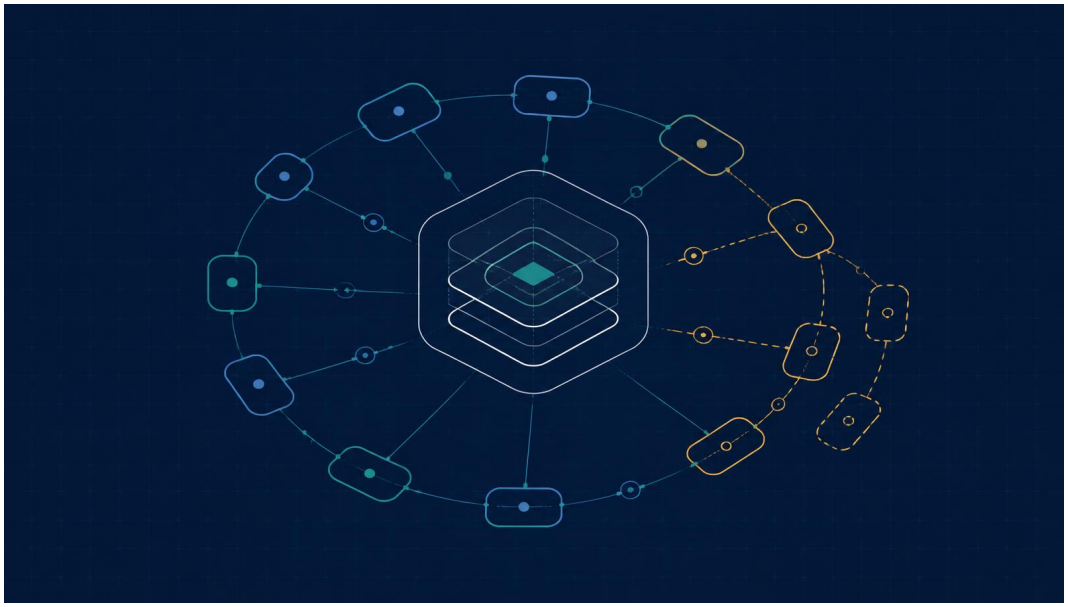
Executive Guide

Your AI Application Probably Won’t Fail Because of a Bug

The hidden cost of treating AI support as an afterthought



Prepared by

Vikram Katyani

Founder, IntelliMinds Digital

Executive Summary

AI applications don't stop working only because of bugs. Explore why changing dependencies, platform updates and operational ownership make ongoing AI support essential.

There’s a common assumption that once an AI application has been built, tested and deployed, the hard work is done.

It isn’t.

Recently, we were asked to investigate an AI-powered application that had started experiencing intermittent failures in production.

Contents

1.	The wrong question
2.	Fixing the problem wasn’t the expensive part
3.	This is where support becomes much cheaper than people think
4.	The people who built it shouldn’t have to rediscover it
5.	From project to product
6.	What ongoing AI support actually means
7.	There’s a hidden cost to reactive support
8.	Production isn’t the finish line
9.	The cheapest time to support an AI application is before it breaks

There's a common assumption that once an AI application has been built, tested and deployed, the hard work is done.

It isn't.

Recently, we were asked to investigate an AI-powered application that had started experiencing intermittent failures in production.

At first, it looked like a bug.

The architecture was solid. The infrastructure was stable. The application had been running successfully.

But something had changed.

One of the cloud services the application depended on had changed, and the integration was no longer behaving as expected.

The application hadn't suddenly become badly written.

The environment around it had moved on.

And that got me thinking about something I don't think organisations consider enough when they build AI applications:

Who is actually looking after this thing once it goes live?

The wrong question

When software stops working, the natural question is:

> *"What's broken?"*

With AI applications, there's another question that can be just as important:

> *"What changed?"*

AI applications depend on a technology landscape that is moving quickly.

- Models change.
- APIs change.
- SDKs change.
- Cloud services are updated or retired.
- Authentication methods evolve.
- Security requirements change.

Even something outside your own codebase can eventually affect your application.

That's simply part of running modern software.

The question isn't whether something will change.

It's what happens when it does.

Fixing the problem wasn’t the expensive part

Once we understood what was happening, the actual technical fix was relatively straightforward.

We reviewed the architecture, identified the affected dependency, updated the integration, rebuilt the environment and carried out testing before preparing the application for redeployment.

But there was a bigger issue.

The application hadn’t been running with an ongoing support arrangement.

So before you can fix something, you first have to understand it.

- What services are being used?
- Which versions?
- How is the production environment configured?
- What has changed since the original deployment?
- Can the problem be reproduced?
- What else might be affected by the change?

That investigation takes time.

And when something is already broken, you’re doing all of this under pressure.

This is where support becomes much cheaper than people think

I’ve seen organisations hesitate over an ongoing support contract because they think:

“Why should we pay for support when everything is working?”

It’s a fair question.

But I think it’s worth looking at the alternative.

You can pay a relatively predictable amount for someone to know your system, monitor it and keep it maintained.

Or you can wait until something breaks and then pay for someone to rediscover the system, investigate the problem, work out what changed and fix it under pressure.

