AI Microservices for Sustainable Innovation in Education

Peter Johnson[a], Phil Ramsden[b], Marcus Messer[a]

[a]*Department of Mechanical Engineering, Imperial College London London, UK*
[b]*Department of Mathematics, Imperial College London London, UK*

Abstract

Automation in education is increasingly applied to processes requiring pedagogical judgement. We argue that such automation should be separate from general platforms used to administrate learning and teaching activities. We propose an architecture for the wider ecosystem based on platforms calling external microservices to provide pedagogical judgements, and that this approach will enable innovation and foster ethical approaches.

We provide a theoretical framework for microservices based on a literature review of software architecture principles and a characteristion of the education sector. We then present results of practical deployment of an exemplar system for providing automated formative feedback. The system delegates all pedagogical judgement to external microservices.

The system was deployed to thousands of active student users, covering 13,000 study tasks and 1.5 million calls to external services. We present case studies of the general deployment, and two specific cases of automated feedback on university-level mathematical proofs, and school level essays in English language. The case studies on deployments provide large scale evidence for the feasibility of microservices to provide pedagogical feedback.

The evidence presented also supports the theoretical framework that microservices for pedagogical judgment promote innovation from a wide variety of experts in niche areas, emancipate the services to use technology relevant to the specific problem, prevent platform lock-in leading to pedagogy lock-in, and foster transparency, equity of access, and higher quality services.

We discuss the future prospects of a microservice architecture for the education ecosystem, including the potential risks of such an approach and future research needs. Our main conclusion is that we present a framework for a new approach to automating pedagogical judgments, and novel, large scale, practical evidence for its feasibility.

Keywords: microservice, AI, innovation, ethics

---

1. Introduction

The use of software for automation in education can be broadly split into two purposes: administrative and educational. In practice, until recently, most automation was administrative. Educational decisions that require judgement and accountability have previously been automated in niche applications (Hollingsworth, 1960;

Valenti et al., 2003; Kulik and Fletcher, 2016) but are now rapidly growing, principally owing to the wide availability of Large Language Models since 2023 (Wang et al., 2024; Zhang et al., 2024).

Pedagogical judgement is needed in, for example: content generation, feedback generation, assessment, analysis, advice, and even decisions such as academic progression. In this paper, we argue that automated pedagogical judgement should be separated from the platforms that are used for general administration of teaching. We aim to articulate why this separation is required, how it should be realised – using microservices – and what are the architectural tensions to be managed; we show a practical example of such a separated system deployed on a scale of thousands of students across multiple institutions and with millions of microservice calls annually.

Our arguments are of two types: first, we argue from practical and technical views of how software innovation works; and second, we make the ethical case for separation to enable educator agency, encourage service transparency, and to provide scaling that leads to equity of access and high quality services.

In this paper we consider it out of scope to ask *whether* pedagogic judgements should be automated, what might be considered *good* judgement, or *how* to automate certain judgements. Our view is at the ecosystem level where we ask, if the automation is happening, and is going to grow, what kind of architecture is appropriate?

The article begins with a literature review and theoretical framework in two parts, providing context for each of the two parts of our argument. We recap the principles of software architecture and modularity, and then focus on interoperability and microservices. We then review the key ethical aspects of the automation of pedagogical judgements, and critically analyse how using microservices affects these ethics. In anticipation of case studies, later in the paper, on the specific example of automated formative feedback, we also summarise the literature in this area.

In the Section 3 we present an exemplar: a learning management system (LMS) specialising in automated formative feedback, where all pedagogical judgements are provided by external microservices. We summarise the user experience, technical architecture, and three case studies used to evaluate the system. In each case study we provide both the methods applied, and the results, focusing on the effects of using a microservice architecture to foster innovation and enable ethical automation of pedagogical judgement.

In Section 4 we discuss key issues in the use of microservices, including the needs for future research, and the technical and ethical tensions that are inherent in such an approach. In Section 5, we conclude that microservices are the recommended architecture for automating pedagogical judgements, provide guidance on using microservices, and point to key developments that are required in the near future.

2. Literature review and theoretical framework

This paper takes a position that the software architecture of the education ecosystem would benefit from a set of microservices that each automate separate, granular pedagogical judgements. In this Section we review the background and literature on software engineering, beginning with a review of modularity in systems engineering.

We critically analyse modularity from a technical perspective, highlighting its role in facilitating innovation under certain circumstances — especially when sub-sytems, such as AI for pedagogical judgements, evolve more rapidly than the wider system. We extend our technical review of modularity to interoperability, and the specific architectural pattern of microservices. We distinguish microservices from general applications, identify the tensions when designing an ecosystem, and relate these to the specific case of education.

We then turn to the ethical perspective, reviewing platform lock-in, transparency, accountability, bias, and equity. Our ethical analysis also leads to a recommendation of microservices. Finally, to complete the context for the study that follows, which focuses on the specific example of automating formative feedback, we review automation efforts in that area.

2.1. Historical context of systems engineering and modularity

Systems engineering considers complex systems as modules that communicate through interfaces, with the inner workings of modules 'hidden' from the separate modules that can be independently replaced. In the early industrial revolution Whitworth pioneered the ideal to "exchange the one [part] for the other" (Whitworth, 1858)[p.12], a concept credited with driving American industrialisation thereafter (Hounshell, 1984). Generalising the concept of modules, for example to complex biological systems, is the idea that

> in a nearly decomposable system, the short-run behaviour of each of the component subsystems is approximately independent of the short-run behaviour of the other components. (Simon, 1962)[p.474]

In both nature and engineering, modular sub-systems can change on a short timescale, without needing the wider system to change at such pace. The concept of separate timescales of change for modules within a system is important for innovation in AI, because technologies and practices around AI are changing rapidly in relation to the wider systems of which they are a part. Modularity is therefore key to sustaining innovation.

Software engineering embraces the concept of modularity, which emphasises that modules should hide 'decisions that are likely to change' (Parnas, 1972), a concept that remains important today (Garlan et al., 1993; Perry and Wolf, 1992; Bass, 2012).

From an economic and organisation perspective Baldwin and Clark (2000) showed how modularity facilitates innovation through parallel development, specialisation, competitive entry at the module level, and platform ecosystems. These findings challenge us to ask the extent to which modules within a software application, platform, or ecosystem should be independent, to facilitate these mechanisms that enhance innovation. Decentralising innovation to separate modules is common in software development (Sullivan et al., 2001). In the case of education software, to what extent should modules that automate pedagogical judgement be independent?

2.2. Critical analysis of modularity

Baldwin (2018) identified that independent modules benefit platform- rather than step-orientated systems; learning management systems (LMS) fit the 'platform' pattern. Baldwin (2018) also emphasised that the pace of change of technology has an important influence on the appropriate architecture. For the current rapid growth and change of AI technologies, isolation to enable innovation is essential.

The limitations of the analysis of Baldwin (2018) and Baldwin and Clark (2000) are that they study single-platform systems, whereas in education there are many platforms, forming an eco-system. Jacobides et al. (2018) distinguished eco-systems by the number of platforms, 'n', including single, low-n, and high-n. As consumers we recognise the Android and iOS platforms on our devices as a low-n system, whereas education is a high-n ecosystem with many virtual learning environments (VLE) or Learning Management Systems (LMS), and the applications ('complementers') that can be deployed through those platforms.

In a multi-platform eco-system, complementers, who develop modules for one or more of the platforms, have a dilemma whether to 'multi-home' their module, i.e. whether to engineer their modules to work within (to 'complement') more than one platform. As examples of multi-homing, the Microsoft Office suite is a multi-homed complement to consumer operating systems, and in education there are many applications that connect to LMSs.

A sector-wide perspective of education invites the question: under which conditions is 'high-n multi-platform' appropriate? In other words, is the current situation good? Jacobides et al. (2018) refine this question to three aspects:

(1) Are complementarities modular, as per Baldwin and Clark (2000)?

(2) Is supermodularity low (no strong advantage in tight integration)?

(3) Are multi-homing costs low?

If true in all cases then high-n platforms are technically appropriate. Are these conditions met in education?

On question (1) we assume here that the reader agrees that the different processes in education can be modularised in the sense of Parnas (1972), and that this is desirable in the sense of Baldwin and Clark (2000) because of the complexity of the system. On question (2) we note that some modules ('applications'), such as STACK, integrate tightly into a particular VLE (Moodle in this case), suggesting some supermodularity. On question (3) there are non-negligible multi-homing costs. The LTI system is an example of reducing multi-homing costs, subject to the (non-zero) cost of engineering LTI connectivity into the application.

The technical grounds for a high-n multi-platform system are, therefore, met to some extent but not entirely. The sector is not best placed to channel innovation in AI for pedagogical judgement.

Should we improve the conditions to facilitate modularity and multi-homing, or should we reduce the number of platforms? The latter is not an option in practice, so we should seek the former. This means we should (1) build modular systems; (2) beware of supermodularity as not conducive to innovation; and (3) decrease multi-homing costs.

2.3. Interoperability

A key characteristic of modules that can reduce multi-homing costs is interoperability. For example, IoT ecosystems are also high-n multi-platform, and in that sector interoperability facilitates many independent platforms, products, applications, and services that can work as an eco-system (Bröring et al., 2017). Interoperability is distinct from modularity in the sense that modules become not just alternative options, but directly substitutable for each other.

As an example in education, there are multiple modules (applications) that connect to LMSs via LTI to provide assessment and feedback utilities. There is choice between these modules. For example, 'Möbius' provides assignments that can be graded, with grades returned to the LMS, and so does Turnitin. Both applications connect via LTI and both return grades to the LMS. They can innovate independently, and the most desirable can be chosen by the educational institution, without changing their LMS. Modularity is present, and the benefits are realised. However, in this example these modules are not 'interoperable' — they cannot be substituted for each other. They are configured differently. There is friction in swapping one for the other.

Palfrey and Gasser (2012) stress interoperability as replace-ability: not just the ability to connect, but the ability to substitute or exit. Palfrey and Gasser (2012) provide a holistic perspective of the technology, data, social, and organisational layers (the 'cake' model) of interoperability summarised in Table 1.

