

VIBE CODING. REALITY CHECK.

The useful part begins after the impressive first draft.

This is a portable companion to the interactive workshop at vibecoding.sahilverse.online. It walks through three genuine micro-apps with deliberately missing rules: a calculator, a searchable portfolio, and a neighbourhood order tracker. Each one begins with a plausible first pass, meets an awkward edge case, and gets a specific repair.

THE CONTRACT

Specify the job. Inspect the assumptions. Test what a demo conveniently avoids. Repair and rerun. Document what will still need human judgment.

Your route through this guide

01-02	Frame the request and the method
03-05	Work through three micro-app case files
06-07	Inspect code and design meaningful tests
08-09	Consider accessibility, security and stewardship
10-11	Do the field exercises and follow evidence

Author: Sahil Tayal / Sahilverse. Published as a free independent educational experiment, 26 October 2026. All scenarios are authored teaching illustrations. No AI agent, live code scanner, certification service, or individual productivity measurement is operating behind these pages.

SPEED IS NOT A SPECIFICATION.

The difference between making an object appear and making the right object.

Vibe coding is an informal label for natural-language-led, AI-assisted programming. Its boundaries are debated. Some people use an assistant as a pair programmer; some accept large generated changes and check them by running; others use code generation for quick prototypes. The term alone cannot tell us the quality, safety or maintainability of the artifact.

A five-move loop

1 / SPECIFY	Who is this for? What counts as correct? What should not happen?
2 / DRAFT	Create the smallest useful version with explicit file structure.
3 / INSPECT	Read assumptions, coercions, dependencies, permissions and data flows.
4 / TEST	Check intended use, missing inputs, accessibility, recovery and misuse.
5 / HAND OVER	Export source, record limitations, assign repairs and review again.

AN HONEST PREVIEW

A run in a sandbox establishes that something works in that sandbox. It does not establish compatibility across devices, resistance to abuse, legal compliance, accessibility conformance or trustworthiness of a connected service.

What the evidence supports

The 2026 Stack Overflow Developer Survey asks respondents what they do with AI coding output; many report validation before use. A 2026 research review describes conflicting productivity evidence across tasks and methods. Neither source supports a single universal claim that AI coding is inherently better or worse.

NUMBER DESK: THE GHOST ZERO.

A calculator that produces an answer before the visitor finishes entering one.

Use case: adding two numbers on a mobile phone. A quick version accepts two inputs and uses JavaScript conversion. When a user leaves a field empty, **Number('')** evaluates to 0. The arithmetic completes and the missing number disappears inside a plausible total.

Illustrative failure record

Input	Expected	Unrepaired	Repaired
2 and 3	5	5	5
blank and 3	Ask for both	3	Error message
-2 and 3	1	1	1

Inspect the rule

DRAFT

const total = Number(rawA) + Number(rawB); // Missing data silently acquires the meaning zero.

REPAIR

Check rawA.trim() and rawB.trim() for blank strings before conversion. Validate finite values, announce an error in an aria-live status and do not produce a total until both fields exist.

Try the actual exported code

In the web workshop, select **Number Desk**. Run the test view, inspect the known failing edge case, apply the repair, then export the ZIP. Extract it locally and open index.html. The workshop uses authored illustrative checks; the exported code is independently runnable and must be checked again.

SHOW & TELL: THE EMPTY SILENCE.

A functioning search can be a confusing experience when nothing matches.

Use case: a small portfolio with three projects and a text filter. The first draft removes nonmatching cards but offers no message when the result count becomes zero. An unfamiliar visitor may reasonably think the page failed or is still loading.

One screen, two interpretations

Before repair	After repair
Filter: banana. Blank region. No count. No instruction.	Filter: banana. "No projects matched. Try another search or clear the field."

What changed in the code

The source creates project nodes using `textContent` rather than treating project text as markup. The repaired version additionally toggles a visible empty-state paragraph with `role="status"`. This is a design and accessibility decision, not a mysterious AI improvement.

THREE TESTS

(1) Search design: a design card remains. (2) Search a term absent from every card: a clear recovery instruction appears. (3) Clear the field: all cards return, focus remains usable.

Where this transfers

Catalogs, knowledge bases, product lists and internal tools often contain the same failure. A mathematically correct filter can still create an unusable screen. Empty states need language, context and sometimes an escape hatch. A screen reader user also needs a perceivable change.

