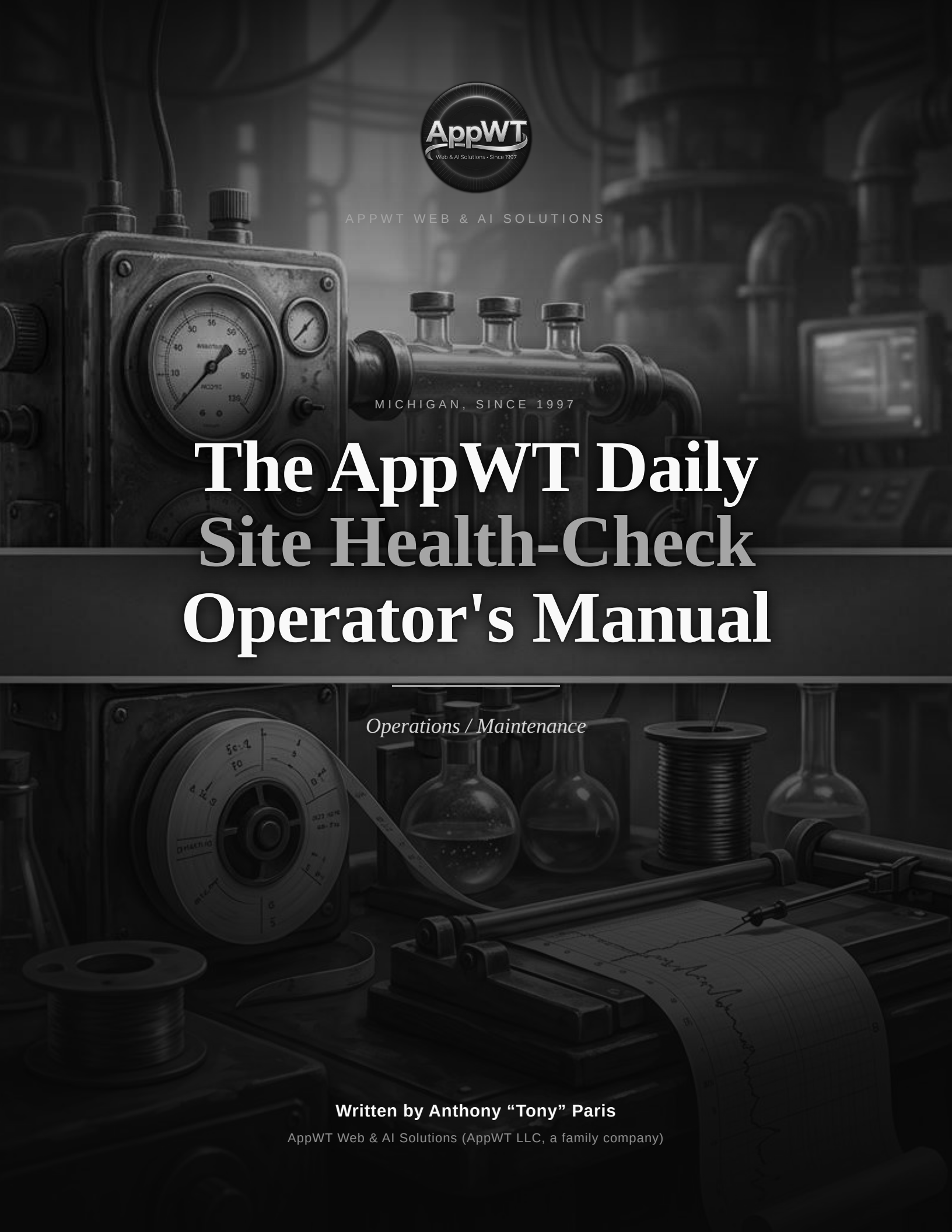


APPWT WEB & AI SOLUTIONS

MICHIGAN, SINCE 1997

The AppWT Daily Site Health-Check Operator's Manual

Operations / Maintenance

Written by Anthony “Tony” Paris

AppWT Web & AI Solutions (AppWT LLC, a family company)

The Daily Site Health-Check Operator's Manual

Operations / Maintenance

Written by Anthony “Tony” Paris

Published by AppWT Web & AI Solutions

AppWT LLC, a family company

The Daily Site Health-Check Operator's Manual

Operations / Maintenance

Written by Anthony “Tony” Paris

AppWT Web & AI Solutions (AppWT LLC, a family company)

Second Edition, August 2026

Published in Livonia, Michigan, United States of America.

