Eduka

WHITE PAPER

IN-HOUSE DEVELOPMENT OR A PROVEN PLATFORM

Build it in-house, or choose Eduka?

15 reasons why building your own school management software, even with the help of artificial intelligence, is a false economy: the pitfalls and the hidden costs.

INTENDED FOR

Heads of school, finance and administration directors, IT managers and governing boards of educational institutions.

October 2026

EXECUTIVE SUMMARY

The essentials on one page

AI coding assistants give the impression that school management software can now be built in-house, in a few weeks, for the price of one salary. This overlooks the fact that writing code has never been the most expensive part of such software.

It is not cheaper

In-house, you alone fund 100% of development, maintenance, support, security and compliance. With a software vendor, these costs are shared across every institution.

The risk stays with you

Data breach, non-compliance, an outage on the first day of term: without a vendor, there is no contract, no insurance and no one to whom the liability can be transferred.

It does not last

An in-house tool depends on one or two people. The day they leave, it becomes technical debt that must be replaced, often at a higher cost than the original project.

The 15 reasons, from the most decisive to the most obvious

1 You pay 1/N, not 100%

2 The hidden costs of "free" software

3 The 80/20 trap: the demo is not the software

4 Security, a responsibility you carry alone

5 No one to transfer the legal risk to

6 Compliance is a full-time job

7 The day the developer leaves

8 You are creating a support desk for life

9 No one on your team is on call

10 Hosting, the forgotten question

11 Every integration is a project, and every integration breaks

12 An in-house tool will never be all-in-one

13 The exit trap

14 The asset you build becomes a liability

15 It is not your core business

ECONOMICS RISK AND COMPLIANCE OPERATIONS SUSTAINABILITY

INTRODUCTION

AI has changed the game… or so it seems

Since the arrival of AI coding assistants, one idea keeps coming up in leadership meetings: **why keep paying for a licence when our IT team, helped by AI, could build our own tool?** The demonstration is often convincing. In a single afternoon, a student management screen appears, it works, and it seems to prove that the hard part is done.

The intuition is understandable, and it stems from a sound intention: keeping costs under control. Yet it rests on a misconception. Writing code has never been the most expensive part of management software. What costs money is everything that comes afterwards: security, regulatory compliance, hosting, support for families and staff, service continuity, changing needs, and the tool's ability to survive the departure of the person who wrote it.

School management software is a critical system. It handles admissions, family invoicing and payments, attendance, assessments and school reports, timetables and communication with parents. It processes the personal data of minors. It must work flawlessly on dates the school does not choose: the first day of term, the end of the grading period, the admissions campaign.

This white paper reviews **fifteen pitfalls**, ranked from the most decisive to the most obvious. The point is not that an in-house IT team is incapable of doing good work; it is that, even when done well, in-house development costs more, exposes the institution further and lasts less long than a vendor solution such as Eduka. It ends with **six questions** to ask your team before making any decision.

5%

This is often the real progress of a project believed to be 80% complete after a successful demo (reason 3).

100%

This is the share of the costs you fund alone in-house, compared with a pooled fraction with a vendor (reason 1).

0

This is the number of parties you can hold liable if an incident occurs on an in-house tool (reason 5).

THE 15 REASONS

Why in-house software is a false economy

1

ECONOMICS

You pay 1/N, not 100%

This is the strongest economic argument, and it is structurally impossible to replicate in-house. With a vendor such as Eduka, the effort spent on development, security, compliance and regulatory monitoring is spread across all client institutions. **You fund only a fraction of it.**

Better still, you benefit from other schools' needs. A feature requested by a secondary school in Dubai, a regulatory change required by an institution in Singapore, an improvement suggested by a school in Paris: all of them reach you without you having to ask or pay for them. Your software is enriched every month by the experience of dozens of other leadership teams.

> In-house, you fund 100% of everything, for a single institution, and you never receive anything from outside.

2

ECONOMICS

The hidden costs of "free" software

An in-house tool has no licence invoice. That does not make it free. The time of the developer, or the team, who designs, tests, fixes and evolves it is a very real payroll cost, to which you must add employer contributions, recruitment, training and management.

