefore higher energy consumption. Our experience suggests the opposite: As developers become more sophisticated in their understanding, they can work more efficiently with AI, making more strategic requests that generate less code but achieve more precise results.

5.3 The Sustainability Level of Our Whole System

The sustainability considerations we faced extended far beyond our interactions with AI during development. They fundamentally shaped the technical architecture of our ADS nursing training system, particularly in our selection of AI models for processing student conversations. The decisions we made regarding which AI technologies to employ represent one of the most consequential sustainability choices teacher-developers face.

The critical decision point came when we needed to determine how our system would understand and respond to nursing students' speech during practice sessions. We essentially faced two pathways: using commercial API services from providers like OpenAI, Anthropic, or Google, or implementing browser-based AI using Transformer.js with locally processed models. In fact, we anticipated that it would be easier to go with commercial APIs because the performance is reliable and users will feel our system is as smart as OpenAI's ChatGPT or Anthropic's Claude. But in that case, we would have to make significant compromises on our sustainability goals, and we decided to go for the browser-based AI approach. This alternative approach using Transformer.js fundamentally altered this environmental equation. By implementing browser-based AI processing, we moved computation from energy-intensive data centers to the students' own devices. The

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

AI model loads once when the student opens the practice system, consuming a one-time energy cost equivalent to a few minutes of web browsing. After that initial load, all conversation understanding and response selection happens locally in the browser, with no additional cloud computation required beyond the initial model download.

We should acknowledge the trade-offs inherent in this architectural choice. The browser-based AI models we could use are necessarily smaller and less capable than the massive models available through commercial APIs. GPT-4 or Claude 3 would provide more sophisticated language understanding, more nuanced response generation, and better handling of unexpected student inputs. Our Transformer.js implementation, running a model with perhaps 100–300 million parameters rather than hundreds of billions, cannot match that level of sophistication. So, even some of our colleagues who tried our system has commented that it is not able to provide the level of sophistication in responses provided by commercial products in markets. However, we do not need our simulated patient to engage in philosophical discussions or demonstrate creative problem-solving. We need the system to understand nursing students' questions and instructions, recognize their emotional tone, and select appropriate pre-recorded patient responses from our carefully designed library. For these specific tasks, a smaller model proved "powerful enough" while consuming a fraction of the energy and generating a fraction of the emissions.

Beyond the AI processing component, we implemented other sustainability-focused optimizations throughout the system. One significant example involves how we handled video recordings of practice sessions. Initially, we implemented video recording in high resolution with minimal compression, reasoning that higher-quality video would better capture students' facial expressions and body language for later assessment. However, we quickly recognized that these uncompressed video files consumed substantial storage space, approximately 200–300 megabytes for a 10-minute practice session. The sustainability implications of this storage footprint extend beyond the obvious disk space consumption. Data storage in cloud services requires energy for the storage hardware itself, for cooling systems to maintain optimal operating temperatures, and for redundancy systems that maintain multiple copies to prevent data loss. AI helped us do the calculation: Storing 100 students' practice videos at 300 megabytes each creates a 30-gigabyte footprint. Maintained over a semester with multiple practice sessions per student, this could easily reach hundreds of gigabytes, all requiring continuous energy consumption.

We addressed this by progressively adjusting video quality downward until we reached a level that remained "good enough" for our pedagogical purposes – specifically, sufficient resolution to observe students' communication approach

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

and body language during automated assessment but not unnecessarily high. By implementing higher compression and reducing the resolution from 1080p to 720p, we reduced file sizes to approximately 50–60 megabytes per session, which is an 80 percent reduction in storage requirements and corresponding reductions in the energy footprint of storing and processing these files. We continue to monitor whether this quality level adequately serves our assessment purposes and are prepared to adjust it if necessary, but initial testing suggests that 720p captures sufficient detail for evaluating nurse–patient communication competency.

These optimizations – from browser-based AI processing, judicious video compression, to efficient data storage – exemplify how sustainability considerations can become woven into technical decision-making without compromising pedagogical effectiveness. The key principle is matching technical capability to actual pedagogical need, rather than simply selecting the most powerful available solution. We do not need the smartest AI or the highest-resolution video; we need technology that serves our educational goals sustainably.

5.4 The Challenges of Educational AI Development While Maintaining Sustainability Goals

We would be dishonest if we portrayed our commitment to sustainability as unwavering and easy to maintain. Throughout our development journey, we repeatedly faced the temptation to abandon our environmental principles in favor of technical impressiveness or perceived competitive advantage. Understanding these temptations and how we navigated them may prove helpful for other teachers considering similar journeys.

A Moment to Reconnect 7

When we chose "good enough" AI over state-of-the-art models, we were making the same kind of judgment you make constantly as a teacher: "What are the minimum resources I need to achieve my pedagogical goal?" You don't use your most elaborate, time-intensive teaching strategy for every single lesson – you match complexity of approach to importance of outcome, always working within resource constraints. You don't assign the longest-possible essay when a shorter one would achieve the learning objective. These decisions aren't compromise; they're professional judgment. The technical decision to use Transformer.js instead of GPT−4 is the same kind of judgment, just in a different domain. You already possess the decision-making framework, matching resources to goals, accepting "good enough" when it genuinely serves your purpose. The technical application is new but the thinking pattern isn't.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

The most persistent temptation arose from demonstrations and presentations. When preparing to present our system at academic conferences or to colleagues at professional development workshops, we acutely felt the limitations of our browser-based approach. We knew that if we switched to GPT-4 or Claude, the simulated patient responses would be dramatically more sophisticated. The conversation would flow more naturally. The patient would handle unexpected student questions more gracefully. The system would simply seem more impressive, more "cutting edge," more aligned with the current enthusiasm surrounding LLMs. This temptation intensified when we observed colleagues presenting their own AI-enabled educational tools. Many had chosen commercial APIs or just having students use ChatGPT, Claude, DeepSeek, or Qwen. Their systems could engage in complex dialogues, generate novel scenarios dynamically, and showcase the full power of state-of-the-art language models. In comparison, our system with its pre-recorded responses and lightweight browser-based AI felt constrained, perhaps even dated.