A NOTE ON SPELLING

This manual is read in the United States, Canada, the United Kingdom and Australia. Where a word is spelled differently in different countries, both spellings appear the first time the word is used, like color/colour. After that, one spelling is used so the page stays easy to read.

HOW TO USE THE TECHNICAL NOTES

Smaller italic notes marked *For your tech person* hold the deeper detail: exact settings, file names and commands. You can skip every one of them and the manual still makes complete sense. Hand them to whoever looks after your website.

IMPORTANT

This manual describes security practices that AppWT uses and recommends. It is written as general guidance for business owners. It is not legal advice, and it is not a guarantee that any system cannot be attacked. Security reduces risk. No honest provider will promise to remove risk entirely.

Every incident described in this manual comes from work AppWT actually performed. Client identities are deliberately withheld, because a client's security incident is not ours to publish. Where the business involved is AppWT itself, we name ourselves.

FOREWORD

A Note From Tony

The call that started this book was about a phone number. A two-person accounting office had gone quiet since January and wanted to know whether the number on their website was wrong. It was not. Their contact form had been accepting messages and delivering none of them for six weeks, and it had shown every one of those people a thank-you page on the way out.

I have been building and fixing websites for business owners since 1997. In all that time I have almost never been called about a site that was down. Sites that are down get noticed within an hour, because customers complain and the owner picks up the phone. What I get called about is the quiet version. The form that stopped delivering. The certificate that failed to renew for eleven weeks before anybody saw a warning. The sitemap that fell from eight hundred URLs to forty-one after a rebuild that looked perfect.

That is what this manual is for. It is the daily audit we run at AppWT, written out so you can run it yourself. Seventy-eight numbered points, grouped by layer, split into thirty-one daily, thirty-two weekly and fifteen monthly. Chapter twelve is the whole list. Chapter thirteen tells you which sixty a machine can collect and which eighteen need a person, and why letting a script sign off the eighteen is how a false record gets created.

Here is what I promise. Every check in this book produces a value you can write down with a date. Not a green tick, a value. Days remaining on the certificate. URLs in the sitemap. Minutes for the canary form message to arrive. Kilobytes on the services page. Files changed since yesterday. A tick tells you about today and nothing about the week, and almost every failure in this trade shows up as a trend before it shows up as an outage.

Here is what I will not promise. I cannot promise your site will not go down, because a fair share of what carries a modern site belongs to somebody else. I have not put a single industry statistic in these pages, and where a number matters and I cannot stand behind it, you will find the mechanism explained instead. Certificate lifetimes, performance thresholds, tax and accessibility obligations and platform support dates all change, sometimes more than once a year, so the book tells you which source to read rather than pretending to be that source.

One rule runs through every chapter and I want it stated before you start. A check that cannot fail proves nothing. Before you trust a monitor, break the thing it watches on purpose and confirm it screams. There is a URL in the seventy-eight-point list that is supposed to return a 404, and it is there precisely so that a system quietly answering yes to every question gets caught. Every operator I know has one story about a green dashboard sitting on top of a dead site, and every one of those stories begins with a check nobody ever tested.

Work Part One first, in order, on one site. Five chapters and about a week. It asks you to record ten or twelve numbers a day and change nothing. Operators want to skip it, because recording numbers feels like less work than fixing things. Skip it and you have no baseline, and without a baseline you cannot tell a change from a coincidence. Do it and by the end of the week you will already have found something, which is what happens on almost every site the first time somebody actually counts.