Layer	**Benefits**	**Risks**
Technology	Innovation, lower barriers, diverse participants, adaptability	Fragility, performance, security, tight integration
Data	Sharing, analysis, aggregation	Loss of context; misuse of data
Social	Accessibility, collaboration	Loss of context, local control
Organisational	Policy alignment, market access, regulatory consistency, democratic	sovereignty, internal norms

Table 1: The cake model of Palfrey and Gasser (2012) showing the benefits and risks of *interoperability.*

Interoperability should only be considered if the benefits are valuable, and if so the risks need to be managed. An insightful example is financial payments, which are interoperable despite the security and data risks; in this case the benefits were worth managing the risks. The benefits of interoperability are aligned with those of modularity, the distinction being that interoperability enhances both the benefits and the risks.

In education the broad, diverse, and complex nature of the problems of automation in pedagogical judgement requires innovation from diverse participants, adaptability to local contexts, collaboration across the sector, and to operate in a high-n, multiplatform ecosystem. These are arguments for interoperability, which if followed implies that the risks must be managed and Table 1 provides guidance for developing interoperability.

2.4. Microservices

So far we have honed our arguments on interoperable modules developed by complementers. One potential architectural pattern to implement interoperable modules

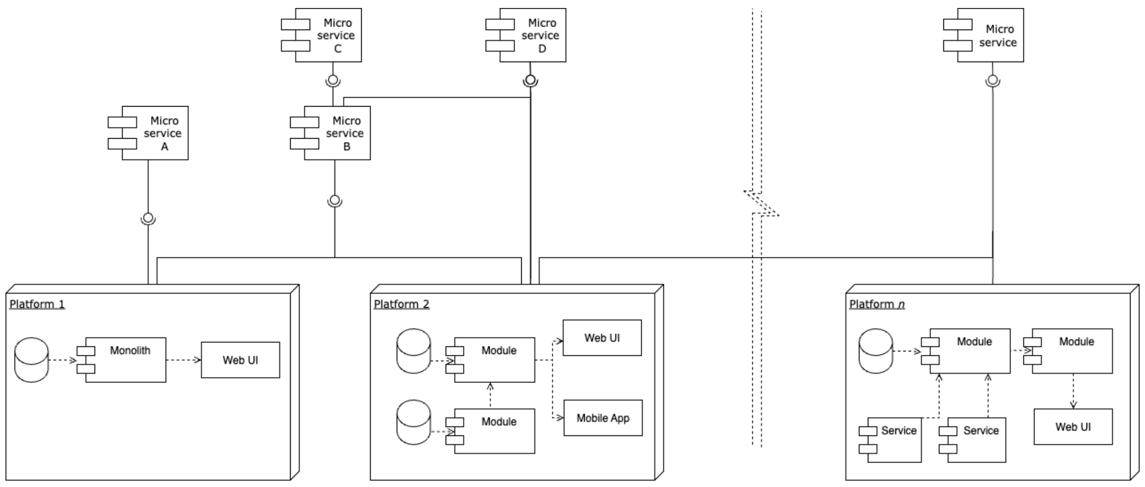


Figure 1: A conceptual sketch of a high-n, multiplatform ecosystem, with external microservices. The internal architecture of the platforms is not constrained hence various options are portrayed; the key point is the large number of independent, external microservices that can each be called by many platforms.

is called *Microservices* (Lewis and Fowler, 2014). The distinction between a 'microservice' and a more general modular, interoperable 'application' is a focus on small, granular functionality.

For example, applications that connect to an LMS via LTI typically have a range of functions, leading to broad and complex interactions between the platform and the application that connects to it. In contrast, microservices specialise in smaller, more granular tasks. Microservices deploy and execute independently, for example not sharing memory or compute resources with other parts of a system, and are updated separately. Many microservices are used via a simple HTTP request and response. Although not conceptually new, the growth of microservices in the last 10 years has been facilitated by container technology, cloud computing, and continuous integration/deployment operational approaches (Wang et al., 2021).

A conceptual sketch of what a microservice architecture might look like in a high-n multi-platform ecosystem like education is given in Figure 1. A large number of microservices is available and can be used by multiple platforms; microservices can also call other services (see also Bröring et al., 2017).

Following Lewis and Fowler (2014) and the review of Dragoni et al. (2017) we can identify key benefits and risks in using microservices. Microservices are appropriate when key measures of independence are required, namely development, deployment (memory and compute), scaling, and technical stacks. Additionally, the need for modularity, interoperability, and granular functions, are reasons to consider microservices. The granularity ensures the principle of innovation through modularity, freeing functions to be developed independently depending on the expertise required.

Other requirements to justify microservices include the feasibility (or acceptability) of message-based and higher-latency interactions.

Reasons not to use microservices include the overhead of communication between teams, the overall system complexity, potential immaturity of the system, error tracking and management when issues arise, and friction in deployment due to the complexity of system-wide testing and the multi-stage nature of system-wide updates. A tabulated trade-off for microservices is included in Table B.6 in the Appendix.

In summary, a monolith architecture works well as a default, except where it is critical that services can be developed and deployed independently of one another.

In this paper, we argue that educational software involving automated pedagogical judgement is just such a case, for which a microservice architecture is appropriate. In addition to the balance of issues summarised in Table B.6, we recall our review that education software requires a low cost of multi-homing and granular, independent tasks functions. The microservice architecture, with interaction through messaging, is ideal to meet this need.

The arguments so far have focused on technical aspects, and we will now consider the ethics of automating pedagogical judgement.

2.5. Ethical aspects of pedagogical judgements

The distinction between pedagogical judgements and administrative processes was made by Selwyn et al. (2023), who advocated for resistance to automation of the role played by the teacher. Nevertheless, powered by the revolution in large language models, a rapid rise in empirical studies of LLMs in education has been reported, highlighting

> "GenAI's role as an assistant and facilitator in learning support, a subject expert and instructional designer in teaching support, and its contributions to diverse feedback methods and emerging assessment opportunities." (Zhang et al., 2024)

Pedagogic judgements are now, and quite suddenly, being automated on a much wider basis than in the past. We review the ethics of automatic pedagogical judgement from the perspective of *platform lock-in*, *transparency*, *accountability*, *bias* and *equity* of access.

Platform lock-in describes a situation where institutions are unlikely to change platforms. In addition to the multi-year timescale to procure a platform and integrate it into organisational workflows, the effect of 'generative entrenchment' refers to the decreased cost of future investments in the platform that has already been integrated, due to compatibility (Pangrazio et al., 2023).

As pedagogic judgements become part of platform offerings, platform lock-in risks becoming pedagogy lock-in — a new, and major, ethical issue. We argue that architectures that break this link create a stronger ethical framework for the automation of pedagogical judgement. As a minimum, such automation should not be inherently governed by the platform and should be configured by teachers or by separate applications running on the platform. Microservices, while an extreme version of this modularity, would facilitate breaking this chain.

Transparency is a clear issue in debates around ethics in AI (Holmes et al., 2022; Klimova et al., 2023; Fu and Weng, 2024). If a pedagogic judgement is automated, it should be clear what process was followed. The black-box nature of neural-network-based technologies conspire against full transparency, but even if processes themselves are opaque, the patterns of outputs can be made transparent.

Model cards (Mitchell et al., 2019) can contribute to transparency, which would require education-specific tests. Tests such as the 'benchmark of pedagogical knowledge' by Lelièvre et al. (2025) show an early example, however the tests require more nuance around the contested nature of pedagogy and the importance of context. The key point for the present work is that incentives to provide model cards, and to test transparently and according to the needs of the community, are not aligned if automated pedagogical judgements are within a platform. For example, if testing of an automated process shows bad results, will platforms transparently communicate the result? Separating the processes is essential for healthy incentives to report on automation processes transparently. Interoperable microservices would enhance competition, increase the agency of the educator community, and be more conducive to transparency.

Accountability is integral to AI ethics debates (Holmes et al., 2022; Klimova et al., 2023; Fu and Weng, 2024). Accountability refers to the responsibility that educators take for the experience of the students, including the ramifications of automated pedagogical judgements. If the educator cannot freely choose from different ways to automate pedagogical judgement, and expectations are put on the educator to use the available services, then both the agency and accountability of the educator are at risk. Choice via interoperable microservices opens the possibility of greater accountability.

The arguments so far on lock-in, transparency, and accountability are all grounded in the desire for agency within education institutions to choose how (if at all) pedagogical judgements are automated. The element of choice is also essential to debates around issues of bias, equity, and power. All automated processes contain bias and the knock-on effects can be unforeseen (Klimova et al., 2023). Bias is a complex ethical area, for example it is common to state that 'no bias' is a principle of ethical AI (Klimova et al., 2023), which betrays a lack of understanding that training models on data is the practice of introducing bias – bias towards the data set. Within the scope of the present article it is sufficient to say that choice and educator agency are a pre-requisite to understanding and managing bias. Microservices facilitate the choice and encourage transparency – but do not inherently address the bias.

Equity as the availability of technology to all institutions is a significant issue (Pangrazio et al. (2023)). Microservices break the chain of platforms and services, so that inaccessibility of platforms (e.g. due to cost or organisational barriers) does not create a barrier to services available on the platform. Equity can play out in nuanced ways, for example if the most advanced, or high quality, automated pedagogical judgements require significant investments by educational institutions to tune to their context (for example iterative prompt engineering), then institutions with more resource will develop higher quality services (Buckingham Shum et al., 2023).

To the extent that institutions want to share the fruits of their investments, they can make services available for other institutions to improve equity of access. While this argument is true of all educational technology in principle, the point made here relates to the low barrier to sharing when interoperable microservices are used in an eco-system (Fullan and Langworthy, 2014; Reigeluth et al., 2013; Omiyad Network, 2019; Otto and Kerres, 2022; Gottschalk and Weise, 2023).