Then there is the **opportunity cost**: every hour spent rebuilding software that already exists is an hour not spent on the digital projects that are specific to your institution, the ones that truly set you apart.

The right comparison is therefore not "a licence versus zero", but "a licence versus the payroll devoted to the tool, every year, with no end date". Put that way, the equation almost always favours existing software, proven and built by professionals. The grid on the page "The full cost, item by item" lets you make this comparison properly.

THE QUESTION TO ASK

Is the annual fully loaded cost of the people who will build and maintain the tool lower than the cost of the licence? And will it still be in five years' time?

3

ECONOMICS

The 80/20 trap: the demo is not the software

You have seen a prototype in action, and all that supposedly remains is "the finishing touches". That is precisely where most of the work lies. Management watched AI produce, in two hours, a working student management screen; they conclude that the project is 80% done. **It is 5% done.** What is missing never shows up in a demo:

- access rights by profile: leadership, school office, finance, teachers, parents, students;
- migration of previous years' history;
- edge cases: blended families, shared custody, siblings, class changes during the year, discounts and payment plans;
- performance under load: 1,200 families logged in at the same time on the evening school reports are published;
- database changes without any loss of information;
- backups that can actually be restored;
- compliance, documentation, training and support.

> AI excels at producing plausible code. School management software does not need to be plausible: it needs to be exact on 30 June at 5 pm.

4

RISK AND COMPLIANCE

Security, a responsibility you carry alone

School software concentrates some of the most sensitive data there is: the identity and address of minors, family circumstances, health information, academic results, families' bank details. It is a prime target.

By its very nature, a tool developed in-house offers fewer guarantees than software designed, tested and monitored by security professionals. Has it been audited by a third party? Has it undergone a penetration test? Are security patches for the components it relies on applied as they are released? Who monitors intrusion attempts? And the threat landscape is getting tougher: the AI tools that help write code help attackers find its flaws just as much.

As a decision-maker, this responsibility is yours. In the event of a breach, families will not remember that it was a reasonable budget decision. They will remember that their data was exposed for a saving which, compared with the institution's budget, will look trivial.

THE QUESTION TO ASK

Who audited the code, when, and where is the report?

5

RISK AND COMPLIANCE

No one to transfer the legal risk to

This is a legal argument, tailor-made for a finance director, and one that is almost never put forward.

When you work with a vendor, you sign a contract: a data processing agreement within the meaning of Article 28 of the GDPR, service commitments, a named contact, and a provider covered by its own professional indemnity insurance. Part of the risk is thereby transferred, both contractually and through insurance.

With an in-house tool, there is no counterparty. The institution is the data controller and carries out the processing itself. You cannot sue yourself, nor claim on an insurance policy against your own team. If an incident occurs, the entire liability, financial and sometimes personal, remains within the institution.

> A licence is not just software. It is a partner who can be held accountable.

6

RISK AND COMPLIANCE

Compliance is a full-time job

A school is subject to the data protection regulations of every country in which it operates, and sometimes to those of the country where its families live. Depending on your locations, these may include:

GDPR
European Union and European Economic Area, and any institution that processes the data of European residents.

PDPA
Singapore: consent, purpose limitation, breach notification, appointment of a data protection officer.

PDPL
United Arab Emirates: the federal personal data protection law, with its own rules on transfers outside the country.

National laws
Many countries in Africa, Asia and the Middle East have their own law and their own supervisory authority.

Behind these texts lie very concrete obligations, which the tool must make it possible to meet:

- keeping a **record of processing activities** and, for minors' data processed on a large scale, carrying out a **data protection impact assessment**;
- responding to **data subject rights** (access, rectification, erasure, portability) within one month under the GDPR;
- **notifying a data breach** to the supervisory authority within 72 hours;
- applying **retention periods** and actually deleting expired data;
- controlling **where data is hosted** and regulating transfers outside the protected area;
- ensuring security "appropriate to the risk" and **logging access** to the data.