This book is business and technical information, and it is not legal, tax or compliance advice. Chapter eleven stops at exactly that point and sends you to a licensed attorney in your own jurisdiction, because accessibility and privacy obligations differ by country, state, sector and organization size, and they have moved repeatedly in recent years. You can find and fix problems and report what you found. You cannot certify a client as compliant, and a sentence that reads as though you have is a real risk to your business.

Some of what is here will be uncomfortable reading if you already run a maintenance service. Your backups may not include the database. Your monitor may be watching the one page that never breaks. Your alert channel may already be muted by somebody on your own team. Those are ordinary findings and the book treats them as ordinary. What matters is that you found them on a quiet Tuesday with a checklist rather than at nine at night with a client on the phone.

Last thing. Maintenance done well produces nothing anybody can see, which is why good operators lose accounts to clients who say the site was fine all year. It was fine because of the work. Chapter fourteen is about making that visible, pricing it from your own measured minutes, and keeping the record that answers any question a client asks with a number and a date. That record is the difference between a service and a favor.

Anthony “Tony” Paris

Founder, AppWT Web & AI Solutions · Livonia, Michigan

BEFORE YOU BEGIN

How to Use This Manual

This manual is built to be run, not read once. Keep one sheet per site and one row per day, and put every value you measure into it as you go. Fourteen chapters produce one checklist, one log, one incident record and one report, and chapter fourteen turns those into a service you can price.

The parts

Part One covers the daily loop, which is everything that can fail overnight while the site still looks perfect. Chapter 1 defines the six layers and the four classes of silent failure. Chapter 2 takes uptime apart into six separate steps and builds a monitored URL list that includes a page which must return 404. Chapter 3 covers certificate expiry and the eight ways automatic renewal quietly stops. Chapter 4 covers DNS delegation, the domain's own expiry and the mail records that decide whether your confirmations arrive. Chapter 5 turns sitemaps, robots rules and canonical tags into three numbers you compare every morning.

Part Two covers the checks a script can prepare and only a person can settle. Structured data compared against the real price, the real stock and the real opening hours. Speed measured on a phone with an empty cache rather than on your own desk. The forms, carts, phone links and confirmations that fail while showing a thank-you page. Backups nobody has ever restored, and the quarterly drill that turns an archive into a recovery time you can quote. The security values worth reading daily, and the line where a paid security audit begins. Then the accessibility and legal items that decay quietly with ordinary content editing.

Part Three makes it an operation. Chapter 12 is the full seventy-eight-point list, grouped and numbered, split into thirty-one daily, thirty-two weekly and fifteen monthly. Chapter 13 automates sixty of them, names the eighteen that need a human by number, and rebuilds your alerting so the channel does not get muted. Chapter 14 covers the monthly report, retainer pricing from your own timed passes, response commitments you can actually keep, and the incident record that ends arguments before they start.

The boxes, and what each one means

DO THIS TODAY

A gold box is one action you can finish now. If you only ever act on the gold boxes, you will still be far safer than when you started.

FROM OUR FILES

A dark box is a real incident from AppWT work. Nothing in these boxes is invented. Where a detail was never verified, we say so rather than fill the gap.

FOR YOUR TECH PERSON

A smaller italic note like this one carries the professional detail: exact file names, headers, settings and commands. Skip every one and the manual still reads completely. Forward them to your web person.

The tables, the sheets and the two code blocks

The tables are meant to be worked from rather than admired. Failure classes and the check that catches each. Six steps behind one page load. Eight renewal failure modes. Eight DNS record types. Performance causes with the fix and who does it. Retainer tiers with the exclusions written as carefully as the inclusions. The two code blocks are column headings for sheets you build in whatever spreadsheet you already use: the daily log in chapter 13 and the incident record in chapter 14. Type them in once and the format stops being a decision you make every time.