The scale of application of educational technology also affects its quality. Larger scale ensures a more diverse user base for testing, and a larger resource for development. A common pattern in the literature is to test a single cohort local to the developers (Deeva et al., 2021; Ma et al., 2014; VanLehn, 2011). Interoperable microservices open the possibility of much larger testing sets, for example millions of participants.

To summarise, microservices foster a more ethical ecosystem by separating services from platforms, enabling educator choice and agency, enhancing transparency, promoting equity of access, and increasing quality.

2.6. Automated formative feedback

Following a review of the technical and ethical perspectives, we turn our attention to a specific type of judgement to facilitate a more concrete, practical analysis later in the paper. Of the many forms of pedagogic judgement that could be automated, we use automated formative feedback as a case study in this article. In this Section we briefly summarise research on automated formative feedback, which is defined (Black and Wiliam, 1998; Sadler, 1989) as informing a student on:

- A goal ('Where am I going?')
- Progress towards the goal ('Where am I?')
- How to progress toward the goal ('Where shall I go next?').

Formative feedback is a prime candidate for the automation of pedagogic judgements because it is known as a high impact intervention (Hattie and Timperley, 2007; Hattie, 2009), but there are resource constraints in practice that limit its impact. For example, in schools teachers spend time outside the classroom generating formative feedback (Merrimack College, 2023; Allen et al., 2021; Airasian, 1997; Burgess et al., 2023; Butt and Lance, 2005; Henderson et al., 2019) and in university formative feedback can be mistargeted due to its association with summative assessment (Winstone and Boud, 2022) and is often not given for its purely formative value. A focus on formative feedback for early automation is also practical due to the low stakes and high tolerance for risk if there are errors in the feedback.

Empirical guides to good feedback are reviewed by Shute (2008), and theoretical frameworks reviewed by Panadero and Lipnevich (2022). Some of the existing efforts to automate feedback (formative and/or summative) are listed in Table 2. The hundreds of systems, even pre-ChatGPT, highlight the 'high-n' nature of the automated formative feedback ecosystem.

The trend in studies of automated feedback was to develop an independent application or platform, that does not offer interoperability pedagogical judgement algorithms. For example scripts in Maple work in Möbius and can be shared but only within that platform; similarly scripts in Maxima work in STACK and can be shared but only within that application. One exception is Google's LearnLM (Jurenka et al., 2024) which is now embedded in Google Gemini and is generally accessible via an API. At the time of writing, Google's competitors were similar educational modes for their LLMs. However, in cases like LearnLM in Gemini, when a general model is available, prompting (context engineering) will still be part of

Table 2: Examples of over 200 automated feedback systems

Type	Examples
Automated assessment system (AAS)	STACK (Sangwin, 2007, 2013), Lambda Feedback (Johnson et al., 2025), Artemis (Krusche and Seitz, 2018), JACK (Goedicke et al., 2008), MATLAB Grader (Tejado et al., 2023), and 109 Systems listed by Deeva et al. (2021).
Intelligent tutoring systems (ITS)	Carnegie CLEAR tutor, ASSISTments (Heffernan and Heffernan, 2014), 22 systems reviewed by (Ma et al., 2014), and 68 systems in Graesser et al. (2012)
Writing assistants	OnTask & SRES, ECoach, AcaWriter (Buckingham Shum et al., 2023), and six systems in Nunes et al. (2022).
AI assistant	Khanmigo (Shetye, 2024), seven systems reviewed by Labadze et al. (2023), Google LearnLM and 43 other systems reviewed by Jurenka et al. (2024).

the automation process, and the prompting will be contextual. Users in a similar context could benefit from access to the same prompts, but may want to switch the underlying model while keeping the prompts.

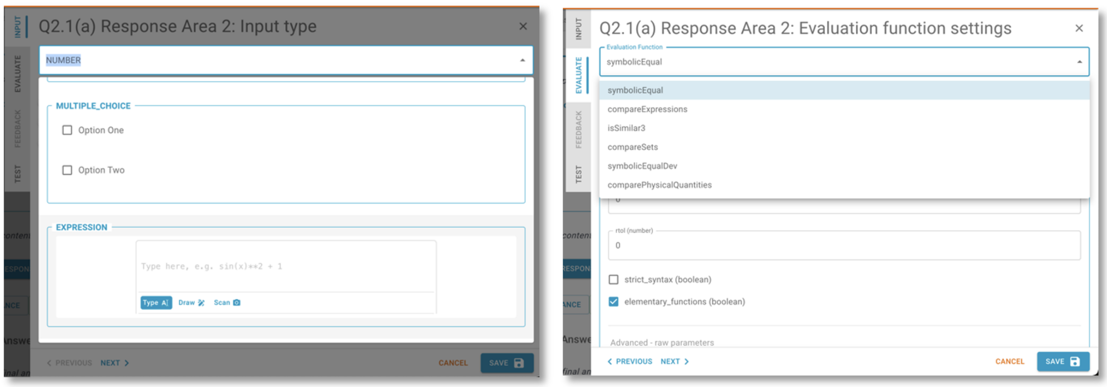
3. Case studies

In this Section, we present an exemplar of a deployed and tested educational platform that employs microservices to make pedagogical judgements. We focus on a specific case where formative feedback is provided by a microservice. By focusing on a specific application, we illustrate how microservices can be used to make pedagogical judgements; the reader can then extrapolate a similar use for other applications such as content generation, summative assessment, and so on.

We present [platform name], which is a specialist learning management system (LMS) for self-study, providing automated formative feedback. All feedback is provided by interoperable microservices. We present the three separate case studies using the platform, in each case highlighting the evidence to support the arguments made in the background and literature review. All microservices use publically available code [reference removed for double blind review]. The first Case study focuses on the scale of deployment and the development of microservices. The next two Case studies, with contrasting applications in university mathematics and school English language, provide deeper insight into the pedagogical judgements being automated, and the concrete ways in which microservices facilitated innovation and supported ethical approaches.

3.1. Description of the [platform name] application

The platform is web-based for users. The base of the application is a content management system, where teachers curate study materials and publish them to students. Content management is based on markdown, with mathematics, images, and videos embedded. Secure login and role-based access allows teachers, administrators, and students the appropriate functionality. Figure 2 shows how a teacher curates automation on the platform, selecting and configuring one of the available



Q2.1(a) Response Area 2: Input type

NUMBER

MULTIPLE_CHOICE

- [ ] Option One
- [ ] Option Two

EXPRESSION

```
Type here, e.g. sin(x)**2 + 1
```

Type Draw Scan

Q2.1(a) Response Area 2: Evaluation function settings

Evaluation Function: symbolicEqual

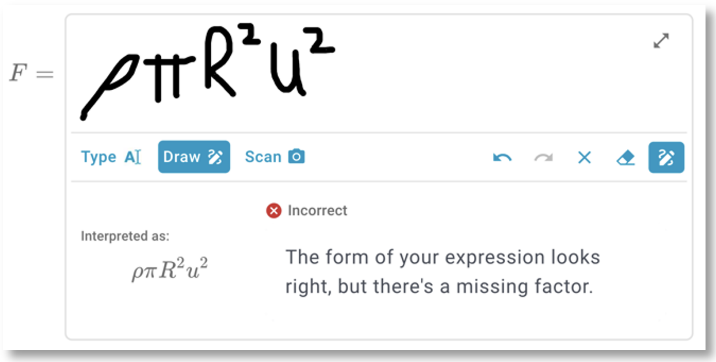
- symbolicEqual
- compareExpressions
- isSimilar3
- compareSets
- symbolicEqualDev
- comparePhysicalQuantities

rtol (number): 0

- [ ] strict_syntax (boolean)
- [x] elementary_functions (boolean)

Advanced - raw parameters

Figure 2: Two key steps for the teacher. Left: selecting an interactive component. Right: selecting and configuring a microservice.



$F =$

Incorrect

Interpreted as: $\rho\pi R^2u^2$

The form of your expression looks right, but there's a missing factor.

Figure 3: Student experience: a handwritten expression is entered, and timely, formative feedback is provided — via a microservice.

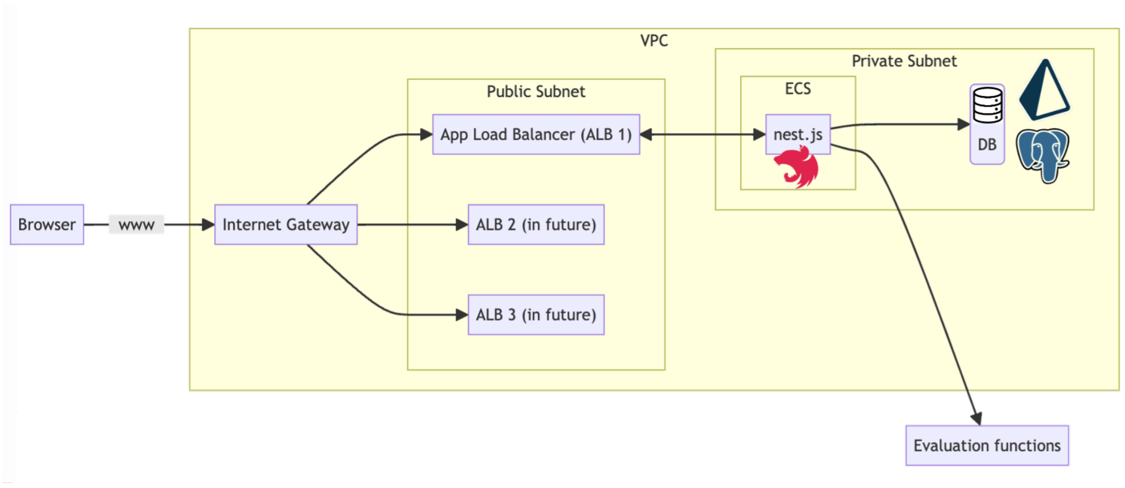


Figure 4: Software architecture for [platform name], from left-to-right: web requests are managed by a load balancer that routes to a server instance, run using nest.js, using Amazon Elastic Container Service (ECS). Within a private subnet, the Database (DB) is built on PostgreSQL and Prisma. The entire application is in a virtual private cloud (VPC). When automated feedback is required, an external 'Evaluation function' is called as a microservice, using an HTTP request.

microservices. Note that no code is required by the teacher. Figure 3 shows an example of the student view, receiving timely, automated, specific, feedback that requires pedagogic judgement.

