COLDCODE SI LABS LIBRARY

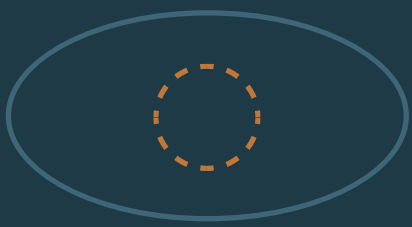
Nobody Was Watching

How founders run autonomous growth systems without harming strangers, their brand or themselves

Theories, frameworks and one candid case for founders, growth leads and strategists

Courage Garikai

Funded by Coldcode Si Labs



FIRST EDITION · 2026

NOBODY WAS WATCHING

Copyright © 2026 Courage Garikai.

Published by Coldcode Si Labs Library, Harare, Zimbabwe.

First edition, 2026.

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted
form or by any means without the prior written permission of the publisher, except in the case
quotations in a review or certain other noncommercial uses permitted by copyright law. The fr
worksheets and checklists in this publication are meant to be used, copied and adapted by the

This is a work of analysis and practical guidance. It is not financial, legal, investment or profes
advice, and no outcome is promised: results depend on the reader's own market, effort and juc
Platform behaviour and figures were checked at the time of writing and change.

COLDCODE SI LABS LIBRARY · Harare, Zimbabwe

Contents

About this edition 4

PART I · The quiet machine

1 · The suite that passed 10
2 · Why autonomy got cheap 16
3 · Unwitnessed autonomy 21
4 · Strangers pay first 27

PART II · Anatomy of an unwatched pipeline

5 · The wrong recipients 34
6 · Borrowed identities 39
7 · Duplicate clocks 44
8 · Silent failure 49
9 · Misreading the world 54
10 · Authority and the money path 59

PART III · Witnessing

11 · The witness map 65
12 · The account inventory 71
13 · Reading the ledgers 76
14 · Dry runs and the defects that fixes create 81
15 · Health screens that tell the truth 86

PART IV · Building to be watched

16 · One clock, one voice, one identity, one surface 92
17 · Outreach that strangers can stop 97
18 · Armed, not active 102
19 · Ceilings as design 107
20 · Stops and the human in the loop 112

PART V · Leading autonomous growth

21 · Metrics that hide harm 118
22 · Law, ethics and reputation 123
23 · Governing the builders 128

24 · A ninety-day programme 133

TOOLKIT · Frameworks, guide, glossary and references

25 · Framework compendium 140
26 · Facilitator's guide 146
27 · Glossary 151
28 · References 155

About this edition

First edition · 2026

This book is written for founders, solo operators, growth leads, strategists and directors who run, or are about to run, growth systems that act without a person in the loop. Examples are a scheduled job that finds prospects, a runner that emails them, a module that posts to social accounts and a webhook that switches a customer's account on or off when a payment arrives. It is a discussion book. Each chapter sets out a question, the theory that helps answer it, a working case, a framework you can use, the limits of the evidence, the points on which reasonable people disagree, questions for discussion and a short exercise.

The case is Tapstall, a shop-link service for informal and social-commerce merchants in Zimbabwe and South Africa. On 28 October 2026 its live version, 1.9.1, was audited and then rebuilt the same day as version 2.0.0. On the morning of the audit its own test suite passed 420 tests out of 420. In the same code, the outreach runner had already emailed 17 addresses, and 14 of them were not prospects at all. Tapstall belongs to the same owner who commissions the work of Coldcode Si Labs. The audit and the rebuild were carried out by the owner working with AI coding agents, and the same process produced the analysis in this book. It is therefore a bounded case, not an independent evaluation. At the time of writing Tapstall had **no paying customers**. Nothing in this book is evidence that its pipeline succeeds commercially.

This book does not say how the first version of the pipeline was produced. Some of its defects resemble failure modes documented in AI-assisted work. They are equally consistent with one busy person wiring several products together on shared accounts. Where that question arises it is treated as open.

Throughout, three kinds of statement are kept apart:

- **Research findings** are cited to their sources.
- **Facts about the case** come from the live deployment, its stored data, its sent mail, its account inventory and its commit history, as of October 2026.