What this book does not cover

This is a maintenance manual, not a build manual and not a marketing manual. You will find no design guidance, no content strategy, no keyword work and no advice on winning clients. Deep performance engineering, full security auditing, accessibility conformance work and payment gateway setup each belong to separate AppWT titles, and this book points at them rather than repeating them badly. Nothing here is legal, tax or compliance advice, and chapter eleven says so at the exact point where it matters.

If you only have one afternoon

Do five things on your largest site, in this order. Read the certificate expiry off the live connection and write the number of days down. Count the URLs in the sitemap and compare that number against the page count in the content system. Submit the contact form with a unique marker and go and find it in the destination inbox, junk folder included. Confirm at least one backup exists somewhere that is not the web server. Then check whether any archive, database dump or version control folder is sitting in the web root where anybody can download it. All five are quick, all five are free, and between them they catch the failures that cost the most.

Where to keep the outputs

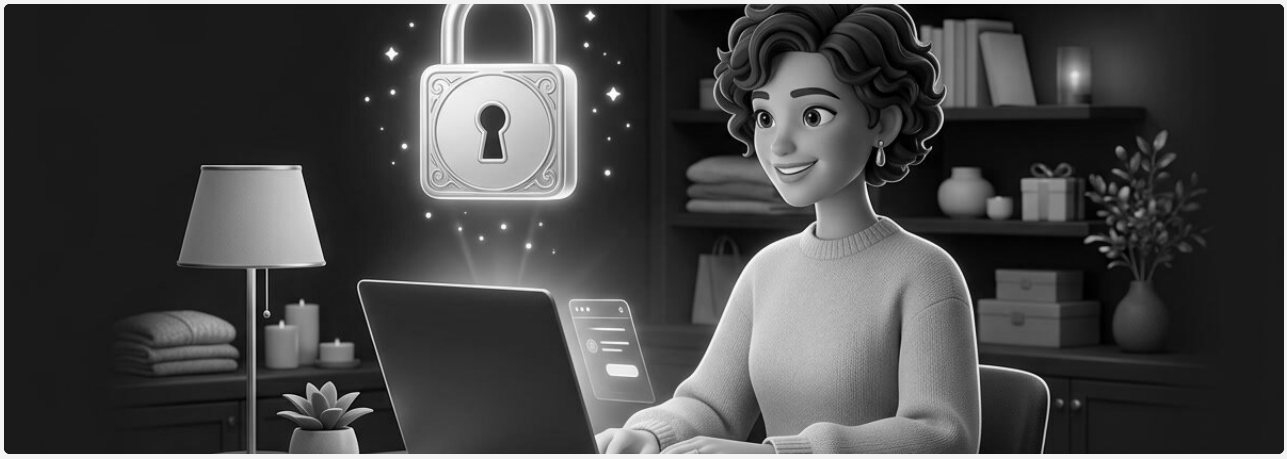
Keep one folder per site and treat it as part of the service. It holds the site file with every baseline value and a date, the monitored URL list with a reason per entry, the daily log, the access list with a name against every account, the restore drill records, the incident log and the monthly reports you sent. Keep the old versions rather than overwriting them, because a value compared against itself can never show a change, and last month's sheet is the only thing that proves this month's decision worked.

Contents

PART ONE

The Daily Loop

Five chapters covering the checks that change without warning: whether the site answers, whether the connection is trusted, whether the domain and mail records still say what you think, and whether the files that tell search engines what exists are still telling the truth. These run every morning because every one of them can fail overnight while the site looks perfect.



CHAPTER 1 PART ONE

The Failure Nobody Reported

Chapter 1. What site health means, and why a broken site so rarely tells you it is broken

Most site failures do not announce themselves. The page still loads, the logo is still there, and something behind it stopped working weeks ago. This chapter names the four ways a site fails quietly, sets the operator's standard for a check that can actually fail, and starts the log that every later chapter writes into.

A two-person accounting office called on a Tuesday in March. Their question was about the phone. New inquiries had gone quiet since January, and they wanted to know whether the phone number on the site was wrong. It was not. The number was correct, the page loaded, and the site looked exactly the way it had looked for three years. The contact form on that page had been accepting messages the whole time and delivering none of them. A mail setting had changed at the host in early January. The form kept showing the thank-you page. Nobody on either side had a reason to look.