KIRANA COUNTER: AN ORDER OF ZERO.

A shop form teaches the difference between a number and a valid business event.

Use case: item, quantity, unit price and a running total. A naïve implementation accepts numeric values but forgets the domain rule: an order line should represent a positive whole number of units. Zero is a valid number and an invalid sale.

The stated business rules

Field	Rule	Why it matters
Item	Nonblank name	An unlabelled line is difficult to resolve.
Quantity	Positive integer	Zero or fractional units may violate this example's stock model.
Unit price	Finite, nonnegative number	Missing and invalid prices should not become totals.
Ledger	Memory only in this demo	The exercise handles no real customers or payments.

Repair scope

Check each field before appending a line. On failure, keep the order unchanged and explain what needs correction. On success, append a readable line and recalculate the total. The exported mini-app does **not** implement stock deduction, returns, taxes, receipts, customer identity or payment authorization.

HUMAN QUESTION

Would you deploy a real shop system that forgets everything on refresh? Probably not. The limitation is acceptable in a learning demo because it is disclosed. In a real system, persistence, backups, offline conflicts and correction workflows need separate design.

MAKE A PROMPT WORTH TESTING.

What to write down before asking any assistant for code.

A vague request transfers ambiguity to the generator. A good brief names a person, an actual task, inputs and constraints, the negative cases to handle, the deliverables, and the limits of evidence.

A portable specification template

Audience	Who should complete this task, on what device?
Core job	What observable result matters?
Inputs	What is valid? What is missing? What is unsafe?
States	What happens when loading, empty, wrong or complete?
Constraints	Keyboard, screen reader, privacy, offline, performance?
Deliverables	Source files, README, tests, assets, fallback?
Review	Who checks? What is out of scope?

WORKED BRIEF: KIRANA

Build an offline micro-app for recording fictional line items. Name and quantity are required. Quantity must be a positive whole number, price a finite nonnegative number. Show readable errors, support keyboard entry, avoid accounts and third-party calls, provide an `index.html/styles.css/app.js` handoff and a manual test plan.

Make the constraints falsifiable

“Make it accessible” is too vague to test. “Every action works by keyboard, errors are text, focus is visible, dynamic output has appropriate live announcements, and reduced-motion users can complete the task” is more testable. Do not claim certification based on a prompt.

TEST THE PROMISE, NOT THE SCREENSHOT.

A checklist you can use even without automation expertise.

The five ways a demo usually hides work

Happy path	Try intended values and verify the exact expected result.
Edge path	Try blank, repeated, extreme and unexpected values.
Interaction	Try touch, keyboard, focus and screen reader announcements.
Environment	Test 360px, large text, different browsers, no JavaScript.
Operational	Review where data travels, what persists and how errors recover.

A record worth keeping

Write test case, expected output, observed output, date, environment, pass/fail, and remaining unknowns. A test that was never executed is **NOT TESTED**, not silently green. Record screenshots when appropriate; they complement but do not substitute for behavioural checks.

PRACTICE: REWRITE A VAGUE TEST

Weak: “The order form works.” Strong: “With item Tea, quantity 2, price ₹15, submitting appends one line, total reads ₹30.00, and the message is available through the status region.” Now invent a test for quantity zero.

The workshop's three-check teaching model is transparent. Its checks are not a comprehensive test runner. Every exported ZIP also includes TEST_PLAN.md for independent verification on a real device.

MAKE THE DEFAULT BIG ENOUGH.

People and devices are not identical; good defaults account for that.

Accessibility is part of product correctness. Use semantic controls, visible labels, clear focus, reasonable contrast, meaningful text errors, and motion that respects user preferences. Try zoom and narrow widths. Keyboard access is necessary but not sufficient for full conformance.

Small but consequential details

Problem	Better practice
A placeholder is the only label	Provide a persistent visible label associated with the control.
An error changes only color	Use text explaining what happened and how to recover.
A status appears visually only	Choose an appropriate live region or focus strategy.
Animation blocks the task	Offer skip and reduced-motion behaviour.
The phone layout clips buttons	Use fluid widths and test actual narrow viewports.