We would fantasize about temporarily using commercial APIs "just for the demonstration," reasoning that the environmental cost of a single presentation would be negligible compared to ongoing use. We could also show that the system is very capable. The tension became particularly acute when we considered funding and publication. AI-related research often depends on demonstrating great innovation and technical sophistication. Reviewers of grant proposals and research manuscripts may frequently favor work that pushes technical boundaries, that employs the most advanced available models, that generates results that could not have been achieved with simpler approaches. Our choice of lightweight, sustainable AI might be interpreted as a technical limitation rather than principled commitment. Would our work be taken less seriously? Would funding opportunities diminish? Would publication in high-impact venues become more difficult?

Moreover, we confronted questions about our responsibility to students. If commercial APIs can provide nursing students with more realistic practice experiences, are we not obligated to use them despite the environmental cost? The purpose of educational technology is to enhance learning. If our sustainability commitments compromise educational effectiveness, are we not prioritizing environmental principles over student needs? These questions haunted us during moments of doubt.

5.5 Contextualizing Pedagogical Need in the Global AI Frenzy

What ultimately kept us committed to sustainable development is a shift in how we framed the purpose and value of our work. An old Chinese saying from the

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Song Dynasty poet Xin Qiji – "为赋新词强说愁" (forcing oneself to express sorrow just to compose impressive verses) – captures precisely the temptation we faced. The saying describes someone manufacturing emotions they do not genuinely feel simply to create poetry that appears sophisticated. We realized that adopting commercial AI APIs merely to demonstrate technical impressiveness would be exactly this: adding complexity we do not need to appear cutting-edge, forcing sophistication onto our work for the sake of appearances rather than pedagogical purpose. We came to recognize that our contribution is not primarily about building the most technically impressive system. It is about demonstrating that teachers can build pedagogically effective educational AI while making responsible environmental choices. The value of our work lies precisely in showing that "powerful enough" technology, thoughtfully implemented, can serve educational goals sustainably. This realization connected to broader questions about the direction of educational AI development. The current trajectory, driven largely by commercial companies and their imperatives for competitive advantage, prioritizes ever-larger models, ever-more-sophisticated capabilities, and ever-increasing computational resources. This trajectory is neither inevitable nor necessarily beneficial. If teachers, researchers, and educational institutions simply accept it, if we adopt whatever technology companies offer without questioning the environmental costs, *we cede our agency* in shaping how AI develops and how it serves education. As scholars, teachers, and now AI developers, we recognize that we have a responsibility to articulate an alternative vision. Not everyone must follow this path, but there is space for people like us to have a say in the direction of educational AI development, to practice and demonstrate how sustainability can be emphasized in the development process, and to prove that pedagogical effectiveness and environmental responsibility are not necessarily in conflict.

This stance requires us to reframe success. We stopped measuring our system primarily against the technical capabilities of commercial alternatives. Instead, we asked different questions: Does our system achieve its pedagogical objectives? Do nursing students develop communication competence through practice with our simulated patients? Can the system be maintained and scaled without unsustainable resource consumption? Do we model for students and colleagues that environmental responsibility matters in educational technology choices?

This philosophical stance had practical implications for how we engaged with critiques and comparisons. When colleagues or reviewers noted that commercial API-based systems could generate more sophisticated dialogue, we learned to respond not defensively but with clarity about our design principles. Yes, GPT-4 is more capable than our Transformer.js implementation. But our system meets our pedagogical needs while consuming a fraction of the

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

energy and generating a fraction of the CO2 emissions. That trade-off represents an intentional choice, not a technical limitation we would overcome given sufficient resources.

We also found it important to frame our sustainability commitments not as sacrifice but as responsibility. We are not giving up impressive capabilities for environmental principles; we are taking seriously our role as educators in modeling responsible technology use. Every tool we build teaches a hidden curriculum, not just about nurse–patient communication but about what we value in technology: extraction or stewardship, convenience or care, spectacle or sustainability. Just as we teach students to evaluate sources critically, to question assumptions, and to consider the broader implications of their actions, we model for students and colleagues that technology choices have consequences beyond immediate functionality. The environmental footprint of our tools matters. The energy consumption of our infrastructure matters. These concerns do not compete with educational quality because they constitute part of what educational quality means in an era of climate crisis.

This does not mean teachers should never use commercial AI services. There may be applications where the sophistication of large models genuinely enhances learning in ways that justify environmental costs. But it does mean that educational AI development should involve conscious choice about when such resources are truly necessary, rather than automatic adoption of whatever is newest or most powerful. It means building literacy about the environmental implications of different technical architectures. It means valuing "powerful enough" solutions that achieve pedagogical goals sustainably.