That is the shape of almost every serious site problem. Not a crash. Not a blank screen. A working-looking page with something dead behind it. A crash gets reported within an hour, because customers complain and the owner calls. A silent failure runs until somebody counts. The whole purpose of a daily health check is to be the thing that counts, on a schedule, before the customer does. This book is the operator's manual for that work. It covers what to check, how to check it so the check can actually fail, what to do with the answer, and how to run the whole thing as a paid service rather than as an unpaid extra.

Three Questions, in Order

Site health sounds vague until you break it into three questions and ask them in order. Is the site reachable. Is it trusted. Is it doing its job. The order matters, because a lower question cannot be answered by fixing a higher one. Perfect structured data does nothing for a site that returns a server error. A fast page does nothing for a site whose certificate expired last night. Work from the bottom up, every time, and stop chasing the interesting problem before the boring one is settled.

1. Reachable. The domain resolves, the server answers, and the answer is the status code you expect on the pages you care about.
2. Trusted. The certificate is valid and not close to expiry, the connection is encrypted end to end, and the browser shows no warning.
3. Findable. Robots rules, sitemaps, canonical tags and structured data agree with each other and with the pages that actually exist.
4. Usable. The page renders on a phone, it loads in a reasonable time, and a person can read it and act on it.
5. Working. Forms deliver, carts take payment, phone links dial, and the confirmation the customer expects actually arrives.
6. Recoverable. A backup exists, it is recent, it is stored somewhere the server cannot reach, and somebody has restored one.

Those six layers are the spine of this book. Chapters two through five cover reachable, trusted and findable, which are the checks you run every day because they change without warning. Chapters six through eleven cover the rest, which need a human eye and a slower rhythm. The last three chapters turn the whole thing into a routine you can run in twenty minutes, automate most of, and bill for.

The Four Ways a Site Fails Quietly

Silent failures are not random. They fall into four groups, and once you can name the group you already know roughly where to look. Expiry failures happen because something had a date on it and nobody wrote the date down. Delivery failures happen because a message left the site and never arrived, and the site has no way to know. Drift failures happen because somebody changed a setting for a good reason and a second thing depended on the old setting. Dependency failures happen because code you did not write changed underneath you.

FAILURE CLASS	WHAT ACTUALLY HAPPENS	WHY NOBODY NOTICES	THE CHECK THAT CATCHES IT
Expiry	A certificate, a domain registration, a card on file or an API key reaches its end date.	It works perfectly until the exact moment it does not, so there is no warning period to observe.	A dated list of every expiring item, checked daily against a countdown rather than a reminder.

FAILURE CLASS	WHAT ACTUALLY HAPPENS	WHY NOBODY NOTICES	THE CHECK THAT CATCHES IT
Delivery	A form submission, an order confirmation or an alert leaves the site and never reaches an inbox.	The visitor sees a thank-you page, so the site reports success while delivering nothing.	A real submission sent on a schedule to an address you control, confirmed as received.
Drift	A plugin, a redirect, a robots rule or a DNS record is changed and quietly breaks a second thing.	The change was deliberate and looked correct in isolation, so nobody looked at what depended on it.	A recorded baseline value per item, compared against today's value, with any difference shown.
Dependency	A third-party script, a payment library or a platform feature changes behavior/behaviour or shuts down.	Nothing on your server changed, so every internal check still passes.	A rendered-page test in a real browser plus a monthly read of vendor change notices.

A CHECK THAT CANNOT FAIL PROVES NOTHING

Before you trust any monitor, break the thing it watches on purpose and confirm the monitor screams. Point it at a hostname that does not exist. Rename the sitemap for sixty seconds. Submit the form with a bad address. If the check stays green while the subject is broken, the check is decoration. Every operator has a story about a green dashboard sitting over a dead site, and every one of those stories starts with a monitor nobody ever tested.