The platform was developed from the outset with emphasis on microservices for pedagogic judgment. The cloud native architecture follows a traditional structure, with a web-client, server, and database. The client and server are built in TypeScript using 'next.js' and 'nest.js' libraries, respectively. The database is built in PostgreSQL, managed by Prisma, with APIs using GraphQL. The application is built in OCI containers and deployed using continuous deployment/continuous integration on AWS. Horizontal scaling is incorporated with AWS Fargate.

The differentiating feature of the application is the provision of automated formative feedback on student responses to study tasks. The application has no internal mechanism to make pedagogical judgements or generate formative feedback. All feedback is provided by external microservices, as illustrated in Figure 4, called by an HTTP request.

Currently 'Evaluation functions' are the main type of microservice in use, which evaluates a response from a student and provides feedback. 'Chat' functions are also called as microservices and are in growing use but are not detailed here for brevity. Future microservices may include, for example, data analytics processes such as providing high-level feedback to students or predictive analytics for staff; or students. All these external services involve pedagogical judgement, and are therefore outside the functionality of the core application.

To illustrate the operation of the microservices, we show a simple example evaluation of a symbolic (mathematical) expression. An HTTP request is used to pass contextual data to the microservice and to receive a response. An example use of a microservice is given below as a curl request:

```
curl --request GET \
 --url https://XX.amazonaws.com/default/symbolicEqual \
 --header 'Content-Type: application/json' \
 --header 'command: eval' \
 --data '{"response": "x + x + y - y", "answer": "2*x"}'
```

An example output is as follows, showing only the 'result' component of the JSON object that is returned:

```
"is_correct": true,
"feedback": "",
```

In the example the request is to compare a student response $(x+x+y-y)$ to a reference answer $(2*x)$. The microservice `symbolicEqual` uses a Computer Algebra System (CAS, in this case SymPy) and in this example returned a validation metric stating that the response is correct; no further feedback was provided.

The key characteristics of the microservice in this simple example are:

- Contestable: the output involves pedagogical judgement.
- Technology agnostic: no pre-requisites on hardware or run time.
- Interoperable: can be substituted; can be called from any platform.
- Granular: the function is small in scope.

The correctness of the student response $(x+x+y-y)$ is contestable. There are contexts in which the response is not considered correct. The teacher will need to understand the functionality of the `symbolicEqual` service and decide whether it is appropriate to their context.

The HTTP request is not contingent on the run time (in this case Python) or hardware configuration (in this case a cloud function, AWS Lambda). As technologies evolve, the microservices can capitalise. In mathematics, the use of Lean Theorem Prover or Wolfram Language may be fruitful. In natural language, while further external calls to services such as from OpenAI or Google can be made, an alternative future could be that models become smaller and can be fine-tuned and deployed on smaller rented or owned GPU hardware.

The interoperability of the microservice is evident in the example that follows. The `symbolicEqual` service evolved from a pilot programme but can now be replaced by a next generation service, `compareExpressions`. With the same HTTP request but to a different address, the new service gives the following output:

```
"is_correct": true,
"feedback": "The response can be simplified further.",
"tags": ["response = answer_TRUE", "response =
    answer_SAME_SYMBOLS_FALSE"]
```

The new microservice is distinguished by providing a feedback string, and 'tags' to help with later analytics that will drive further improvements to the service. It is

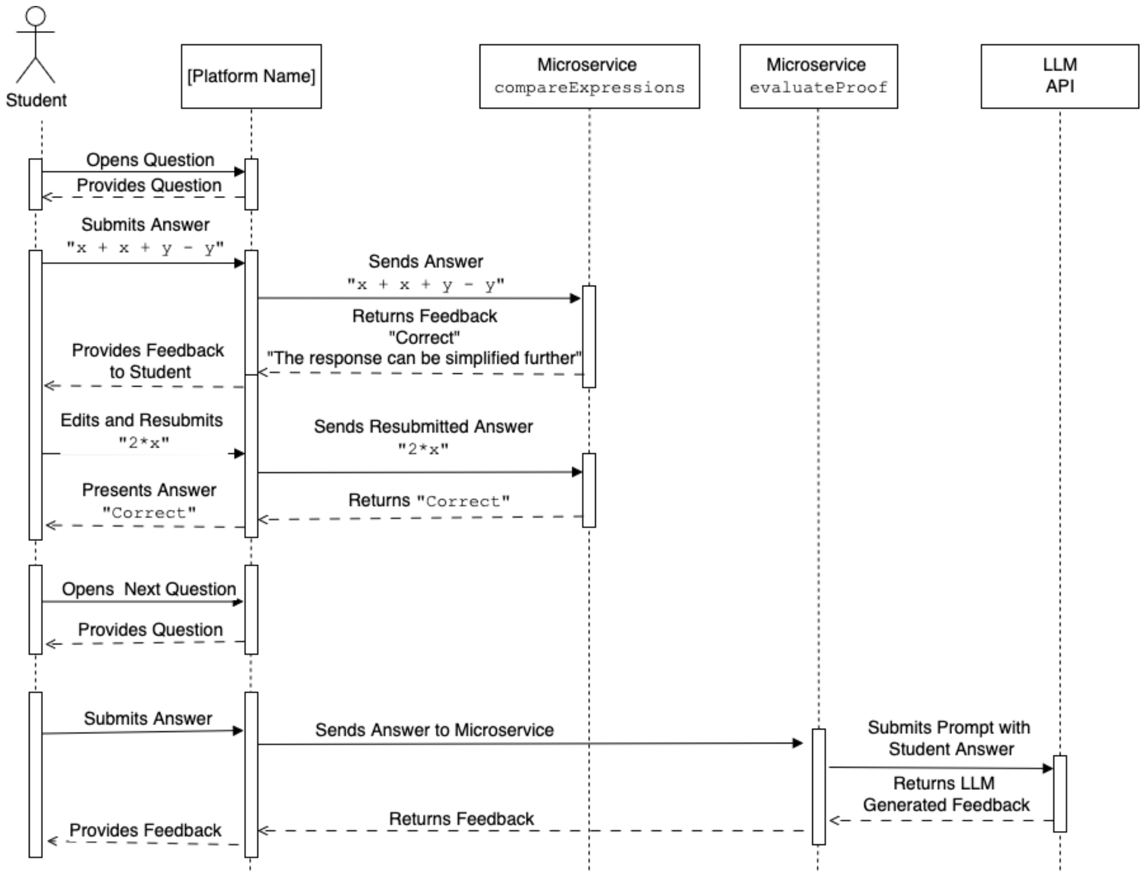

Student
[Platform Name]
Microservice compareExpressions
Microservice evaluateProof
LLM API
Opens Question
Provides Question
Submits Answer "x + x + y - y"
Sends Answer "x + x + y - y"
Returns Feedback "Correct" "The response can be simplified further"
Provides Feedback to Student
Edits and Resubmits "2*x"
Sends Resubmitted Answer "2*x"
Presents Answer "Correct"
Returns "Correct"
Opens Next Question
Provides Question
Submits Answer
Sends Answer to Microservice
Submits Prompt with Student Answer
Returns LLM Generated Feedback
Returns Feedback
Provides Feedback

Figure 5: An example interaction between a student using an LMS, and external microservices providing feedback on various types of questions, including comparing mathematical expressions, and providing feedback on mathematical proofs.

contestable whether the feedback provided is higher quality; for example, teachers may not want to provide such a strong hint or may consider the hint as not strong enough — these are pedagogical judgements. The interoperability of the services allows the teacher to choose or develop an alternative.

An example sequence of how the student interacts with the microservices, via an LMS, can be found in Figure 5. First, the student submits their answer to the LMS, which in turn sends their submission to the microservice `compareExpresions`, which provides the feedback that "The response can be simplified further.". The student then submits a revised version, and in this case, the answer is correct, and no further feedback is provided. Finally, the student loads a different task which requires a proof; they submit a proof to `evaluateProof`, then uses a further call to an LLM to generate feedback, which is passed back via this same `evaluateProof` microservice to the student in the LMS (we discuss `evaluateProof` furher in Section 3.3).

3.2. Case study 1: deployment across an institution

An overall study of institution-wide adoption of the platform at [university name] is provided, emphasising the usage patterns, scalability and flexibility of its use. This study is primarily quantitative and is used to illustrate that the microservice architecture is feasible at scale.

The [platform name] was piloted in 2021, deployed to a single cohort of 180 engineering undergraduate students in 2022, and gradually grew in adoption until 2025 when its deployment covered four faculties, 15 departments, 60 modules, 100 teachers, 3,000 active users (students), 13,000 questions, and approximately 1 million formative feedback events per academic year. Predominant usage was in departments of engineering and physical science at one institution, but it was also used in medicine, business, English, including in multiple universities and schools.

Each module was administered by an independent 'module leader' (academic staff), who curated their self-study content which was akin to a 'tutorial sheet' (UK) or 'Problem Set' (US). As is common in UK universities, these Problem Sets were considered a necessary part of the course of study. The Problem Sets, although expected to be completed in preparation for a summative exam, were not directly summatively assessed.

Teachers, with support from student partners, produced or imported content, edited it to ensure it was accurate and presentable, and then configured automated formative feedback by selecting a microservice and defining the relevant contextual parameters (such as reference answers, accuracy requirements, or acceptable symbols).

Teachers chose freely from the available microservices, including the option to develop their own microservices. The results section will show which services became available, who developed them, and how they were deployed.

Microservices created to launch the platform are listed in Table 3. New services created independently are listed in Table 4 showing by example that the creation of microservices by independent parties is feasible. The number of services (22) is small compared to the number of calls (1.5 million), and the number of educational tasks (approximately 13,000), showing that investing in a microservice is not just economically feasible but can have a large pedagogic impact across a multitude of tasks.