The second option can become considerably more expensive.

And the cost isn’t just the consultant’s hourly rate.

It’s also:

- business disruption
- users who can’t do their jobs
- emergency investigation
- rushed changes
- additional testing
- uncertainty
- internal time spent coordinating the response

A support contract isn’t just an insurance policy against failure.

It’s a way of making the cost of ownership predictable.

The people who built it shouldn’t have to rediscover it

This is another reason I think ongoing support matters.

Imagine an application has been live for 12 months.

The developer who built it is no longer working on it every day.

Then something breaks.

You call them.

Before they can even start fixing it, they need to reconstruct the picture:

- What version is running?
- What changed?
- Which services does it depend on?
- What does production look like?
- What happened in the last deployment?
- Where’s the current code?

That’s not necessarily anyone’s fault.

It’s just what happens when operational knowledge isn’t maintained.

A good support arrangement changes that.

From project to product

This is where I think organisations need to change their mindset.

> *An AI project has a start date and an end date. A production application doesn’t.*

The project question is:

“Can we build it?”

Then:

“Can we deploy it securely?”

Then comes the question that tends to get less attention:

“Can we keep it reliable?”

And eventually:

“Can we keep it reliable as the technology around it changes?”

Those are very different responsibilities. It is the same shift we describe in moving a prototype into production — the build is only one part of the commitment.

What ongoing AI support actually means

Support shouldn’t simply mean:

> *“Call us when something breaks.”*

Good support is proactive.

It can include:

- monitoring application health and availability
- monitoring failures and performance
- keeping track of AI services, APIs and SDKs
- watching for platform changes and service retirement
- regression testing significant changes
- security and access reviews
- maintaining development, test and production environments
- keeping dependencies up to date
- incident response
- maintaining operational knowledge of the system

The aim isn’t to eliminate every possible failure.

That’s unrealistic.

The aim is to make sure that when something changes, someone notices before the business discovers it the hard way. That is essentially what managed AI hosting and support is for.

There’s a hidden cost to reactive support

One approach I’ve seen is to assume that the original developer can simply be called if something goes wrong.

Sometimes that works.

Often, the first job isn’t fixing the problem.

It’s figuring out what the system looks like now.

- Which services is it using?
- Which model versions are configured?
- What has changed since it was deployed?
- Can we reproduce the problem?
- Where is the latest code?
- What does the production environment actually look like?

It is entirely possible for that investigation to take longer than the eventual fix.

And by then, users may already be affected.

That’s why I think operational ownership needs to be considered while an AI application is being built, not after something breaks.

Production isn’t the finish line

This experience reinforced something we’ve been thinking about a lot at IntelliMinds.

Getting an AI application into production is a milestone.

It isn’t the end of the journey in any meaningful operational sense.

The application now has a life.

- It has dependencies.
- It has users.
- It has security requirements.
- It has a technology environment that will continue to evolve.

And someone needs to own what happens next.

If you already have an AI application in production, I’d ask five questions:

- 1. Who is monitoring it? Not occasionally — day to day, with someone accountable for noticing.
- 2. Who understands its dependencies? The services, models, SDKs and integrations the application relies on.
- 3. Who is watching for changes to the services it relies on? Platform updates, deprecations and retirements rarely announce themselves to the business.
- 4. Who can test and deploy a fix safely? Access, environments and a route to production that doesn’t create new risk.
- 5. Who would respond if it failed tomorrow morning? And how long would it take them to understand the system first?

If the answer to all five is:

“We’ll find someone if something happens,”

you may have a support model.

But it’s probably a reactive one.

And reactive support is usually the expensive kind.

The cheapest time to support an AI application is before it breaks

I don’t think every AI application needs a large support contract.

But every production AI application needs clear ownership.

For some organisations, that will be an internal team.

For others, it will be the original development partner.

For others, a specialist managed support arrangement may make more sense.

The important thing is that someone knows: what the system is, what it depends on, what “healthy” looks like and what to do when something changes.

Because the question isn’t:

“Will something ever change?”

It will.

The question is:

“Will you discover the change through your monitoring — or through your users?”

That’s the difference between managed AI in production and waiting for something to break.

And increasingly, I think that’s a distinction worth paying attention to.

> *The cheapest time to support an AI application is before it breaks.*

Author Vikram Katyani — Founder, IntelliMinds Digital. Helping organisations move AI from experimentation into production through practical strategy, custom development and managed AI operations.

Continue the Conversation

IntelliMinds Digital helps organisations turn AI ambition into dependable business capability. If this guide has raised questions about your own AI programme, we’d welcome the conversation.

Next steps

- **Take the free AI Readiness Assessment** — a five-minute diagnostic with a personalised report.
- **Book a 30-minute discovery call** to discuss your priorities.
- **Explore the full library** of AI Executive Guides at intellimindsdigital.com/insights/

Read the latest version online
https://intellimindsdigital.com/insights/your-ai-application-probably-wont-fail-because-of-a-bug/

Scan to open the live article, which is kept up to date as our thinking evolves.

IntelliMinds Digital
Founder: Vikram Katyani
vikram@intellimindsdigital.com
www.intellimindsdigital.com