- **Proposed frameworks, constructed examples and recommendations** are labelled as such. They are proposals to test, not proven laws.

The people and businesses who received messages from the system are not named. Neither are the three real small businesses it contacted. No professional biography, endorsement, ISBN or independent audit is claimed. The author and funding credits are those specified by the commissioning owner. AI tools assisted in drafting this
© 2026 Courage Garikai. All rights reserved. Published by Coldcode Si La
Library, Harare. No part may be reproduced without written permission,
the frameworks and exercises, which are meant to be copied.

The promise of this book

Until recently, running a growth operation meant employing people to do it. Someone researched prospects, wrote to them, posted on social media and kept the books. Each of those people was also a witness. A researcher who found that a "prospect" was a weather service did not write to it. A salesperson who saw a reply saying "please stop" stopped. A bookkeeper who saw a payment that matched no customer asked about it. Most of the time these judgements were never written down, because nobody needed to write them down. They happened in someone's head on the way to doing the job.

Automation removes the people, and with them the witnesses. A small venture can now run a lead hunter, an email runner, a social poster and a payment matcher on free cloud services, at no cost, around the clock. It can build all of that in days. What it does not get for free is anyone placed to notice when the system is doing something it should not. The machine does exactly what it was told. The effects land on strangers, on other ventures and on the founder's own reputation. No report of those effects comes back.

This book calls that condition **unwitnessed autonomy**. It argues that founders who automate growth need a new habit to match. Before switching an action on, they should ask who will see its effects and how word of those effects will reach them. The title comes from the case. Every problem in Tapstall's growth pipeline was visible to someone: the magazine reviewer who received a sales email, the followers of another brand who saw Tapstall's posts, the merchant whose store was suspended without the warning email ever being sent. Nobody was watching on behalf of the business.

Who this book is for

- **Founders and solo operators** who automate prospecting, outreach, posting or billing, and must decide what to switch on and what to watch.
- **Growth leads and strategists** who design pipelines, choose channels and set targets, and need to know which metrics hide harm.
- **Operators of several products** who share accounts, mailboxes and credentials across ventures, and need to see how one product can act in another's name.
- **Directors, investors and advisers** who are shown growth dashboards and need questions that get past them.
- **Educators and consultants** who want a contemporary case and frameworks to teach with.

No programming knowledge is assumed. Technical details appear only where they change a decision, and are explained when they do.

How to use this book

Each chapter follows the same pattern:

1. **An opening question** that the chapter tries to answer.
2. **Theory:** the research and ideas that bear on the question, cited so that you can read further.
3. **The Tapstall case:** what the case shows, and what it does not.
4. **A framework or tool** you can apply.
5. **What the case does not show:** the limits of the evidence for that chapter's claims.
6. **Points of debate:** where the argument is contestable.
7. **Discussion questions** for a reading group, a class, a leadership team or your own reflection.
8. **Apply it:** a short exercise for your own systems.

The chapters can be read in order or by part. Groups meeting weekly can take one part at a time over five sessions. Chapter 25 collects every framework in one place. Chapter 26 is a facilitator's guide, Chapter 27 is a glossary and Chapter 28 lists the references.

The structure

- **Part I · The quiet machine** (Chapters 1–4) explains how a growth system can pass every test while harming people, defines unwitnessed autonomy, and shows why strangers bear the first costs.
- **Part II · Anatomy of an unwatched pipeline** (Chapters 5–10) dissects the forms the problem takes: wrong recipients, borrowed identities, duplicate clocks, silent failures, misread replies and leaks of authority on the money path.
- **Part III · Witnessing** (Chapters 11–15) sets out the witness map, the account inventory, reading the ledgers, dry runs that catch the defects that fixes create, and health screens that tell the truth.
- **Part IV · Building to be watched** (Chapters 16–20) describes how to build so that effects are seen: one clock, one voice, one identity and one surface; outreach that strangers can stop; distribution that waits for its own accounts; free-tier ceilings as design; and kill switches.
- **Part V · Leading autonomous growth** (Chapters 21–24) addresses metrics, law and reputation, governing the agents and tools that build these systems, and a ninety-day programme.