Educational subjects on the platform were predominantly engineering and physical sciences, but also included mathematics, medicine, business, and English. Usage was predominantly in one institution, but pilots in multiple universities and schools are also underway. Those arrangements are nascent and are not reported here.

In the period 2022-2025 microservices were called 1.5 million times. Due to the exponential growth of the platform usage, most of these calls (more than half) were in the academic year 2024-25. The leading microservices in quantity of calls are listed in Table 5. The leading services (48%) are for symbolic mathematical expressions (`symbolicEqual`, array`symbolicEqual`, `comparePhysicalQuantities`, or `compareExpresions`). The new function `compareExpresions` succeeded `symbolicEqual` and `comparePhysicalQuantities`, but the latter are still used in legacy content. The next most common service (32%) is for numerical quantities with tolerances (`isSimilar`), and the next (17%) is for multiple choice questions. Those leading services comprise 97% of calls.

The scale and scalability of the microservices are illustrated by an analysis of the `compareExpresions` service in Figure 6. In Figure 6a the invocations (service calls) per hour show over 400 occasions with over 1,000 invocations per hour implying calls every second at times. Figure 6b shows how often the rate of invocations required

Table 3: Microservices created with the platform. Services are built in Python. These services are independent, but were created by the same team that created the platform.

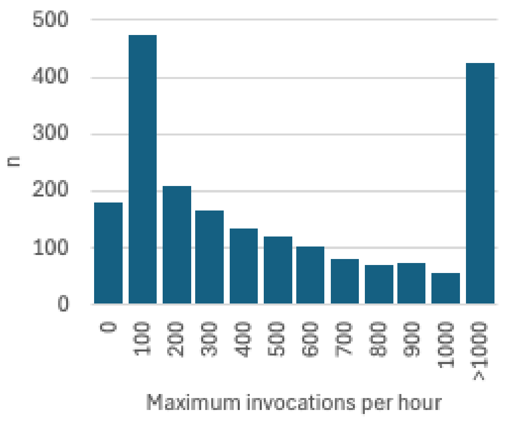
Service	**Technical summary**
`Boiler plates`	Non-deployed, public boiler plates that can be cloned and used to build functions in Python, Lean, Wolfram.
`survey`	Returns 'Thank you, your response has been logged', to enable the platform to be used for surveys.
`isExactEqual`	Compares to strings for exact equality.
`arrayEqual`	Recursively compares each element of an array, invokes exactEqual (service) to execute the comparison. Used for multiple choice.
`isSimilar`	Compares two numbers for equality within tolerances provided as parameters.
`symbolicEqual`	Parses expressions and checks equality with SymPy.
`arraysymbolicEqual`	Recursively compares each element of an array, invokes `symbolicEqual` to execute the comparison.

Table 4: Notable microservices created with the platform. Services are built in Python. These services are independent, but were created by the same team who created the platform.

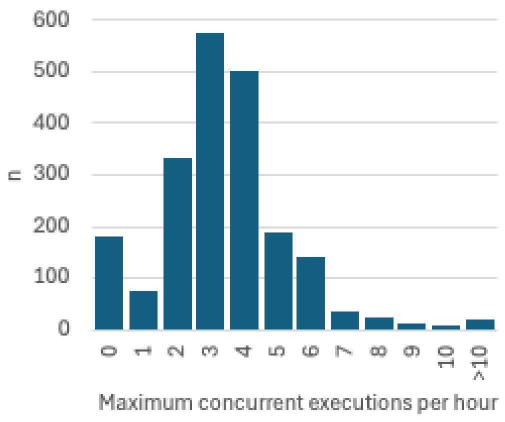
Service	**Technical summary**	**Originator**
`compareExpresions`	Second generation comparison of symbolic expressions. Uses SymPy, mostly used for equality but also for syntax.	Created by a specialist researcher without access to the application and no software engineering skills. Forked from `symbolicEqual`.
`buckinghamPiTheorem`	Validates groups in dimensional analysis. Applies to fluid mechanics. Based on Lundengard et al (2023).	Taken by a researcher, from an algorithm used by a teacher in another platform, and generalised as a microservice.
`compareBoolean`	Parses logical expressions to compare truth tables. Uses SymPy and a custom parser. Applies to basic electronics and computing.	Created by a student partner who was assisting an academic.
`evaluateProof`	Feedback on mathematical proofs written in natural language. Uses LLMs.	Created independently by a group of mathematicians and later connected to [platform name]. See Section 3.3.
`GCSEenglish`	Feedback on essays for GCSE English (a UK qualification at age 16).	Created independently by other organisations and connected to [platform name]. See Section 3.4.
`eduVision`	Vision processing to check lab hardware such as electronics assembly.	Created independently by a third-party team. Still in development.

Table 5: Quantity of calls for the microservices

Service	**Number of calls**	**Proportion**
isSimilar	469k	32%
arrayEqual	254k	17%
`symbolicEqual`	235k	16%
`compareExpresions`	229k	16%
array`symbolicEqual`	128k	8.7%
comparePhysicalQuantities	103k	7.0%
isExactEqual	37k	2.5%
Other	6.2k	0.4%



(a)



(b)

Figure 6: Scale and scaling of the microservice `compareExpresions` for 10th Oct ‘24 to 24th Dec ’24 (2000 hours). Invocations (a) shows the number of calls, and concurrent executions (b) shows when multiple compute instances were required due to scale with demand. Concurrent executions implies a process had not finished before another one was required.

concurrent execution of the service to handle the traffic. Concurrent executions were often required, for example five or more concurrent executions were required on 430 occasions in three months (21% of time when split by the hour), and 9 or more were required on 27 occasions (1.3%). In contrast, there were 181 hours out of 2,000 (9%) where no executions were required.

The results on the scale and scalability illustrate microservices for pedagogical judgement at a scale much larger than a single class or cohort and illustrate the feasibility of the architecture advocated in this article. Hardware arrangements for the microservices – in this case AWS Lambda – scaled independently of the application, which is one of the arguments for microservices.

Time scales for development of microservices can be reported anecdotally. An illustrative example is the niche function `compareBoolean`, for logic in electronics, which was developed by a student partner over a few weeks during a summer placement. The service was called 1,044 times in the subsequent academic year, showing that it was used by a single cohort. The service is ready to be used on a larger scale – or to be updated or replaced in future. Statistical data on time scales for innovation across the ecosystem is not available due to the limited number of services currently available. A more qualitative picture can be gained from the results of case studies 2 and 3.

The predominance, in terms of usage, of services for symbolic mathematics and numerical answers reflects the origins of the platform as a STEM education platform, before the emergence of large language models. Within the remaining 3% of calls are pilots which are yet to be used on large scale but are pioneering new methods of feedback generation at a small scale. Those pilots are notable for fulfilling the needs articulated in the first part of the article, and in the next two sections we study those pilots in more detail. It is notable at this point, however, that the large scale use of `compareExpresions` is a validation of the microservice architecture in that the service was developed externally to the platform, on a shorter timescale, and introduced interoperably to replace the `symbolicEqual` function.

One key feature of the platform and microservice system that was deployed was the ‘no code’ paradigm, in which teacher-users select and configure microservices, but do not write the code. The parametric configuration of a service is part of the teacher agency in the pedagogical judgements that are made. This can be, for example, defining which symbols are equivalent. Such nuances are important. In the case of symbolic expressions in physical sciences, in one context two symbols can be equivalent (such as upper or lower case letters), but in other contexts the difference is important.

To summarise Case Study 1 on the deployment of the platform, it shows a diversity of usage requiring different pedagogical judgements. Some of the diversity is evident in the range of subjects that are studied on the platform, and others in the types of microservices developed and the specialist knowledge required (the next two case studies will expand on this point). It is also notable that the microservices use a different runtime (Python in this case, to make use of SymPy) to the platform (Typescript/Javacript due to the web application), and that the scalability, interoperability, transparency, accountability, and accessibility of the microservices was all due to the architecture employed. Finally, Case Study 1 shows that innovation in microservices was feasible for independent parties and with a low barrier to entry.

3.3. Case study 2: application to proof in undergraduate mathematics

The second case study is in the use of Large Language Models to evaluate mathematical proofs drafted by undergraduate students in Mathematics. In an introductory module on rigorous proof of theorems from axioms, students are inducted into a form of Mathematics that is novel to them after leaving school.

Students require feedback that relies on pedagogical judgement. Feedback answers the question “Is my answer at the right level of rigour?”. Judgements on the ‘level of rigour’ are contextual and cultural.

Feedback is required on correctness, on reasoning and communication with standards based on the university Mathematics context. It should also identify errors, clarify misconceptions and suggest improvements.

One way of providing feedback is to require formal proofs in a precise symbolic language, and to employ *theorem provers*, as pioneered by Buzzard (2022). The syntactic requirements of formal proofs present a challenge to novice mathematicians in addition to the new mathematical ideas they must master; students often perceive this syntactic challenge as less to do with mathematics than with programming (Iannone and Thoma, 2024). Further, symbolic-language proofs guarantee correctness but not necessarily explanatory power (Hanna and Yan, 2021).

The pedagogical judgement used in the case study here was to consider symbolic-language proving as a specialised field, and instead to focus on conventional mathematical proofs using a less strict, more readable syntax, and reasoning that is rigorous but flexible.

The sequence of events in this case study was that initially a project to develop automated feedback using ‘reasoning’ LLMs, such as OpenAI o3 and o4 series, was initiated at [University name] independently of [platform name]. When the two projects learned of each other, the LLM usage was connected to [platform name] as a microservice without needing modifications, facilitating deployment to large cohorts of students and timely delivery of the automated formative feedback.

To aid in evaluating the performance of the microservice, a grading step was included in the algorithm but hidden from students, allowing quantitative evaluation of the performance of the algorithm. The OpenAI o4-mini model was also used to assess *coverage and conciseness* of LLM feedback, by extracting comments from human markers and the LLM, and listing what appear to be essential elements of the feedback; this list was then checked by a human expert. For each piece of feedback, the Jaccard index, a measure of how precisely the actual feedback matches the essentials, was then calculated.