A CAUTION ABOUT BADGES

WCAG 2.2 provides testable success criteria, not a magic label acquired by adding ARIA attributes. Review against the relevant criterion and the actual experience, ideally with assistive technologies and people using them.

Useful reading

W3C WCAG 2.2: <https://www.w3.org/TR/WCAG22/>

web.dev Learn Accessibility: <https://web.dev/learn/accessibility>

MDN cross-browser testing:

https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Testing/Introduction

A DEMO IS NOT A SECURITY AUDIT.

Why shipping even a tiny app brings responsibility.

Static, local-first teaching projects have smaller attack surfaces than apps with login, private data, money movement or databases. But “uses no backend” is not proof that all future extensions are safe. Think about the additional risks whenever a starter becomes a service.

Threats to revisit if the project grows

New capability	New question
Accounts and roles	Are access controls enforced on a trusted server?
Personal data	Can it be avoided? Where is it sent, stored and deleted?
Online payments	Who handles authorization, fraud, disputes and PCI scope?
Third-party scripts	Do you know the provider, version and data exposure?
Deployment pipeline	Who can change code and how do you roll back?

CLIENT-SIDE RULES HAVE LIMITS

Validation in a browser improves the user experience, but browsers are controlled by visitors. Server-side rules must enforce permissions and business constraints in any application with a backend. See OWASP Top 10:2025 and MDN client-side form validation.

In this workshop

There is no input upload, live AI API, analytics library, backend account, order persistence or remote code execution. The preview runs in a sandboxed iframe without same-origin permission. User-supplied descriptions become plain text in export notes, not executable source. The handoff still requires a real review before repurposing the starter.

A 15-MINUTE PRACTICE SESSION.

An exercise with observable outputs rather than points.

Round 1 - choose and predict (3 minutes)

Choose Number Desk, Show & Tell or Kirana Counter. Before looking at the known flaw, write what the first draft might fail to communicate. For what user? What would be a misleadingly successful-looking result?

Round 2 - observe and repair (6 minutes)

Run the illustrative checks, try the runnable preview, open the source, and explain the missing rule in your own words. Apply the provided fix and rerun. Was the changed condition smaller or larger than you expected?

Round 3 - hand over (6 minutes)

Export the starter ZIP. Extract it. Open index.html in a browser. Use keyboard only for the main path. Write one new test the workshop did not include and one change you would require before inviting a real person to rely on it.

A MINI EXPERIMENT RECORD

Intended behaviour: _____

Observed behaviour: _____

Competing explanation: _____

Repair applied: _____

Still unverified: _____

If an issue survives the repair, keep it as evidence, not embarrassment. A reproducible surprising result is more valuable than a decorative claim of completeness.

EVIDENCE, HANDOVER, & NEXT QUESTIONS.

Keep sources, responsibilities and uncertainty attached to the artifact.

Carefully selected reading

Stack Overflow Developer Survey 2026 - AI data

<https://survey.stackoverflow.co/2026/ai/data>

Michels et al. (2026), state-of-the-art review [preprint]

<https://arxiv.org/abs/2608.20446>

OWASP Top 10:2025 - web security overview

<https://top10.owasp.org/2025/>

MDN - client-side form validation

https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Forms/Form_validation

W3C - WCAG 2.2 recommendation

<https://www.w3.org/TR/WCAG22/>

MDN - introduction to cross-browser testing

https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Testing/Introduction

web.dev - Learn Accessibility

<https://web.dev/learn/accessibility>

Cloudflare Pages - Direct Upload documentation

<https://developers.cloudflare.com/pages/get-started/direct-upload/>

Connected Sahilverse worlds

The Responsible Builder - <https://thebuilder.sahiltayal.com/>

The Archive of Attempts - <https://attempts.sahiltayal.com/>

SYNAPSE - <https://synth.sahiltayal.com/>

The World Engine - <https://worldengine.sahiltayal.com/>

The selected atlas - <https://sahilverse.online/atlas/>

THE HANDOVER QUESTIONS

What did we actually build? Who is it for? Which exact test can disprove the happy story? What have we not checked? Who repairs it after the celebration?

Explore the interactive workshop at <https://vibecoding.sahilverse.online/> or contact sahil@sahiltayal.com. This guide was authored as an educational companion; the site and all reading links should be rechecked after public deployment.