

TECHNICAL HEALTH CHECK

A Bitcoin education platform, built by its founder on Replit

ENGAGEMENT	REVIEWED	TEAM
November 2025	Codebase, hosting, database, deploys, security basics	Founder, building alone

01 Plain-English summary

Fix it. Don't rebuild it.

The code is good. It is a well-organised monorepo with automated tests already in place, and you like what the product does. The problem is not the code; it is where the code runs.

The app, its database and every release all live on Replit. The site goes down too often, releases go straight to production with no plan for downtime, and when it breaks you are waiting on Replit support, which has not fixed it. None of that gets better as you grow; it gets worse.

My recommendation is to keep the codebase as it is and move it somewhere you control: hosting to **AWS**, the database to **Neon** (managed Postgres), with monitoring, error tracking and a proper release pipeline. Start on one server and add more only when real usage needs them, so you are not paying for capacity you do not use.

Fix Verdict. No rebuild needed	1 Critical risk: hosting outages	5 Other risks: 3 high, 2 medium	~1 hr Planned downtime for the move
--	--	---	---

This is a sample. It is based on a real engagement; the client's name and some details are changed. A real report follows the same structure.

02 What's solid

Worth saying first, because it decides everything else: most of this should not be touched.

- + **Code quality.** One monorepo, organised and readable. Someone new could find their way around it.
- + **Automated tests** are already there.
- + **Payments run on Stripe**, which is the right, boring choice.
- + **The database is backed up.**
- + **The product itself.** You are happy with what it does, so there is no reason to change features to fix the platform.

03 What's risky, ranked by business impact

RISK AND WHAT IT COSTS YOU		SEVERITY
1	The site goes down too often on Replit hosting Learners hit errors instead of lessons, and every outage spends trust you are trying to build. It is already happening, before any real growth.	CRITICAL
2	No monitoring or error tracking You find out the site is down from users, and you cannot see which errors they are hitting.	HIGH
3	Releases go straight to production No pipeline, no checks before a release goes live, no plan for downtime. Every release is a small gamble.	HIGH
4	Reliance on Replit support When hosting fails, the fix is in someone else's queue, and it has not been resolved.	HIGH
5	Login has no rate limiting Email-and-password sign-in accepts unlimited attempts, which leaves accounts open to password guessing.	MEDIUM
6	Secrets are tied to the hosting platform The Stripe key and other credentials live in Replit, so moving or rotating them depends on it.	MEDIUM
7	One server, no room to scale Not a problem at today's usage. It becomes one if usage climbs, so it is planned for rather than paid for now.	WATCH

04 What will break first

Availability. It is already breaking: the app goes down on Replit today, and a busier week, a promotion or a release at the wrong moment will make it worse. The code is not what fails first; the platform it runs on is.

That is good news, because it is the cheapest of the possible failures to fix. It does not need a rewrite.

05 Architecture and data model

A single monorepo holds the web app and its backend, on a Postgres database. The structure is sound and I would not change it.

- + **Keep** the monorepo and the data model as they are.
- + **Move** the database to Neon, a managed Postgres service with point-in-time restore. It sits outside the app's hosting, so the database is no longer tied to the same platform as the app.

06 Authentication, access control and multi-tenancy

Users sign in with email and password, built into the app. It works and I would keep it.

- + **Add rate limiting** to sign-in, so repeated failed attempts are slowed and then blocked.
- + **Multi-tenancy:** not applicable. This is one platform serving individual learners.

07 Security basics

- + **Secrets:** move the Stripe key and other credentials into AWS Secrets Manager, out of the hosting platform.
- + **Sign-in:** rate limiting, as above.
- + **At the edge:** when traffic justifies a second server, put a web application firewall (AWS WAF) in front of the load balancer, so common attacks are filtered before they reach the app.

08 Infrastructure, deployment and scalability limits

Today everything runs inside Replit. The proposal moves it onto infrastructure you own, in two steps, adding cost only when usage calls for it.