Record the Value, Not the Verdict

Most people who run checks record a verdict. Uptime, pass. Certificate, pass. A verdict tells you about today and nothing about the week. Record the value instead. Not certificate pass, but certificate expires on the eleventh of next month, forty-one days out. Not sitemap pass, but sitemap contains 312 URLs, and yesterday it contained 314. A verdict cannot show you a trend, and almost every quiet failure shows up as a trend first. Two URLs disappearing from a sitemap is not an outage. It is the first visible edge of something that becomes an outage in six weeks.

So the first artifact this book asks you to build is a log with one row per day and one column per value. It can live in a spreadsheet. It does not need software. What it needs is the number, the date, and the same measurement method every time, because a comparison only means something when both sides were measured the same way. When you change how you measure something, write that down too, and stop comparing across the change.

SIX WEEKS OF NOTHING, PRICED OUT

The two-person accounting office ran the numbers after the form was fixed. In the same period the previous year they had logged 22 inquiries through the site. This year they logged 3, all of which had come by phone. They knew from their own records that roughly one inquiry in four became a client, and that a new client was worth about $1,400 to them in the first year. Nineteen missing inquiries at those figures came to roughly five clients and around $7,000 of first-year work. Their maintenance arrangement at the time was an hourly call-us-when-it-breaks arrangement, which had cost them nothing that quarter. A daily form-delivery test would have caught the failure on day one. The point is not the exact dollar figure, which is theirs and not a benchmark. The point is that the cost of the silent failure was many times the cost of the check that would have found it, and that is nearly always true.

What Changed, and What Did Not

The checks in this book are older than the web. Is it up. Is it correct. Can we get it back. What changed is the number of moving parts under a single ordinary site. A small business site in 2026 usually sits on top of a host, a separate DNS provider, a certificate that renews itself automatically, a mail sending service, a payment processor, an analytics tag, a chat widget, a booking tool and a content system with two dozen extensions. Each one is somebody else's system with its own change schedule and its own failure modes. None of them tells you when it changes.

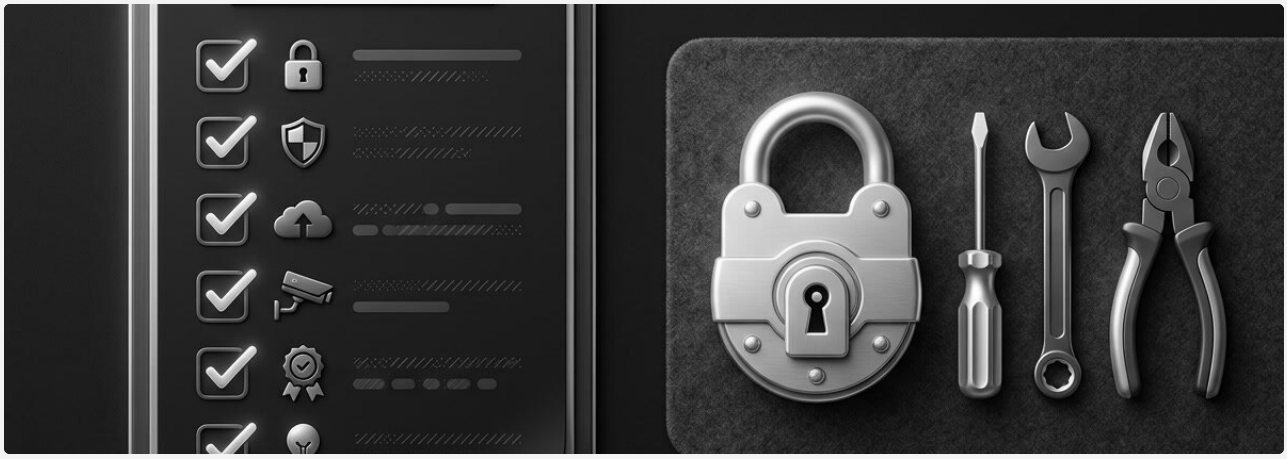
Two things also got harder to see. Certificates now renew on their own, which removed a yearly manual task and replaced it with an automated process that fails quietly when it fails at all. And a growing share of what reaches a customer is assembled by something other than a browser window, including search result panels and AI assistants that read the page and restate it elsewhere. A site can be perfectly healthy for a human visitor and unreadable to those readers, which is why structured data and plain text schedules get their own chapter later. The mechanism is what matters here. More systems, less visibility, same three questions.