As we reflect on our development journey, we recognize that sustainability shaped not only what we built but who we became as teacher-developers. The constraints forced creativity. The learning reduced dependency. The ethical commitments enriched purpose. We did not build the most technically sophisticated nursing ADS possible. We built a system that works, that serves our students effectively, that we can maintain and improve ourselves, and that embodies our values about education and environmental responsibility. That accomplishment, we believe, represents genuine innovation not in technical sophistication but in demonstrating sustainable pathways for teachers to reclaim agency in educational AI development.

6 Opening a Crack

What we have shared in the preceding sections is not a template to replicate but a story of opening a crack, bringing light into what can feel like the dark side of academic and educational life, overburdened by publish-or-perish pressures,

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

KPI-driven performance metrics, and the erosion of professional meaning. This project is our way of taking the initiative rather than being pushed, of doing work that feels genuinely ours rather than work done merely to satisfy external demands. We are not positioned as experts who have solved the problem of teacher agency and are now offering guidance from above. We are fellow educators and researchers who have been experiencing the same burdens, the same sense that our professional identities are being hollowed out by forces beyond our control. This Element shares what we discovered when we decided to reclaim something for ourselves. The metaphor of the crack deserves brief elaboration. We do not claim to have torn down walls or revolutionized educational AI development. What we hope to do is to find a small opening, a place where light can enter a space that otherwise feels impenetrably dark. The technical mystique we described in earlier sections has convinced us, as it convinces many educators, that AI development is beyond our reach. But cracks, once opened, have a way of expanding. Light enters, and gradually one can see more clearly. Other cracks become visible. What once seemed solid begins to reveal its fractures. This is not triumphant language; it is honest language about modest beginnings that nonetheless matter: *Teachers and educators deserve and should have a say in educational AI development*.

6.1 Positionality, Shared Struggle, and Honest Reckoning

This Element is not a prescription from those who have succeeded to those who should follow. We write as academics who feel the weight of contemporary academic life: the pressure to publish, to secure funding, to meet institutional KPIs. These pressures erode agency for university-based educators, just as commercial AI pressures erode agency for school teachers. The forms differ, but the underlying dynamic is similar: external demands displacing intrinsic purposes, metrics substituting for meaning, performance replacing genuine engagement with our work. Our project began not as a research output to satisfy metrics but as an attempt to do something that felt meaningful on our own terms. We wanted to take the initiative rather than being pushed. The nursing ADS we built emerged from this desire: not primarily from a gap in the research literature or an opportunity for publication, though it became both of those things. It emerged from wanting to reclaim some portion of our professional lives for work that feels genuinely purposeful.

In this sense, we are not here to help teachers regain agency; we are doing it ourselves and sharing that process. The distinction matters. We are not dispensing advice from a position of achieved mastery. We are fellow travelers on a journey that remains incomplete, sharing what we have learned so far while

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

acknowledging how much we have yet to understand. When we describe barriers and how we navigated them, we speak from recent experience of those barriers, not from distant recollection. The frustrations we describe were genuinely frustrating. The doubts were real doubts. The moments of wanting to abandon the project and leave it to "real" developers – those moments happened. This positionality shapes everything that follows. We are not claiming that our pathway is optimal or that our technical solutions are elegant. We are claiming that we, as scholars trained in linguistics and semiotics, not computer science managed to build a working educational AI system that serves our pedagogical purposes. If we can do that, then perhaps the barriers are lower than they appear. Perhaps other educators, with different backgrounds and different purposes, can also find their own pathways. That possibility is what we hope to open.

Intellectual honesty requires that we name the conditions that made our specific pathway possible. We hold postdoctoral and professorial positions at a research university in Hong Kong, a knowledge production hub with substantial research infrastructure. Our principal investigator is willing to grant intellectual freedom and embrace our sustainability initiative rather than directing us toward more conventional research outputs. We have research funding that allowed time for learning and experimentation, time that many educators simply do not have. These conditions enabled this particular form of agency reclamation. We state this not to discourage educators in different circumstances but to resist the false suggestion that our experience is simply transferable. It is not. A primary school teacher in an under-resourced school, already working sixty-hour weeks, cannot realistically undertake the kind of extended learning journey we describe. A university lecturer on a teaching-only contract, measured entirely by student evaluations and course completion rates, faces different constraints than we did. An educator in a context where AI tools are mandated from above, with no room for local innovation, operates under different conditions entirely. What we can say is this: Our conditions were not extraordinary. Many educators have some degree of the resources we had, such as some time for professional development, some institutional tolerance for experimentation, some access to the free tools and communities we utilized. The question is whether those resources, however modest, might be directed toward opening cracks rather than merely responding to external demands. Different educators in different conditions will find different cracks, different ways of bringing light in. Our pathway is one possibility, not the possibility.

The time pressures we face are not unique to us as they are structural features of contemporary education at all levels. School teachers face marking, planning, meetings, pastoral care, administrative duties, and the emotional labor of supporting students through difficulties that extend far beyond the curriculum.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

University educators face teaching loads, publication pressures, grant applications, committee work, and the increasing expectation that they will be available to students at all hours. The feeling that one is always responding to external demands rather than pursuing intrinsic purposes. We emphasize this shared burden because we want to be clear about what we are not suggesting. We are not adding "one more thing" to an already overwhelming load. We are not proposing that teachers should learn to code in their nonexistent spare time. We are not romanticizing technical work as inherently liberating while ignoring the very real constraints that make such work difficult or impossible for many educators. What we are sharing is one way we found to reclaim space within that load: to do work that felt like ours. For us, that meant building an educational AI system. For other educators, it might mean something entirely different: developing new pedagogical approaches, engaging in collective advocacy, pursuing critical evaluation of existing tools, or simply refusing to adopt technologies that do not serve their students. The common thread is not AI development but the decision to take the initiative rather than merely respond and to shape one's relationship with technology rather than being shaped by it.