Formative feedback was deployed in a module for 295 first year students, of whom 49 chose the option to trial the formative feedback. The number of submissions per student varied by question, but was approximately double the number of students depending on the question, suggesting that feedback commonly led to immediate action. These results show that the system was successfully deployed with the use of LLM-driven feedback on student attempts at proofs.

To illustrate results of from this pilot we use an exercise on the topic of *Number Systems* is reproduced in Box 1.

Box 1: Example maths question

Show that every nonzero element in $\mathbb{N}$ has a unique predecessor in $\mathbb{N}$ assuming only the Peano axioms and the definition of a predecessor: for $x, y \in \mathbb{N}$, x is a predecessor of y if $S(x) = y$.

Without needing an understanding of the mathematics the reader can see that the students is not asked for a mathematical expression but a sustained logical argument (in contrast with the type of task considered by the service `compareExpresions`). An example student response is given in Box 2, which was given feedback by `evaluateProof` reproduced in Box 3.

Box 2: Example student response

Let y be an arbitrary natural number such that y is not equal to 0. We want to prove that there exists a unique natural number x such that S(x)=y. Because of the map S from P2, there exists a predecessor x for y from S(x)=y. It must also be unique because of P4, as for two distinct y1 and y2 in the natural numbers, y1=S(x1) not equal to y2=S(x2) implies that x1 is not equal to x2.

Box 3: Example maths feedback

Your solution demonstrates a good understanding of the Peano axioms, particularly in identifying and applying the relevant axioms to the problem. Here are some points for improvement:

1. **Existence of Predecessor**: You correctly mention that the map S from P2 implies the existence of a predecessor for y. However, it would be beneficial

to explicitly state that since $y \neq 0$, by P3, there must exist some x such that $S(x) = y$.

2. **Uniqueness of Predecessor**: Your explanation of uniqueness using P4 is mostly correct. You correctly identify that the injectivity of S ensures that if $S(x_1) = y$ and $S(x_2) = y$, then $x_1 = x_2$. However, your explanation could be clearer by explicitly stating that for any y, if there were two predecessors x_1 and x_2, then $S(x_1) = S(x_2) = y$ would imply $x_1 = x_2$ due to injectivity.

3. **Structure and Clarity**: Your argument is generally well-structured, but it could benefit from a more explicit step-by-step approach. Clearly delineating the existence and uniqueness parts of your argument would enhance clarity.

Overall, your solution is on the right track, but adding more explicit reasoning and structure would strengthen your argument.

Without requiring mathematical knowledge, the reader can observe that the fine-grained feedback reflects domain-specific expertise, which was enabled by the use of an LLM, but also was curated by expert instructors — only possible by the low barrier to entry in the microservice architecture.

The feedback contains key pedagogical judgement, aligning with the educational context, requiring assumptions and deductions to be explicit. These are the relative levels of rigour that are required — more than the student offered, but less than a formal symbolic language.

Alignment between the pedagogical judgements shown by the LLM and those of human experts was not perfect. Comparison of LLM feedback with blind contributions by human experts showed that the LLM, despite attempts otherwise in the prompting structure, were typically too long and less readable. Human experts produced shorter, more targeted feedback. Jaccard scores comparing the feedback showed that LLM feedback sometimes erred towards the non-essential. This critical analysis is essential to making progress in the use of LLMs for automated formative feedback.

The distribution of hidden marks from the LLM, compared to expert humans, showed a tendency of the LLM to occupy a middle ground; it was rare to receive robust criticism in LLM feedback, but also rare for LLM feedback to judge a response as complete. In contrast human experts identified whether or not the proof was considered ‘complete’.

The critical analysis on the length and relevance of feedback, and inability of the LLM to commit to a position, are key research insights. The separation of concerns to an externally developed microservice enables transparency of evaluation and sharing these research insights. Responses have been on the time scale of weeks or months, with projects in July 2024 informing deployments in September 2024; and again in July 2025 informing September 2025. This experience supports the claim that microservices facilitate rapid innovation.

The pilot development and deployment of `evaluateProof` required significant resource that not all institution can commit. The public code, and platform-independent microservice deployment, increases the accessibility and improves equity of access.

3.4. Case study 3: application to English language in a school

The third case study is in the use of Large Language Models to provide feedback on essay drafts written by children in a UK school for a national English Language qualification (GCSE) for students aged 14 to 16. This case study was a three-way collaboration between the [platform name], a school, and an exam board.

The drive to provide automated formative feedback came from teachers at a school looking to capitalise, in 2024, on the emerging capabilities of LLMs. Interest from the exam board, [X], arose from the desire to provide constructive formative feedback to schools that would aid in preparation for exams provided by the board. The collaboration worked by the school providing manually created exemplar feedback, and the exam board iteratively developing prompts to provide feedback. The school deployed the system to their students using [platform name], and selected the microservice curated by the exam board. In depth research on the value of the feedback, the student experience, and the teacher views, were gathered through surveys and focus groups.

One example assessment question was provided to all participants:

> Your local library is running a creative writing competition. The best entries will be published in a booklet of creative writing. Write a description of a mysterious place, as suggested by this picture [accompanied by a figure] (REDACTED, 2020)

Feedback was split into 'technical accuracy', and 'content and organisation', and broken down into praise, targets, and encouragement. It's notable that the decision to structure feedback in this way is a pedagogical judgement that we would not expect a software company to be competent to make. The judgement is made by the experts and applied in the context.

Prompting used the OpenAI o4-mini (as of January 2025) and was developed through iteration, including feedback from school teachers.

Deployment was to 121 students in years 9 and 10 (ages 13-14 and 14-15), in classes of 10. A survey of students and a focus group with four teachers took place in January 2025.

Ethical use of the pilot for research was managed by the internal review board of [X].

In this case study a collaboration between teachers and exam boards iteratively improved and then deployed LLM-based feedback to school students on their essays in English language, and was deployed to two cohorts of students that received automated formative feedback. In total 117 students submitted an essay, all of whom also completed a survey which showed that 75% found the feedback helpful and 73% understood everything in the feedback.

An example feedback message is given in Appendix A. The fact that teachers and an exam board independently developed their feedback approach is evidence that the barrier to entry to innovating is very low. The contrast of the expertise required to give feedback to school-level students in English language, compared to university-level mathematical proofs in Case Study 2, highlights how microservices enable experts in a diverse range of niche areas to contribute to automating pedagogical judgements.

Similar to Case Study 2, the use of LLMs in Case Study 3 shows the adaptability of microservices to employ new technologies; in contrast, the platform itself did not change significantly during the innovation, nor was it originally designed for LLMs.

Many elements of the feedback in Appendix A require pedagogical judgements. A typical example is that a student's phrases:

> "paint a beautiful picture in the reader's mind"

The adjective 'beautiful', and the inference of what is in the mind of the reader, are subjective. These pedagogical judgements are contextual and contestable, justifying the focus on the ethics of automation.

The pilot development was a research project led by [X] which critically analysed the feedback. Critical analysis after the pilot showed, for example, that teachers praised the level of detail that was possible, relative to manual feedback. However, teachers also criticised the length of responses relative to input by the student, and misjudgements about the most important aspects of the essay to give feedback on. Teachers suggested replicating their tone, personalising feedback based on student abilities, and alternative formats to support the diversity of needs amongst students. This type of critical analysis promotes improvements to the feedback, but requires a level of transparency – including identifying weaknesses of the microservice – that is feasible when the developers of the microservice are not invested in the platform that delivers it. The point here is that the pilot showed evidence that transparency is improved when the microservice is developed independently of the platform.

The algorithms used to produce the feedback are publicly available. Other organisations or platforms can call the microservice, or reuse the code, making it accessible widely in the sector – including those organisations without the resource to conduct the research and development.

3.5. Summary of results

Results were presented in three case studies. We can summarise the results as providing evidence for the following:

- Lower barriers to entry by reducing services to the most granular level
- Decoupled services led to a diversity of experts contributing in different niche areas
- Technology independence allowed each service to adapt to the problem at hand and to adopt new technologies, such as computer algebra systems (CAS) and LLMs.

We showed that microservices foster a more ethical ecosystem by separating services from platforms:

- Educator choice and agency to change pedagogical judgements without changing the platform
- Incentives for transparency in the performance of the services
- Promote equity of access and higher quality by reducing barriers to sharing resources and increasing the number of platforms that can call a service.

4. Discussion

The evidence presented in Section 3 is an exemplar of a platform that delegates all automated pedagogical judgement to external microservices. This novel approach supports the framework in Section 2 identifying the technical and ethical benefits of a microservice ecosystem. The scale of deployment, diversity of pedagogical applications, and architectural commitment to external microservices are novel and provide a new direction for the education sector to foster ethical and innovative use of AI in pedagogical judgment.

Part of the argument for granular microservices is the low-friction substitutability of the services. We provided one such example in practice in Section 3, and we expect more such substitutions as the ecosystem grows. When those cases materialise, they will provide further validation of a microservice framework.

Open testing of microservices is key to the transparency and agency aspects of our ethical arguments. The examples provided in Section 3 are all open source, and we provided specific examples of open, critical testing and analyses of the exemplar microservices (`GCSEenglish` and `evaluateProof`). While this evidence supports our ethical arguments, the community will need further evidence that transparency is enhanced by using external microservices in the ecosystem. To enable that evidence on a broad scale, the sector needs to agree on the criteria with which automated pedagogical judgements are evaluated. We recommend collaborative research to that end, to provide guidance to innovators on transparent and critical testing of services.

In Section 2 we categorised the education sector as high-n multiplatform. The microservices presented in Sections 3-**??** are available for any platform, while the evidence presented here is limited to a single platform. The full benefits of the low cost of multi-homing across high-n platforms will be realised once multiple platforms exist to call the same microservices. Such an expansion will be an important next step in the development of a microservice ecosystem.