> A green dashboard is a claim, not a measurement.

One more habit before the checks begin. Write down what you are responsible for and what you are not, per site, in plain words. An operator who is blamed for a payment processor outage they cannot see, or for content the client edits directly, will not last a year. A short written scope, agreed once, turns every future argument into a reference rather than a fight. Chapter fourteen gives the wording. For now, list the sites you look after and write one line each about who owns the hosting, who owns the domain, who owns the content and who gets called at night.

WHAT TO REMEMBER

- Most site failures are silent, so the site keeps looking correct while something behind it has stopped working.
- Ask the layers in order, because reachable comes before trusted, trusted comes before findable, and nothing above rescues a failure below.
- Quiet failures fall into four classes: expiry, delivery, drift and dependency, and naming the class tells you where to look.
- Break a monitor on purpose before you trust it, because a check that cannot fail is decoration.
- Record the measured value with a date rather than a pass or fail, because trends appear in values and never in verdicts.
- Write down, per site, who owns the hosting, the domain, the content and the emergency phone call.



CHAPTER 2 PART ONE

Uptime Is Not One Question

Chapter 2. Building a reachability check that can actually fail

A monitor that says a site is up has usually answered a much smaller question than the one you asked. This chapter takes reachability apart into the six steps a real visitor goes through, shows what each step proves and what it hides, and builds a check that watches the pages the business is paid from rather than only the front door.

A dental clinic taking around forty bookings a week called on a Thursday morning. Their monitor had reported 100 percent uptime for the month. Their booking page had been returning a server error since the previous evening. Both statements were true. The monitor was checking the homepage of the main domain, which is a static page served by the web server. The booking system lived on a separate subdomain, ran a database behind it, and had fallen over when that database ran out of disk space. Nothing the monitor watched had changed. Everything the business was paid from had stopped.

Uptime feels like a single yes or no. It is not. Reaching a web page takes six separate steps, each run by different software, each able to fail on its own. A monitor answers whichever step it happens to reach and reports that as the whole picture. Your job as an operator is to know which step your check is actually testing, and to make sure the pages that carry the money are among the pages being tested.

The Six Steps Behind One Page Load

STEP	WHAT HAS TO WORK	WHAT A PASS PROVES	WHAT A PASS STILL HIDES
Name lookup	The domain resolves to an address through public DNS.	The domain is registered and the records are being served.	Nothing about whether the server at that address is running.
Connection	The server accepts a connection on port 443.	A machine is alive and listening.	Whether the software behind the port is the right software or is healthy.
Secure handshake	The certificate is valid, in date, and matches the hostname.	A browser will not show a warning screen.	Whether the certificate expires in three days.
Status code	The server returns the code you expect for that URL.	The application ran far enough to answer.	Whether the answer contains the content the page is supposed to have.
Body content	The response contains a string you chose in advance.	The page rendered its real content, not an error template.	Whether scripts that build the page in the browser ran at all.
Rendered page	A real browser loads the page and the visible content appears.	A human visitor sees a working page.	Whether a form on it delivers, which is a different check entirely.

Read that table as a ladder. Most cheap monitors stop at step four. They ask for a URL, they see a status code in the 200 range, and they mark the site green. That is useful and it is not enough. A content management system that has lost its database often returns a friendly error page with a 200 status, because the error page itself loaded fine. A cached page can serve for hours after the thing behind it died. A redirect chain can quietly end somewhere you never intended, and the final page still answers 200.