These texts change, and authorities regularly publish new guidance. Every change must be analysed, turned into features, developed, tested and then deployed. Does your developer have the time, and the legal expertise, for this? For a vendor serving institutions in several countries, this monitoring is ongoing work, pooled across all clients (reason 1).

THE QUESTION TO ASK

Can you produce, today, the record of processing activities and the impact assessment for the tool?

7

SUSTAINABILITY

The day the developer leaves

A significant share of the institutions that join Eduka had built their own tool. Their story is almost always the same: the developer who created it left, nobody else understands the code, and the tool can no longer be maintained. Every fix becomes risky, every improvement impossible.

An in-house tool is, first and foremost, one person's knowledge. As long as they are there, everything works. The day they change jobs, countries or careers, the institution discovers that it depends on a system nobody controls any more. In the long run, it is a project almost always doomed to fail: you need to plan its replacement before you have even started it.

THE QUESTION TO ASK

If your developer resigns tomorrow morning, how many weeks will it take a replacement to become self-sufficient on their code?

8

OPERATIONS

You are not hiring a developer: you are creating a support desk for life

On go-live day, your developer stops being a developer. They become the helpdesk for 800 families who cannot log in the day before the payment deadline, for teachers who cannot find where to enter marks, for administrative staff who change every year and need training all over again.

This cost appears in no project budget, and it is permanent. With a vendor, it is organised, equipped, pooled and included: a support team, a knowledge base, online help, training.

9

OPERATIONS

No one on your team is on call

Results day, invoicing night, admissions week: what happens if the only person who knows the tool is on holiday, on sick leave, or resigned the week before? A school administration cannot be put on pause.

A vendor has a team, a testing process, separate environments for development, acceptance testing and production, and an organisation designed for service continuity. A lone developer deploys to production on a Friday evening, and keeps their fingers crossed.

THE QUESTION TO ASK

Who fixes the tool at 7 am on 2 September if the timetables do not display?

10

OPERATIONS

Hosting, the forgotten question

Where will the tool run? The question is rarely asked at the outset, and it changes everything.

On a server at the school? Then you have to manage the hardware, the firewall, system updates, off-site backups, the power supply, availability from outside for families, and interventions at weekends or during the holidays.

In the cloud? Then you have to budget for a recurring cost, check that the hosting provider and the data location comply with your obligations, and have the skills to secure an infrastructure exposed to the internet, at a time when automated attack tools keep improving. You will also need to have that security audited.

Between hosting, backups, monitoring and audits, the bill rises sharply, and it recurs every year. With Eduka, all of this is part of the service.

THE QUESTION TO ASK

When was the last **restore** test of the backups? Not the backup: the restore.

11

OPERATIONS

Every integration is a project, and every integration breaks

Accounting, payment provider, directory, single sign-on, learning platform, mobile app, email: school management software lives among many other services. Every connector developed in-house is a project in its own right, and each one can break silently the day the remote service changes its interface.

Who is watching? Nobody, until the day you discover that families' payments have not reached the accounting system for three weeks. With a vendor, connectors are maintained for all clients, and a breakage is detected and fixed once, for everyone.

12

OPERATIONS

An in-house tool will never be all-in-one

A school's scope is immense: admissions and re-enrolment, invoicing and payments, attendance, assessments and school reports, timetables, classroom life, communication with families, canteen, extra-curricular activities, transportation, school trips, staff data. An in-house tool usually covers a small part of it. The rest is handled by other software.

You then have to build and monitor synchronisations, enter the same information twice, reconcile data that drifts apart. And for families, it means as many usernames, passwords and different apps. Parents are the first to suffer, and the school's image suffers with them. A truly all-in-one solution such as Eduka shares a single database and a single login across all these areas.

13

SUSTAINABILITY

The exit trap: in-house tools never plan their own end

When the time comes to replace it, and it will, you will have to extract ten years of history from an undocumented database, with no standard export, and business rules buried in the code. This undertaking regularly costs more than the original project, and it always comes at the worst possible moment.