When multiple platforms use the same microservices, standards for service APIs will be required. The research presented here motivates the development of API standards. We recommend that the community collaborates on common approaches to microservice API definitions, most likely using HTTP and a RESTful API.

Our critical review of microservice frameworks in Section 2 identified both benefits and risks of microservices. The evidence in Section **??** supports the benefits of microservices, but we must also consider the risks.

Key risks in microservices include the complexity of managing different services, the friction that can emerge in deploying system-wide changes, and the reliability issues that can result from such a decentralised architecture. At the scale of the exemplar reported in this paper, such issues were negligible and managed through conventional measures such as unit testing, automated deployments, and graceful error management. As the ecosystem grows these risks will grow and will need more attention.

Another key risk of microservices is in data sovereignty and security. The case study of formative feedback, presented here, reduced the risk level by exchanging granular and anonymous data. As the range of automated pedagogic judgement by microservice grows, the ethics of data sharing will become more complex. It remains to be seen at what point the extent of contextual data that is required by the microservice will cross the philosophical line between a granular 'service' and more comprehensive 'application'. This issue will come in to focus as the ecosystem develops.

The research presented here provides a foundation for the education sector to sustainably innovate in the use of AI microservices for automated pedagogical judgements. Developments on this foundation that will realise the potential of this approach, and are recommended as future research, are:

- Further development, deployment, and transparent testing of new niche microservices in formative feedback
- Exploration of other pedagogical judgements that can be automated by external microservices, such as content generation, summative assessment, learning analytics, and general/high-level study support.
- Community agreement on the criteria by which automated pedagogical judgements should be evaluated, including testing methods and data sources for testing
- Redevelopment or new development of LMS platforms to call external microservices, to leverage the microservices that will be developed
- Critical evaluation of the educational impact of these technologies in the context of the framework we provide, namely on greater innovation, wider and more equitable deployment, improved transparency and agency, and impact on student experience and learning.

5. Conclusion

Microservices enable technical innovation and ethical behaviour in the development of automated pedagogical judgements. Technical innovation is facilitated by lowering the barrier to entry to the smallest possible task size, decoupling services to invite the diversity of expertise needed in all the niche areas, and granting technology independence to adapt to the varying applications and changing technologies. Ethical behaviour is enhanced by educator agency in service choice, incentives for transparency on the performance of the services, and facilitating highly interoperable and scalable deployments that promote equity of access and higher quality.

The paper provides an overall framework for the sector to follow, based on a theoretical perspective on software architecture and ethical discussions around AI in education. Evidence to support these arguments comes from our exemplar deployment of a new learning management system (LMS) for automated formative feedback, in which all pedagogical judgements are sourced from external microservices. The scale of deployment to thousands of monthly users, across a diversity of subject areas, multiple institutions, and with one million annual service calls, shows the feasibility of such a system.

Further validation of the theoretical arguments for microservices is provided in the three case studies we presented. The first case study showed the scale of use of microservices, including replacing a service interoperably with an upgraded service, and scaling a service to thousands of invocations per hour more than 20% of the time. The independent development, deployment, and scaling validate the argument to separate microservices. Our examples included a microservices using a computer algebra system in Python, which was independent of the Javascript-based platform that called this external service.

In the second case study we showed examples of the use of AI to provide feedback on mathematical proofs at university level, illustrating the nuances of the pedagogical judgement involved — in this case how much rigour is, or isn't, required. These examples show the level of expertise required to develop the automation algorithms, the feasibility of completely independent practitioners developing microservices for granular functions, and the transparency with which such independent services can be critically analysed.

The third case study, using AI to provide feedback on English language essays in schools, showed how two organisations, completely independent from each other and from the platform that was used for deployment, could collaborate on the development of a microservice that requires pedagogical judgement and a very different expertise to the mathematics case study. The evidence of rapid and independent innovation, and accountability for the pedagogic judgements, were clearly evident in this case study too. All of the microservices presented here were open source and therefore available both as source code and as a microservice available to other platforms or users.

The study presented here shows how microservices can facilitate innovation and an ethical approach to automated pedagogical judgements and we recommend this approach to the community. Future developments should focus on multi-platform usage, wider development of services, community engagement on testing criteria for transparent evaluations, and agreement on the necessary standards for APIs that will work across the ecosystem.

Acknowledgements

[Redacted for blind review]

References

Airasian, P.W., 1997. Classroom assessment. ERIC.

Allen, R., Benhenda, A., Jerrim, J., Sims, S., 2021. New evidence on teachers' working hours in england. an empirical analysis of four datasets. Research Papers in Education 36, 657–681. doi:10.1080/02671522.2020.1736616.

Baldwin, C.Y., 2018. Design rules, volume 2: how technology shapes organizations. Harvard Business School Research Paper Series 19, 042.

Baldwin, C.Y., Clark, K.B., 2000. Design rules, Volume 1: The power of modularity. MIT press.

Bass, L., 2012. Software architecture in practice. Pearson Education India.

Black, P., Wiliam, D., 1998. Assessment and classroom learning. Assessment in Education: Principles, Policy & Practice 5, 7–74. doi:10.1080/0969595980050102.

Bröring, A., Schmid, S., Schindhelm, C.K., Khelil, A., Käbisch, S., Kramer, D., Le Phuoc, D., Mitic, J., Anicic, D., Teniente, E., 2017. Enabling iot ecosystems through platform interoperability. IEEE Software 34, 54–61. doi:10.1109/ms.2017.2.

Buckingham Shum, S., Lim, L.A., Boud, D., Bearman, M., Dawson, P., 2023. A comparative analysis of the skilled use of automated feedback tools through the lens of teacher feedback literacy. International Journal of Educational Technology in Higher Education 20, 40. doi:10.1186/s41239-023-00410-9.

Burgess, S., Rawal, S., Taylor, E.S., 2023. Teachers' use of class time and student achievement. Economics of Education Review 94, 102405. doi:10.1016/j.econedurev.2023.102405.

Butt, G., Lance, A., 2005. Secondary teacher workload and job satisfaction: Do successful strategies for change exist? Educational Management Administration & Leadership 33, 401–422. doi:10.1177/1741143205056304.

Buzzard, K., 2022. URL: `https://talmo.uk/2022/slides/Buzzard.pdf`. accessed 29th July 2025.

Deeva, G., Bogdanova, D., Serral, E., Snoeck, M., De Weerdt, J., 2021. A review of automated feedback systems for learners: Classification framework, challenges and opportunities. Computers & Education 162, 104094. doi:10.1016/j.compedu.2020.104094.

Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R., Safina, L., 2017. Microservices: Yesterday, Today, and Tomorrow. Springer International Publishing, Cham. pp. 195–216. doi:10.1007/978-3-319-67425-4_12.

Fu, Y., Weng, Z., 2024. Navigating the ethical terrain of ai in education: A systematic review on framing responsible human-centered ai practices. Computers and Education: Artificial Intelligence 7, 100306.

Fullan, M., Langworthy, M., 2014. A rich seam: How new pedagogies find deep learning .

Garlan, D., Shaw, M., et al., 1993. An introduction to software architecture. Advances in software engineering and knowledge engineering 1.

Goedicke, M., Striewe, M., Balz, M., 2008. Computer aided assessments and programming exercises with JACK. Technical Report. ICB-Research Report.

Gottschalk, F., Weise, C., 2023. Digital equity and inclusion in education: An overview of practice and policy in oecd countries. URL: `https://www.proquest.com/working-papers/digital-equity-inclusion-education-overview/docview/2849362537/se-2`.

Graesser, A.C., Conley, M.W., Olney, A., 2012. Intelligent tutoring systems. APA educational psychology handbook, Vol 3: Application to learning and teaching. , 451–473doi:10.1037/13275-018.

Hanna, G., Yan, X.K., 2021. Opening a discussion on teaching proof with automated theorem provers. For the Learning of Mathematics 41, pp. 42–46. URL: `https://www.jstor.org/stable/27091220`.

Hattie, J., 2009. Visible Learning: A Synthesis of Over 800 Meta-analyses Relating to Achievement. Routledge.

Hattie, J., Timperley, H., 2007. The power of feedback. Review of educational research 77, 81–112.

Heffernan, N.T., Heffernan, C.L., 2014. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. International Journal of Artificial Intelligence in Education 24, 470–497. doi:10.1007/s40593-014-0024-x.

Henderson, M., Ryan, T., Phillips, M., 2019. The challenges of feedback in higher education. Assessment & Evaluation in Higher Education 44, 1237–1252. doi:10.1080/02602938.2019.1599815.

Hollingsworth, J., 1960. Automatic graders for programming classes. Communications of the ACM 3, 528–529. doi:10.1145/367415.367422.

Holmes, W., Porayska-Pomsta, K., Holstein, K., Sutherland, E., Baker, T., Shum, S.B., Santos, O.C., Rodrigo, M.T., Cukurova, M., Bittencourt, I.I., Koedinger, K.R., 2022. Ethics of ai in education: Towards a community-wide framework. International Journal of Artificial Intelligence in Education 32, 504–526. doi:10.1007/s40593-021-00239-1.

Hounshell, D., 1984. From the American system to mass production, 1800-1932: The development of manufacturing technology in the United States. 4, Jhu Press.

Iannone, P., Thoma, A., 2024. Interactive theorem provers for university mathematics: an exploratory study of students' perceptions. International Journal of Mathematical Education in Science and Technology 55, 2622–2644.

Jacobides, M.G., Cennamo, C., Gawer, A., 2018. Towards a theory of ecosystems. Strategic Management Journal 39, 2255–2276. doi:10.1002/smj.2904.

Johnson, P., Fenton, J., Ramsden, P., Chatley, R., Ribera-Vicent, M., Karl, L., 2025. Formative feedback on engineering self-study: Towards 1 million time per year per cohort, in: 2025 IEEE Global Engineering Education Conference (EDUCON).

Jurenka, I., Kunesch, M., McKee, K.R., Gillick, D., Zhu, S., Wiltberger, S., Phal, S.M., Hermann, K., Kasenberg, D., Bhoopchand, A., et al., 2024. Towards responsible development of generative ai for education: An evaluation-driven approach. arXiv preprint arXiv:2407.12687 .