6.2 Dealing with Self-Doubt and Agency

Throughout this journey, we encountered advice from people who appeared more technically knowledgeable than us. Colleagues suggested different system architectures. Reviewers proposed more advanced technologies. Online commenters recommended alternative approaches. Some readers of this Element may be thinking the same: "I could do this better than the authors." Of course, that may well be true. There are always smarter people, more elegant solutions, more sophisticated technologies. But here is what we learned: Blindly following advice from those who appear more expert does not necessarily make the work easier or better. In fact, it can undermine the very agency you are trying to build. When we defer completely to technical suggestions we do not understand, we lose the thread of our own pedagogical purposes. We end up with code we cannot maintain, decisions we cannot justify. This is not an argument against taking advice; we learned enormously from others throughout this project. It is an argument for taking advice from a position of agency. We needed to know our pedagogical goals clearly enough to evaluate whether a suggestion served them. We needed to understand enough technically to assess whether an alternative approach was genuinely better for our purposes or simply more impressive. And we needed to be willing to say: "That may be a more sophisticated solution, but I do not understand it well enough to maintain it, and my current approach serves my students adequately."

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Agency in learning means: "I take your advice seriously, I will try to understand it, and I will decide whether it serves my pedagogical purposes rather than assuming you must be right because you know more than I do." This requires developing enough literacy to evaluate advice, which is precisely what we described in earlier sections as "just enough" technical knowledge. You do not need to know as much as the advisor to assess whether their advice fits your situation. You need to know your situation well enough, and understand the advice well enough, to make an informed decision about fit. This stance can feel uncomfortable sometimes, even presumptuous. Who are we, with our limited technical knowledge, to question advice from those with vastly more expertise? But the alternative is worse: building systems we do not understand to solve problems we do not have, pursuing sophistication for its own sake, recreating dependency under the guise of collaboration. Teacher-led development requires teachers to lead, which means teachers must ultimately decide what serves their students, even when that means respectfully declining expert suggestions.

6.3 Returning to Yang: Agency, Identity, and Multiple Pathways

Let us return to Yang, the teacher we introduced at the beginning of this Element. In 2023, Yang found himself feeling that his expertise had been devalued, positioned as a consumer of AI technology rather than an agent who could shape it. The question we must now address is not whether Yang should become an AI developer. The question is: How might Yang reclaim professional agency in an AI-influenced landscape? And crucially: What would it mean for Yang to take the initiative rather than being pushed? To understand Yang's predicament theoretically, we draw on Bakhtin's concept of dialogism: Meaning and identity emerge through responsive interaction, not isolated action (Bakhtin, 1981). Yang's agency is not something he possesses individually but something constituted through his relationships, with students, colleagues, technologies, and institutions. His professional identity as a teacher is not a fixed property but an ongoing achievement, continually negotiated through these relationships. When AI enters his classroom uninvited through his students' use of ChatGPT, it disrupts the network of relations through which his identity has been constituted.

We also draw on Latour's actor-network theory: Agency is distributed across human and nonhuman actors (Latour, 2005). Yang's relationship with AI is not simply that of user and tool but a network of mutual shaping. The AI shapes what is possible in Yang's classroom, but Yang's responses also shape how AI functions in that context, what tasks it is assigned, how its outputs are evaluated, what role it plays in student learning. The implication is that reclaiming agency

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

does not require mastering technology but requires repositioning oneself within the network of relations (Law, 1992). Yang need not become a programmer to become an agent. He needs to shift from being shaped by AI to actively shaping his relationship with it. This theoretical framing opens space for multiple pathways to agency. Becoming an educational AI developer, as we have described, is one viable pathway but not the only one. Consider several directions Yang might pursue:

Yang might pursue critical evaluation and informed refusal. He develops enough AI literacy to evaluate tools critically and advocate for rejection of those that do not serve his students. He learns to ask penetrating questions: What data was this system trained on? What values are embedded in its design? What does it optimize for, and at what cost? He becomes the colleague others consult when considering new AI tools, not because he can build them but because he can assess them. This is agency through informed resistance and the capacity to say no from a position of understanding rather than fear.

Alternatively, Yang might pursue pedagogical adaptation and creative repurposing. He does not build AI tools but develops pedagogical strategies that redirect AI use toward his educational goals. He becomes expert at prompt design, at structuring assignments that use AI productively rather than as a shortcut, at teaching students to be critical AI users who can evaluate and improve upon AI-generated content. He transforms the threat of AI-enabled cheating into an opportunity for teaching critical literacy. This is agency through pedagogical innovation, using educator expertise to shape how technology functions in learning contexts.

Or Yang might pursue AI development as we have described throughout this Element. He learns to work with AI coding assistants to build tools serving his specific pedagogical vision, perhaps a system that helps students practice English conversation with culturally appropriate feedback, or a tool that analyzes student writing to identify patterns he can address in instruction. This is agency through technical creation, developing the capacity to build rather than only use educational technology.

Most realistically, Yang would pursue some hybrid path, combining elements of these approaches based on his interests, constraints, and context. Perhaps he develops critical evaluation skills that inform selective development, building some tools while advocating against others. Perhaps he combines pedagogical innovation with collective advocacy, developing classroom strategies while also working to change institutional policies. The pathways are not mutually exclusive; they can reinforce each other.