A note on the case

Tapstall gives a merchant a shop page: a menu, a WhatsApp ordering button, and a way for buyers to pay the merchant directly with a payment reference that the service matches. It costs US$60 a year after a 60-day free period, with card checkout through Whop, a third-party payment platform. It runs on a free serverless tier, with scheduled jobs on GitHub Actions. Its growth pipeline has five parts:

- a **lead hunter** that looks for small businesses;

- a **gate** on the server that decides which leads may be contacted and assigns them;
- an **outreach runner** that sends the emails and reads the replies;
- **sellers**, independent people paid a 30% commission, who work the leads that reply;
- a **press module** that posts to social accounts three times a day.

That is a typical shape for a small venture with ambition and no budget. Every problem described in this book arose in ordinary components doing ordinary jobs. That is the point. The case is not exotic, and the lessons are not about one product.

The audit coded 24 defects: 11 found in the audit and 13 found during the rebuild, four of which the rebuild itself introduced. Sixteen of the 24 (67%) were what this book calls *witness gaps*. None of the 24 was found by the existing test suite. The working paper that accompanies this book, *Unwitnessed Autonomy* (Coldcode Labs Think Tank Working Paper 2026-03), sets out the coding and the method in academic form.

PART I

The quiet machine

Nobody Was Watching

1 · The suite that passed

> **Opening question.** If every test passes, what do you actually know about what your system did yesterday?

On the morning of 28 October 2026, Tapstall's test suite ran 420 tests and passed every one. The tests checked the code that was live in production at that moment. By the standards most small teams use, the product was in good health. Its owner console showed a pipeline with leads in it, a press module posting three times a day and a growth runner sending emails on schedule.

By the evening of the same day, an audit had found that, in that same code:

- the outreach runner had emailed 17 distinct addresses, and 14 of them were not prospects at all;
- those emails had gone out from the owner's personal mailbox, which was also the published support address of a separate venture;
- the press module had been posting into that other venture's social account for eight days;
- an old copy of the service was still running on the same cloud account, against live data, with a secret that should no longer have worked;
- every email the product promised to send to its own merchants had silently gone nowhere since launch.

None of those problems was a test failure, and none of them could have been. This chapter is about why.

What a test can see

A test is a small experiment. It gives a piece of code an input and checks the output against an expectation. When it passes, it tells you that this code, given this input, in this test environment, produced the expected result. That is valuable knowledge. It is also narrow in four ways that matter for autonomous systems.

A test sees the code, not the deployment. The test suite ran against the code in the repository. It could not see that an older copy of the service, built from older code, was still deployed under another name on the same account. Nor could it see that the credentials bound to the live service belonged to someone else. Those are facts about the deployment: about accounts, settings and secret stores. Code does not contain them.

A test supplies its own inputs. The outreach tests fed the gate realistic-looking leads. They did not feed it a magazine's contact page, a weather service's footer or a literal placeholder address, because nobody imagined those would arrive. Yet those were exactly what the lead hunter was producing. The test inputs were plausible. Production's inputs were not.

A test supplies what production lacks. The merchant-mail tests provided a mail sender, so the code that called it passed. In production, no mail sender was ever connected. The function ran, found nothing to send through and returned quietly. The test proved that the code would send mail if it could. Production never could.

A test cannot see strangers. The most serious effects of the pipeline landed outside the system altogether: in inboxes, on social timelines, in merchants' accounts. No test observes a stranger's reaction, because no test has a stranger in it.

The safety scientist Nancy Leveson (2011) argues that accidents in complex systems usually come not from failed components but from unsafe interactions among components that each work as designed, and from controllers that lack accurate feedback about the process they control. The Tapstall pipeline fits that description closely. Its parts worked. What failed was the feedback.

Normalised success

Diane Vaughan's (1996) study of the Challenger launch decision introduced the idea of the *normalisation of deviance*. An organisation repeatedly sees a warning sign without a disaster following, and gradually comes to treat the sign as normal. A passing test suite can produce something similar in reverse, which we might call *normalised success*. Each green run makes the next one feel like proof, until "the tests pass" stands in for "the system works". The suite was not lying. It was answering a narrower question than the one people heard.