Eduka, on the other hand, knows how to export your data: reversibility is part of the contract.

THE QUESTION TO ASK

A parent exercises their right of access or erasure: how long does it take to respond, and who handles it?

14

SUSTAINABILITY

The asset you build becomes a liability

For a finance director, this is balance-sheet reasoning. An in-house tool cannot be sold, cannot be audited by a third party and cannot be valued. It has no resale value and is hard to hand over. Its value is close to zero on the day it is delivered, and it becomes a liability, technical debt, as soon as the person who built it is no longer there.

A licence is an operating expense: predictable, budgetable, comparable, and **terminable**.

> In-house development is an irreversible commitment disguised as a saving.

15

SUSTAINABILITY

It is not your core business, and you already know it

This is the closing argument, the simplest and the hardest to refute. You do not develop your own payroll software. You do not develop your own accounting software. Why not? Because regulations change, because mistakes are very costly, and because it is not your core business.

School management software is invoicing, plus accounting, plus sensitive data about minors, plus a non-negotiable calendar, plus hundreds of users who are not IT specialists. It is the **worst** candidate for in-house development, not the best.

DECISION-MAKING TOOL

The full cost, item by item

Comparing a licence with "zero" makes no sense. This grid lists the items an in-house project will have to fund, year after year, and how they are handled with Eduka.

Item	In-house development	With Eduka
Initial design and development	Team payroll for months, before anything goes live	✓ Already built and proven in production
Functional improvements	100% at your expense, at the pace of a single team	✓ Pooled across client institutions
Regulatory monitoring and compliance	To be organised and funded yourself, country by country	✓ Handled by the vendor for all its clients
Security, audits, penetration tests	To be commissioned, funded and followed up	✓ Handled by the vendor
Hosting, backups, monitoring	Infrastructure, skills and on-call cover to fund	✓ Included in the service
Support for families and staff	Your team, permanently, with no dedicated budget	✓ Support team, online help, training
Service continuity	Depends on one or two people	✓ A team, separate test environments
Integrations with other services	One project per connector, monitoring to organise	✓ Maintained for all clients
Functional coverage	Partial, supplemented by other tools to synchronise	✓ All-in-one platform, a single login for families
Contractual liability	No counterparty: everything stays with the institution	✓ Contract, data processing agreement, named contact
Exit and reversibility	Unplanned undertaking, often more expensive than the original project	✓ Data export provided for in the contract
Nature of the expense	Irreversible commitment, an asset with no resale value	✓ Predictable, terminable operating expense

To estimate your own situation, add up in the left-hand column the annual fully loaded cost of each person involved (pro rata to their time), hosting, audits and third-party tools over five years, then compare with the cost of the licence over the same period.

KEEP IT TO HAND

Six questions to ask your team before deciding

Decision-makers remember questions better than arguments. Ask these, and listen to the quality of the answers.

1. Who fixes the tool at 7 am on 2 September if the timetables do not display? ☐
   Service continuity (reason 9)
2. When was the last restore test of the backups? ☐
   Not the backup: the restore (reason 10)
3. Can you produce, today, the record of processing activities and the impact assessment for the tool? ☐
   Compliance (reason 6)
4. A parent exercises their right of access or erasure: how long does it take to respond, and who handles it? ☐
   Data subject rights (reasons 6 and 13)
5. Who audited the code, when, and where is the report? ☐
   Security (reason 4)
6. If your developer resigns tomorrow morning, how many weeks before a replacement is self-sufficient on their code? ☐
   Sustainability (reason 7)

If even one of these answers is hesitant, **the in-house project is already in debt**, before it has even begun.

Let's talk about your project

Your job is to teach and help your students succeed. Ours is to give you the tools to do it.

Before committing your institution to in-house development, take the time to compare. Tell us about your needs: we will show you how Eduka meets them, and together we will examine the full cost of both options.

1

An initial conversation to understand your institution and your priorities.

2

A demonstration built around your own use cases.

3

An honest, item-by-item comparison with your in-house project.

Eduka

CONTACT US

www.edukasoftware.com