The point is that there is no single "correct" response to AI's presence in education. Different teachers will find agency through different means, based on

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

their interests, constraints, and contexts. A teacher with programming experience might naturally gravitate toward development. A teacher with strong writing skills might contribute through professional publications and policy advocacy. A teacher with deep pedagogical expertise might focus on instructional strategies that redirect AI use. All of these represent legitimate forms of agency.

What unites these pathways is more important than what distinguishes them. All require moving beyond passive consumption of whatever technology arrives, which is the stance of accepting tools provided by others without questioning whether they serve one's purposes. All require some degree of demystification to understand enough about AI and to make informed choices rather than being paralyzed by the technical mystique. All position teachers as agents who shape their relationship with AI rather than being shaped by it. All involve what Bakhtin might call "answerability" – taking responsibility for one's response to the situation rather than simply reacting to external pressures. And all represent taking the initiative rather than being pushed.

Yang's choice among these pathways should be genuine, not forced by barriers that could be lowered. Those who wish to develop educational AI should find the path navigable, which is why we have written this Element. Those who wish to pursue other forms of agency should not be made to feel inadequate for choosing differently. What matters is that the technical mystique no longer forecloses possibilities, that teachers understand they have options, and that they can pursue whichever options align with their purposes and circumstances.

A Moment to Reconnect 8

If you've reached this point feeling unsure whether AI development is your pathway, that uncertainty is completely legitimate. Not every teacher needs to build technology, just as not every teacher needs to publish research, lead professional development, or take on curriculum leadership. What matters is that you're making an informed choice based on understanding the possibilities, not a frightened retreat based on mystification. Whatever pathway you choose for reclaiming agency, whether it's development, critical evaluation, pedagogical innovation, or collective advocacy, you'll bring the same professional strengths you've always had: deep knowledge of your students, sophisticated understanding of learning, and commitment to education as meaningful work. Those strengths don't become less valuable if you choose not to code; they remain the foundation of your professional identity.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

6.4 Toward Collective Dialogue: Bridging Communities

A problematic separation exists between technology communities and educator communities. Technologists often build educational tools without deep understanding of pedagogical knowledge, such as what makes learning happen, how students actually engage with content, and what teachers need to do their work effectively. Educators often adopt or reject technologies without understanding technical possibilities and constraints, such as what AI can and cannot do, why certain features are difficult to implement, and what trade-offs different architectural choices involve. Each community has developed caricatured views of the other: Technologists sometimes see educators as resistant to innovation and wedded to outdated practices; educators sometimes see technologists as naive about classroom realities and driven by profit rather than learning. This divide serves neither students nor the goal of meaningful educational technology.

Yet people on both sides want connection; they simply do not know how to achieve it. In technology communities, many developers genuinely care about education and are frustrated by building tools that miss the pedagogical mark. They want educator input but do not know how to access it authentically as the focus groups and user testing sessions they conduct often feel superficial, capturing surface preferences rather than deep pedagogical insight. They see educational products driven by metrics and marketability rather than learning outcomes, and they wish it were otherwise. In educator communities, many teachers are exhausted by being told what to use without being consulted, to always having tools imposed from above that do not fit their teaching contexts and create more work rather than less. They have ideas for tools that would actually help but no pathway to realize them, no way to communicate their vision to those who might build it. They want partnership, not product placement.

Genuine dialogic exchange would look different from current patterns. Not technologists building for educators, treating them as dumb end users whose preferences must be satisfied in the so-called easiest possible ways with preferably no learning curves. Not educators consuming from technologists, choosing among options that others have defined and requiring their compliance. But genuine dialogue where both communities learn from each other and create together. Educators would contribute pedagogical vision, like what learning should look like, what students need, what classroom realities must be accommodated. They contribute contextual knowledge such as how their specific students learn, what their institutional constraints are, what cultural factors matter. Technologists would contribute technical possibilities – things like what current AI can do, what emerging capabilities might enable, and what

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

architectural approaches might work. They together would contribute constraint knowledge, such as why certain features are difficult, what trade-offs different choices involve, and what resources different approaches require. Most importantly, shared language would develop through collaboration, not be imposed by either side. Currently, technologists and educators often talk past each other because they use the same words to mean different things or different words to mean the same things. "Learning" means something different to an educator steeped in constructivist theory than to a machine learning engineer. "Model" means something different in pedagogical contexts than in AI contexts. Genuine dialogue would develop shared vocabulary through working together on concrete problems.

Drawing on Bakhtin again: Dialogue requires genuine others, not ventriloquism. When technologists claim to speak for educators based on user research, or when educators claim to understand technology based on using consumer products, both are engaging in a form of ventriloquism, which is putting words in the other's mouth rather than genuinely listening. Authentic dialogue requires each party to encounter the other as genuinely other, with different knowledge, different values, different ways of seeing the problem. This encounter is uncomfortable as it requires acknowledging the limits of one's own perspective, but it is the only path to solutions that genuinely serve both pedagogical and technical goals.

Practical spaces for this dialogue already exist and could be expanded. Open-source communities offer one venue where educators and developers can collaborate on shared projects. Professional development programs can be redesigned to bring both communities together rather than training each in isolation, such as workshops where educators and technologists jointly design educational tools, each contributing their expertise to the process. Research partnerships can be structured as genuine collaboration rather than extraction, not researchers studying teachers as subjects but researchers and teachers working together as co-investigators with different but complementary knowledge.