Klimova, B., Pikhart, M., Kacetl, J., 2023. Ethical issues of the use of ai-driven mobile apps for education. Frontiers in Public Health Volume 10 - 2022. doi:10.3389/fpubh.2022.1118116.

Krusche, S., Seitz, A., 2018. Artemis: An automatic assessment management system for interactive learning, in: Proceedings of the 49th ACM technical symposium on computer science education, pp. 284–289.

Kulik, J.A., Fletcher, J.D., 2016. Effectiveness of intelligent tutoring systems: a meta-analytic review. Review of educational research 86, 42–78.

Labadze, L., Grigolia, M., Machaidze, L., 2023. Role of ai chatbots in education: systematic literature review. International Journal of Educational Technology in Higher Education 20, 56.

Lelièvre, M., Waldock, A., Liu, M., Aspillaga, N.V., Mackintosh, A., Portelo, M.J.O., Lee, J., Atherton, P., Ince, R.A., Garrod, O.G., 2025. Benchmarking the pedagogical knowledge of large language models. arXiv preprint arXiv:2506.18710 .

Lewis, J., Fowler, M., 2014. a definition of this new architectural term.

Ma, W., Adesope, O.O., Nesbit, J.C., Liu, Q., 2014. Intelligent tutoring systems and learning outcomes: A meta-analysis. Journal of educational psychology 106, 901.

Merrimack College, 2023. Merrimack college teacher survey 2023. URL: `https://www.merrimack.edu/academics/education-and-social-policy/about-old/merrimack-college-teacher-survey/`. accessed 29th July 2025.

Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I.D., Gebru, T., 2019. Model cards for model reporting, in: Proceedings of the Conference on Fairness, Accountability, and Transparency, Association for Computing Machinery, New York, NY, USA. p. 220–229. doi:10.1145/3287560.3287596.

Nunes, A., Cordeiro, C., Limpo, T., Castro, S.L., 2022. Effectiveness of automated writing evaluation systems in school settings: A systematic review of studies from 2000 to 2020. Journal of Computer Assisted Learning 38, 599–620.

Omiyad Network, 2019. Scaling access & impact: Realizing the power of edtech. executive summary .

Otto, D., Kerres, M., 2022. Increasing sustainability in open learning: Prospects of a distributed learning ecosystem for open educational resources. Frontiers in Education Volume 7 - 2022. doi:10.3389/feduc.2022.866917.

Palfrey, J., Gasser, U., 2012. Interop: The promise and perils of highly interconnected systems. Basic Books.

Panadero, E., Lipnevich, A.A., 2022. A review of feedback models and typologies: Towards an integrative model of feedback elements. Educational Research Review 35, 100416. doi:https://doi.org/10.1016/j.edurev.2021.100416.

Pangrazio, L., Selwyn, N., Cumbo, B., 2023. A patchwork of platforms: mapping data infrastructures in schools. Learning, Media and Technology 48, 65–80. doi:10.1080/17439884.2022.2035395.

Parnas, D.L., 1972. On the criteria to be used in decomposing systems into modules. Commun. ACM 15, 1053–1058. doi:10.1145/361598.361623.

Perry, D.E., Wolf, A.L., 1992. Foundations for the study of software architecture. SIGSOFT Softw. Eng. Notes 17, 40–52. doi:10.1145/141874.141884.

REDACTED, 2020. GCSE English Language 8700/1: November 2020 Question Paper. `https://REDACTED`. Accessed August 1, 2025.

Reigeluth, C.M., Karnopp, J.R., et al., 2013. Reinventing schools: It's time to break the mold. Bloomsbury Publishing PLC.

Sadler, D.R., 1989. Formative assessment and the design of instructional systems. Instructional Science 18, 119–144. doi:10.1007/bf00117714.

Sangwin, C., 2013. Computer aided assessment of mathematics. OUP Oxford.

Sangwin, C.J., 2007. Assessing elementary algebra with stack. International Journal of Mathematical Education in Science and Technology 38, 987–1002. doi:10.1080/00207390601002906.

Selwyn, N., Hillman, T., Bergviken-Rensfeldt, A., Perrotta, C., 2023. Making sense of the digital automation of education. Postdigital Science and Education 5, 1–14. doi:10.1007/s42438-022-00362-9.

Shetye, S., 2024. An evaluation of khanmigo, a generative ai tool, as a computer-assisted language learning app. Studies in Applied Linguistics and TESOL 24.

Shute, V.J., 2008. Focus on formative feedback. Review of Educational Research 78, 153–189. doi:10.3102/0034654307313795.

Simon, H.A., 1962. The architecture of complexity. Proceedings of the American Philosophical Society 106, 467–482.

Sullivan, K.J., Griswold, W.G., Cai, Y., Hallen, B., 2001. The structure and value of modularity in software design. SIGSOFT Softw. Eng. Notes 26, 99–108. doi:10.1145/503271.503224.

Tejado, I., Nuevo-Gallardo, C., de la Encarnación-Sama, R., Pérez, E., Vinagre, B., 2023. Using matlab grader for formative feedback in engineering degrees, in: INTED2023 Proceedings, IATED. pp. 5563–5573.

Valenti, S., Neri, F., Cucchiarelli, A., 2003. An overview of current research on automated essay grading. Journal of Information Technology Education: Research 2, 319–330.

VanLehn, K., 2011. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educational Psychologist 46, 197–221. doi:10.1080/00461520.2011.611369.

Wang, S., Xu, T., Li, H., Zhang, C., Liang, J., Tang, J., Yu, P.S., Wen, Q., 2024. Large language models for education: A survey and outlook. URL: https://arxiv.org/abs/2403.18105.

Wang, Y., Kadiyala, H., Rubin, J., 2021. Promises and challenges of microservices: an exploratory study. Empirical Software Engineering 26, 63.

Whitworth, J., 1858. Miscellaneous papers on mechanical subjects. Longman, Brown, Green, Longmans, and Roberts.

Winstone, N.E., Boud, D., 2022. The need to disentangle assessment and feedback in higher education. Studies in Higher Education 47, 656–667. doi:10.1080/03075079.2020.1779687.

Zhang, X., Zhang, P., Shen, Y., Liu, M., Wang, Q., Gašević, D., Fan, Y., 2024. A systematic literature review of empirical research on applying generative artificial intelligence in education. Frontiers of Digital Education 1, 223–245. doi:10.1007/s44366-024-0028-5.

Appendix A. Exemplar GCSE English Feedback

Content Feedback

Praise on Ideas

You have created a vivid depiction of the Christmas scene, and your ability to convey the atmosphere is quite engaging. The imagery surrounding Santa Claus and the festive decorations offers the reader a taste of holiday cheer, effectively immersing them in the experience. You've also nicely incorporated personal reflection, allowing the reader a glimpse into the nostalgia and complexity of your feelings about Christmas.

Target on Cohesion

Have you considered ways to improve the transitions between your paragraphs? Some connections feel abrupt, such as the shift from the festive decorations to your past visit. Perhaps you could add a linking sentence that highlights how time has changed the town and your memories, such as: "Reflecting on my last visit, I could hardly recognize the bustling town that had once been so tranquil." This could create a smoother transition, enhancing the flow of your description.

Technical Accuracy

Praise on Vocabulary

Your choice of vocabulary is quite expressive, particularly phrases like "delicate flake drifts silently from the heavens," which paint a beautiful picture in the reader's mind. The use of adjectives such as "gorgeous" and "precious" contributes positively to the mood you're trying to create, and pulling in sensory elements like taste and sight is a great way for readers to connect with your writing.

Target on Spelling

There are a few spelling mistakes that need attention, specifically "Santa clause" which should be "Santa Claus" and "coazy" should be corrected to "cozy." Additionally, "Chritmas" should read as "Christmas." These minor errors distract from your overall prose, so please be sure to proofread your work.

Final Thoughts

The piece feels more narrative than strictly descriptive, shifting between personal experience and festive observations. Instead of solely recounting events, consider focusing more on the sensory elements of the scene, such as the smells of pine, the sound of laughter, and the coolness of the air. This shift can deepen your descriptive writing. Great job, keep writing!

Appendix B. Trade-offs when considering microservices

Table B.6: Trade-offs when considering a *microservice* architecture. Summarised from Lewis and Fowler (2014) and Dragoni et al. (2017).

Issue	**Benefits**	**Drawbacks**	**What is distinctive about microservices**
Team autonomy	Teams can develop, deploy, and scale services independently	Requires coordination of service boundaries and communication	Services align with team boundaries and deployment pipelines, not just code separation
System complexity	Enables decomposition of large systems into manageable parts	Adds distributed complexity, network overhead, data consistency challenges	Encourages independently deployed units with API contracts
Project maturity / size	Useful once system becomes too large for a single codebase	Overkill for small or early-stage systems	Designed for evolving large-scale systems, not early-stage prototypes
DevOps maturity	Enables continuous delivery and scalable infrastructure	Requires advanced automation, deployment, logging, and monitoring	Demands CI/CD, containerization, service discovery — not just modular code
Fault tolerance	Services can fail independently, system can degrade gracefully	Adds complexity in error handling, retries, timeouts, fallbacks	Failure isolation is enforced at runtime boundaries, not just design-time separation
Deployment flexibility	Individual services can be deployed and scaled independently	Requires orchestration and versioning across services	Deployment units are decoupled, unlike a modular monolith
Reuse and specialisation	Teams can own technology stacks suited to each service	Potential tech sprawl, operational complexity	Cross-team heterogeneity allowed (e.g. one service in Go, another in Node.js)
Performance / latency	May improve scalability via independent scaling	Remote calls introduce latency and increased surface area for failure	Services communicate over networks rather than function calls
Modularity vs. microservices	Microservices enforce modular boundaries through network isolation	True modularity doesn't require microservices	Microservices are a **deployment and runtime** pattern, not just a design or code organisation approach