TODAY

All on Replit

Releases → Replit (no pipeline)
App → Replit hosting
Database → Replit
Secrets → Replit
Monitoring → none

STEP 1 · MONTH 1

Your own AWS, one server

Releases → GitHub Actions CI/CD
App → AWS EC2 (one server)
Database → Neon (managed Postgres)
Secrets → AWS Secrets Manager
Monitoring → CloudWatch + Sentry

STEP 2 · WHEN USAGE NEEDS IT

Two servers behind a load balancer, with a firewall

AWS WAF → Application Load Balancer → EC2 servers (two)
Trigger: thresholds agreed in month 2, from what CloudWatch shows about real usage

Scalability limits

- + **One server** is enough to start, and much more reliable than today.
- + **Two servers behind a load balancer** handle growth and let a release go out without taking the site down.
- + **Multiple regions** are not needed at this stage. Revisit if usage or the audience calls for it.

09 Code quality and test coverage

Good. The code is consistent and readable, and there are automated tests. The one change: run those tests on every push in the new pipeline, so a broken change is caught before it is released rather than after.

10 Team and vendor delivery

You built this yourself, and built it well. What you are missing is not more developers. It is a second pair of senior eyes on the parts that are not code: where it runs, how it is released and how you know it is healthy.

The vendor problem is Replit. You are not getting the help you need when hosting fails, and on a platform you do not control you cannot fix it yourself. Owning the infrastructure, with monitoring, removes that dependency.

11 Recommendation: fix, refactor or rebuild

Fix.

Not refactor, and not rebuild. The codebase does not need restructuring, and rebuilding it would spend months recreating something that already works. Every risk in this report is about the platform, the release process and visibility, and each can be fixed without touching features.

Do it in a planned window, with about one hour of downtime.

12 30/60/90-day action plan

In order of business impact. Effort is rough, in working days, and assumes one senior engineer.

DAYS 1-30

Move off Replit and see what is happening

AWS: one EC2 server, configured and secured	2-3 days
Database to Neon, with a restore tested before the switch	1-2 days
Secrets into AWS Secrets Manager	½ day
CI/CD with GitHub Actions: tests on every push, automated releases	2 days
CloudWatch alarms and Sentry error tracking	1 day
Rate limiting on sign-in	½ day
Switch-over in a planned window (about 1 hour of downtime)	½ day

DAYS 31-60

Watch real usage and set the triggers

Review CloudWatch and Sentry; fix what real traffic surfaces	Ongoing
Agree the usage thresholds that will trigger a second server	½ day

DAYS 61-90 AND AFTER

Scale when the numbers say so, not before

When thresholds are reached: a second EC2 server behind an Application Load Balancer	2-3 days
AWS WAF in front of the load balancer	1 day
Multi-region: not yet. Keep monitoring and revisit	—

What happened next

The founder went ahead with the plan.

- + The platform moved to AWS and Neon with about **one hour of downtime**.
- + CloudWatch, Sentry, Secrets Manager, a GitHub Actions pipeline and sign-in rate limiting went in as part of the move.
- + **About three months later**, as usage grew, a second server went in behind a load balancer, with AWS WAF in front.
- + Usage is still monitored, with the next steps ready for when the numbers call for them.

What every Health Check includes

- + **A written report within 5 business days**, like this one: what's solid, what's risky and what will break first, ranked by business impact, with a plain-English summary.
- + **A prioritized 30/60/90-day action plan** with rough effort estimates.
- + **A 45-minute walkthrough call** to go through the findings together.
- + **A clear recommendation**: fix, refactor or rebuild, and why.

Up to 2 repositories or 1 product. Read-only access is enough. NDA signed on request.

\$500 fixed

The full fee is credited to your first month of a fractional CTO retainer if you sign within 30 days of receiving the report.

Request yours at nikhilgarg510.com/technical-health-check

Sample report. Based on a real engagement (November 2025); the client's name and some details are changed.