Policy conversations must also include both technical and pedagogical voices. Bringing educators and technologists together in policy discussions could produce more informed decisions about technology adoption, data privacy, and resource allocation. Teacher-developer communities, like the one we are trying to model in this Element, represent one bridge between these worlds, but they are not the only bridge, and they should not be expected to bear the entire weight of connection.

Schools and universities must create institutional conditions that support dialogue rather than just technology adoption. This means allocating time for

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

teachers to engage with technical communities, not as an add-on to already overwhelming schedules but as a recognized part of professional work. It means providing recognition for collaborative innovation such as valuing the teacher who co-develops an educational tool as much as the teacher who publishes a paper or raises test scores. It means creating funding streams that support partnership rather than just purchasing. And it means developing policy frameworks that require educator voice in AI decisions, mandating teacher input in technology adoption processes rather than allowing administrators to make these choices unilaterally.

The larger stakes of this dialogue are significant. Who shapes educational AI matters, not just for teachers and students but for the future of education itself. Currently, commercial imperatives dominate the field, and understanding the specific challenges of commercial platforms helps clarify what alternatives might offer.

6.5 Commercial Platforms and Teacher Agency: A Critical Examination

Low-threshold commercial AI platforms – what we refer to as commercial educational AI products that aim to make their applications "easier" for teachers – are, we argue, creating new dependencies that ultimately constrain teacher agency. Commercial platforms' appeal is understandable. They require minimal or no technical knowledge, provide customer support, and offer professional legitimacy. For administrators, they provide contracts and accountability. For time-pressed teachers, they promise immediate solutions to pressing problems.

Yet three critical tensions are likely to emerge. First, *erosion of pedagogical specificity*: Platforms optimize for broad markets, not specific contexts. For example, a tool designed for generic learners cannot address Hong Kong students' transition from Cantonese to English-medium instruction with their particular linguistic needs. As teachers begin to rely on these platforms, they have fewer opportunities to develop solutions tailored to their specific pedagogical contexts. At a higher level, when schools adopt identical platforms, teaching converges around platform affordances. Diverse pedagogical approaches, characterized by teachers' varied backgrounds and students' specific needs and personal and cultural backgrounds, give way to platform-determined patterns, which we see as marked by a loss of meaningful pedagogical opportunities. In addition, teacher education emphasizes TPCK (technological pedagogical content knowledge) as integrating content knowledge, pedagogical expertise, and technological understanding. Commercial platforms positioned

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

as "teacher-proof" bypass TPCK development. Teachers aren't asked to reason through *how* tools can be designed differently to serve diverse learning needs; they're encouraged to trust that platform designers have already done that thinking. This shifts teachers from professional decision-makers to implementation technicians, fundamentally altering their professional identity.

In what we propose as teacher-led communities of practice, knowledge flows horizontally: Teachers talk with teachers, sharing practical wisdom and contextual adaptations. Early-career teachers facing institutional precarity can learn from mid-career teachers who have navigated similar challenges. This reciprocal exchange contrasts with university-led professional development positioning researchers as knowledge producers and teachers as recipients. Crucially, teacher-led COPs can develop an alternative infrastructure. When teachers collectively build and share tools and tips, they create ecosystems parallel to commercial offerings: for example, open-source educational AI, collaborative documentation of what works and what doesn't. Technical knowledge distributes across communities; one teacher's programming strength complements another's prompt engineering expertise and another's pedagogical depth.

This is not an argument against all commercial tools. Pragmatically, teachers must work within existing systems. The goal is strategic engagement informed by critical awareness. Teachers understanding AI principles while using commercial platforms can evaluate whether platforms truly serve their pedagogical goals, identify when custom-built tools will provide greater long-term value despite requiring added effort, and advocate in institutional technology decisions from an informed position.

Understanding these dynamics prepares us to examine the specific boundary conditions: the enabling and constraining factors that shape whether and how teachers can pursue alternative pathways.

6.6 Boundary Conditions: What Enables and Constrains Teacher-Led AI Development

The pathway to teacher-led AI development is shaped by specific enabling and constraining boundary conditions that operate at multiple levels (see Table 2).

Consider a mid-career English teacher in a Hong Kong secondary school wanting to develop an AI chatbot for grammar practice. Enabling conditions might include: (1) availability of free AI coding assistants like Claude or ChatGPT; (2) school leadership that values teacher innovation; (3) participation in a teacher learning community that shares coding resources; and (4) strong TPCK from years of teaching experience (Mishra & Koehler, 2006). On the other hand, constraining conditions might include: (1) lack of protected time;

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Table 2 Boundary conditions in teacher-led AI development

Boundary conditions	Enabling factors	Constraining factors
Resource-related conditions	Access to AI coding assistants (some free-tier options available) Open-source models Browser-based development environments that eliminate expensive infrastructure requirements	Limited institutional budgets for professional development Lack of dedicated time for experimentation Resource inequities between schools
Policy and institutional context	School leadership supports teacher innovation Flexibility in curriculum implementation	Rigid AI policies that mandate specific commercial platforms Availability of certain technologies (as experienced in Hong Kong contexts) Neoliberal accountability measures that demand justification through excessive administrative labor
Teacher status and positionality	Senior teachers possess institutional authority to champion new approaches Early-career teachers bring technological fluency and eagerness to experiment	Senior teachers sometimes resist innovation (as Yang's older colleagues demonstrated) Early-career teachers face precarity in their institutional positions, making them vulnerable when taking pedagogical risks Teachers in probationary periods or temporary contracts may avoid AI development for fear of being seen as diverting effort from conventional responsibilities

Knowledge and capacity	Teachers' pedagogical content knowledge (PCK) Technological pedagogical content knowledge (TPCK) Multilingual repertoires for accessing global resources	The "technical mystique" – the misconception that AI development requires unreachable expertise Inadequate teacher education programs that fail to prepare educators for critical evaluation of commercial AI tools and platforms
Community and support structures	Peer support through teacher-teacher communities of practice (COP) Shared problem-solving within teacher-teacher communities of practice (COP) Validation that contrasts with top-down university-led initiatives	Absence of such communities Reliance solely on external experts Constrains teachers' capacity to develop sustained, context-appropriate innovations

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

(2) school policy requiring all AI tools to go through lengthy approval processes; (3) emotional labor of justifying the project to skeptical administrators who prefer commercial solutions; (4) language barriers in accessing English-language technical documentation; and (5) precarity from being on a fixed-term contract, making risk-taking professionally dangerous. Understanding these boundary conditions helps us move beyond simplistic narratives of "teachers just need training" toward recognizing the complex ecology of entangled factors that can enable or constrain teacher-led AI development. Not all these conditions can be controlled by individual teachers, which is why we emphasize building communities of practice and advocating for institutional support structures that expand enabling conditions and mitigate constraining ones.

6.7 An Invitation to Continuing the Dialogue and Journey

We return, in closing, to what this Element has tried to offer: not a manual but a provocation. A story of opening one crack with honest acknowledgment of its limitations and conditions. An argument that multiple pathways to agency deserve exploration and that no single pathway is right for all teachers. A call for dialogue between communities that have too often talked past each other, to the detriment of students and educators alike.

To educators considering this or other pathways to agency: You are not alone in feeling that something has been lost and wanting to reclaim it. The erosion of professional meaning, the displacement of intrinsic purposes by external demands, the sense that your expertise is being devalued by technologies you did not choose and do not control – these experiences are widespread. They are not signs of your inadequacy but symptoms of structural conditions that affect educators across contexts. Taking the initiative to address them, whatever form that initiative takes, is not self-indulgence but professional responsibility.

To technologists seeking genuine educational partnership: Educators are ready for dialogue. Many teachers are eager to contribute their pedagogical knowledge to technology development; they simply need pathways for doing so that respect their expertise and accommodate their constraints. Creating those pathways is partly your responsibility. It means going beyond focus groups and user testing to genuine collaboration. It means treating educators as co-designers rather than end users. It means being willing to have your assumptions challenged by people who know things you do not know.

To institutions, schools, universities, education systems: Create conditions for agency, not just adoption. Teachers cannot take the initiative if their time is consumed entirely by external demands. They cannot develop technical literacy if professional development focuses only on using predetermined tools. They

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

cannot contribute to shaping educational AI if their voices are excluded from technology decisions. The structural conditions that erode teacher agency are not inevitable; they are choices that institutions make and can unmake.

We close with a final, smaller invitation. Return to the very small: the PowerPoint that will not go full screen, the projector that will not connect. Consider the assumption that pervades many classrooms and conferences we have been to: that technology is naturally young people's domain, that students possess innate "tricks" that teachers lack. When teachers routinely ask students to handle these moments, something subtle but consequential happens. Technical capability becomes reframed as generational property rather than learnable skill. Each small delegation reinforces the narrative that "this is not for me." We are not suggesting teachers must never ask for help since asking for help is reasonable and often efficient. But we are suggesting that the pattern of routinely outsourcing even trivial technical tasks to students participates in constructing the very mystique that excludes teachers from larger technological agency. The accumulated weight of these small surrenders shapes professional identity over time, reinforcing the sense that technology belongs to others.

Perhaps before building an AI system, a teacher might simply decide to learn how to put their own PowerPoint in full screen. Not because it is difficult, but precisely because it is not. And because reclaiming agency sometimes begins with the smallest refusal to cede it.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

References

Bakhtin, M. M. (1981). *The dialogic imagination: Four essays* (C. Emerson & M. Holquist, Trans.). University of Texas Press.

Bandura, A. (1997). *Self-efficacy: The exercise of control*. W. H. Freeman.

Dai, D. W. (2024). Interactional competence for professional communication in intercultural contexts: Epistemology, analytic framework and pedagogy. *Language, Culture and Curriculum*, *37*(4), 435–455.

Dweck, C. S., & Repucci, N. D. (1973). Learned helplessness and reinforcement responsibility in children. *Journal of Personality and Social Psychology*, *25*(2), 109–116. https://doi.org/10.1037/h0034248.

Fengchun, M., & Mutlu, C. (2024). *AI competency framework for teachers*. UNESCO. https://unesdoc.unesco.org/ark:/48223/pf0000391104.

Fincham, F. D., Hokoda, A., & Sanders, R., Jr. (1989). Learned helplessness, test anxiety, and academic achievement: A longitudinal analysis. *Child Development*, *60*(1), 138–145. https://doi.org/10.2307/1131079.

Google. (2024). Environmental report 2024. www.google.com/about/datacenters/efficiency/.

Hochschild, A. R. (1983). *The managed heart: Commercialization of human feeling*. University of California Press.

International Task Force on Teachers for Education 2030. (2025). *Promoting and protecting teacher agency in the age of artificial intelligence* [Position paper]. https://teachertaskforce.org/sites/default/files/2025-09/1149_25_Promoting%20and%20Protecting%20Teacher%20Agency_FINAL_3Sep.pdf.

Latour, B. (2005). *Reassembling the social: An introduction to actor-network-theory*. Oxford University Press.

Law, J. (1992). Notes on the theory of the actor-network: Ordering, strategy, and heterogeneity. *Systems Practice*, *5*(4), 379–393. https://doi.org/10.1007/BF01059830.

Lemke, J. L. (1990). *Talking science: Language, learning, and values*. Ablex Publishing.

Li, P., Yang, J., Islam, M. A., & Ren, S. (2023). Making AI less "thirsty": Uncovering and addressing the secret water footprint of AI models. arXiv. https://arxiv.org/abs/2304.03271.

Liu, Y., & Lin, A. M. (2025). CLIL in English-medium nursing education: Teacher collaboration via translanguaging: Trans-semiotising pedagogy for enabling internally persuasive discourse and professional competencies. *Education Sciences*, *15*(8), 983.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Luccioni, A. S., Jernite, Y., & Strubell, E. (2024). Power hungry processing: Watts driving the cost of AI deployment? In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency* (pp. 85–99). ACM. https://doi.org/10.1145/3630106.3658542.

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record*, *108* (6), 1017–1054. https://doi.org/10.1111/j.1467-9620.2006.00684.x.

Muldoon, J., Graham, M., & Cant, C. (2024). *Feeding the machine: The hidden human labor powering AI*. Bloomsbury.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L. M., Rothchild, D., So, D., Texier, M., & Dean, J. (2021). Carbon emissions and large neural network training. arXiv. https://arxiv.org/abs/2104.10350.

Seligman, M. E. P., & Maier, S. F. (1967). Failure to escape traumatic shock. *Journal of Experimental Psychology*, *74*(1), 1–9. https://doi.org/10.1037/h0024514.

Shulman, L. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher*, *15*(2), 4–14. https://doi.org/10.3102/0013189X015002004.

Wang, L., Ertmer, P. A., & Newby, T. J. (2004). Increasing preservice teachers' self-efficacy beliefs for technology integration. *Journal of Research on Technology in Education*, *36*(3), 231–250. https://doi.org/10.1080/15391523.2004.10782414.

Zhai, X. (2025). Transforming teachers' roles and agencies in the era of generative AI: Perceptions, acceptance, knowledge, and practices. *Journal of Science Education and Technology*, *34*, 1323–1333. https://doi.org/10.1007/s10956-024-10174-0.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

AI Disclaimer

Claude Sonnet 4.5 was used for proofreading and making grammatical and stylistic revisions; all outputs were reviewed and verified by the authors.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Cambridge Elements

Generative AI in Education

Mark Warschauer

University of California, Irvine

Mark Warschauer is Distinguished Professor of Education at the University of California, Irvine, with affiliated faculty appointments in the Departments of Informatics, Language Science, and Psychological Science. He is a member of the National Academy of Education and the director of the UCI Digital Learning Lab. Professor Warschauer is one of the most influential scholars in the world on digital learning, digital literacy, and the use of AI in education. He has published 12 books on these topics including with MIT Press, Cambridge University Press, Teachers College Press, and Yale University Press, and some 300 scientific articles and papers. His work has been cited more than 48,000 times, making him one of the most cited researchers in the world on educational technology. He previously served as founding editor of *Language Learning & Technology* and inaugural editor of *AERA Open*.

Tamara Tate

University of California, Irvine

Tamara Tate is a Project Scientist at the University of California, Irvine, and Associate Director of the Digital Learning Lab. She leads the Lab's work on digital and online tools to support teaching and learning including generative AI, partnering with school districts, universities, nonprofit organizations, media and tech developers, and others in iterative development and evaluation. As the PI of a NSF-funded grant, she is studying the use of generative AI in undergraduate writing courses. She also studies secondary student writing as a member of the IES-funded national WRITE Center. She received her B.A. in English and her Ph.D. in Education at U.C. Irvine and her J.D. at U.C. Berkeley.

Editorial Board

Stephen Aguilar, *University of Southern California, US*

Maha Bali, *American University in Cairo, Egypt*

Irene-Angelica Chounta, *University of Duisburg-Essen, Germany*

Shayan Doroudi, *University of California, Irvine, US*

María Florencia Ripani, *Ceibal Foundation, Uruguay*

Bart Rientes, *The Open University, UK*

Neil Selwyn, *Monash University, Australia*

Jiahong Su, *The University of Hong Kong*

Ulrich Trautwein, *University of Tübingen, Germany*

Ying Xu, *Harvard University, US*

About the Series

Generative AI is one of the most disruptive technologies in modern history, with the potential to dramatically transform education for better or worse. This series will address cutting-edge topics on the intersection of generative AI with educational research and practice for diverse learners from early childhood to adult.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107

Cambridge Elements

Generative AI in Education

Elements in the Series

Generative AI in Computer Science Education
Diana Franklin, Paul Denny, David A. Gonzalez-Maldonado, and Minh Tran

Generative Artificial Intelligence and Language Teaching
Benjamin Luke Moorhouse and Kevin M. Wong

Academic Integrity in the Age of AI
Tricia Bertram Gallant, Mary Davis, and Zeenath Reza Khan

Growing up with Artificial Intelligence
Ying Xu and Kunlei He

Teachers as AI Developers
Qinghua Chen, Yiqi Liu, and Angel M. Y. Lin

A full series listing is available at: www.cambridge.org/EAIE

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 05 Nov 2026 at 10:13:52, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009781107