Unified Infotech

CTO Playbook

Preventing Software Failure Without Overengineering

SCROLL DOWN

What’s inside this kit

01. Six-week derisk plan

02. Vendor selection scorecard

03. Weekly steering meeting agenda

04. Decision log for irreversible decisions

05. Contract checklist to prevent scope creep

Use these templates before committing to scope, architecture, or vendors.

How to use this kit

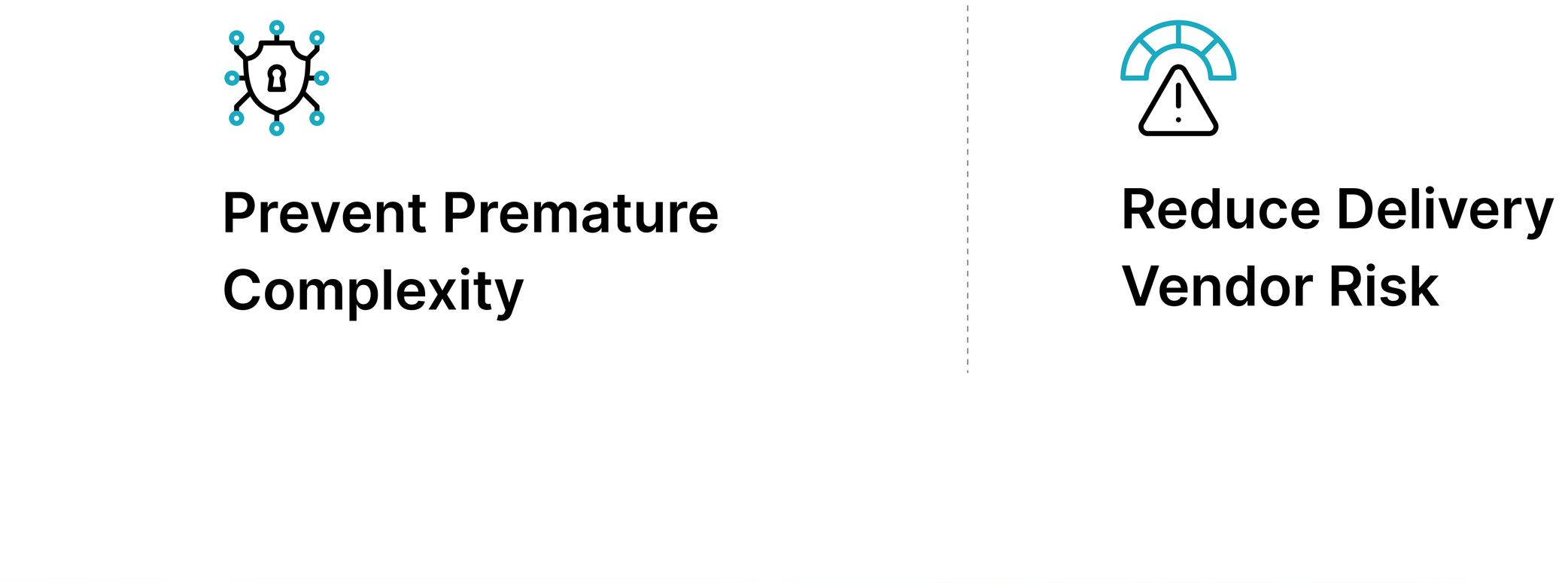
This kit is designed to be used before and during a custom software build.

Validate Reality Early

Keep Decisions Reversible

Prevent Premature Complexity

Reduce Delivery & Vendor Risk



The Six-Week Derisk Plan

Use this plan before committing to a full build, fixed scope, or long-term vendor engagement.

WEEK	FOCUS	OUTPUTS
Week 1	Alignment and constraints	KPI tree, success metrics, stakeholder map, RACI, constraints, definition of done
Week 2	Workflow and data reality check	Current state process map, data owner list, sample data quality report, integration inventory
Week 3	Thin slice scope	One end-to-end workflow scope, acceptance criteria, measurable value hypothesis, and demo plan
Week 4	Risk spikes and feasibility	Prototype risky integrations, confirm performance assumptions, validate security constraints, and edge case list
Week 5	Backlog and delivery plan	Prioritized backlog by ROI and risk, release plan, QA approach, environments, and CI/CD expectations
Week 6	Vendor alignment and contracting	Milestones tied to demos, change control rules, handover terms, support model, and governance cadence

Output at the end of Week 6:

- Prioritized backlog
- Risk register
- A delivery plan that can be defended

Risk Register Template

At the end of Week 1, you'll have a prioritized risk register that identifies potential obstacles. Use this template to start tracking your project's risks from the very beginning.

RISK	WHY IT MATTERS	PROBABILITY	IMPACT	MITIGATION OWNER & NEXT STEP
No empowered product owner	Decision latency increases rework and idle time			
Unclear success metrics	Teams ship outputs without proving outcomes			
Integration unknowns	Late surprises cause timeline and cost shocks			
Data quality issues	Bad data breaks workflows and adoption			
Scope creep without change control	Budget blowouts and missed deadlines			
Overengineering early	Slow delivery, higher maintenance, brittle system			
Security gaps discovered late	Rework and launch delays, increased risk exposure			
Adoption and training ignored	Low usage, shadow processes persist			
Unrealistic timeline	Quality tradeoffs, technical debt, morale loss			
Vendor dependency lock-in	High switching cost, reduced leverage			

Vendor Scorecard

Score each category from 0 to 5 based on what the vendor demonstrates, not what they promise.

CATEGORY	WHAT GOOD LOOKS LIKE	SCORE (0 TO 5)	NOTES
Discovery Discipline	Sells paid discovery; produces backlog, risks, estimates; does not “**guess**” fixed scop		
Outcome Thinking	Builds KPI tree; links features to measurable value; challenges low ROI requests		
Architecture Pragmatism	Defaults to simple architecture; justifies complexity; documents tradeoffs		
Delivery System	Frequent demos; CI/CD; clear environments; predictable cadence		
QA & Release Confidence	Test strategy, automation where it matters, clear acceptance criteria, and release checklist		
Security Hygiene	Threat modeling basics; least privilege; secure coding; dependency hygiene; logging		
Communication & Escalation	Clear status; risks surfaced early; defined escalation path; no surprises		
Reference Relevance	References in similar SMB constraints and complexity, not just big logos		
Handover Readiness	Readable code, docs, runbooks, onboarding, repo hygiene		
Cost Transparency	Rate card and roles clear; change control pricing transparent; no hidden role inflation		

Low scores in discovery or architecture almost always lead to overengineering later.

Steering Weekly Agenda

Use this agenda to keep delivery focused on outcomes instead of activities.

Meeting length: **30 to 45min**

Cadence: **Weekly**

5 min

Outcomes update

What measurable value moved since last week

10 min

Demo highlights

What is now working in the product

10 min

Delivery update

Next milestone, blockers, forecast confidence

10 min

Top risks (up to 3)

Owner, mitigation, decision needed

5 min

Decisions log

Confirm irreversible decisions and owners

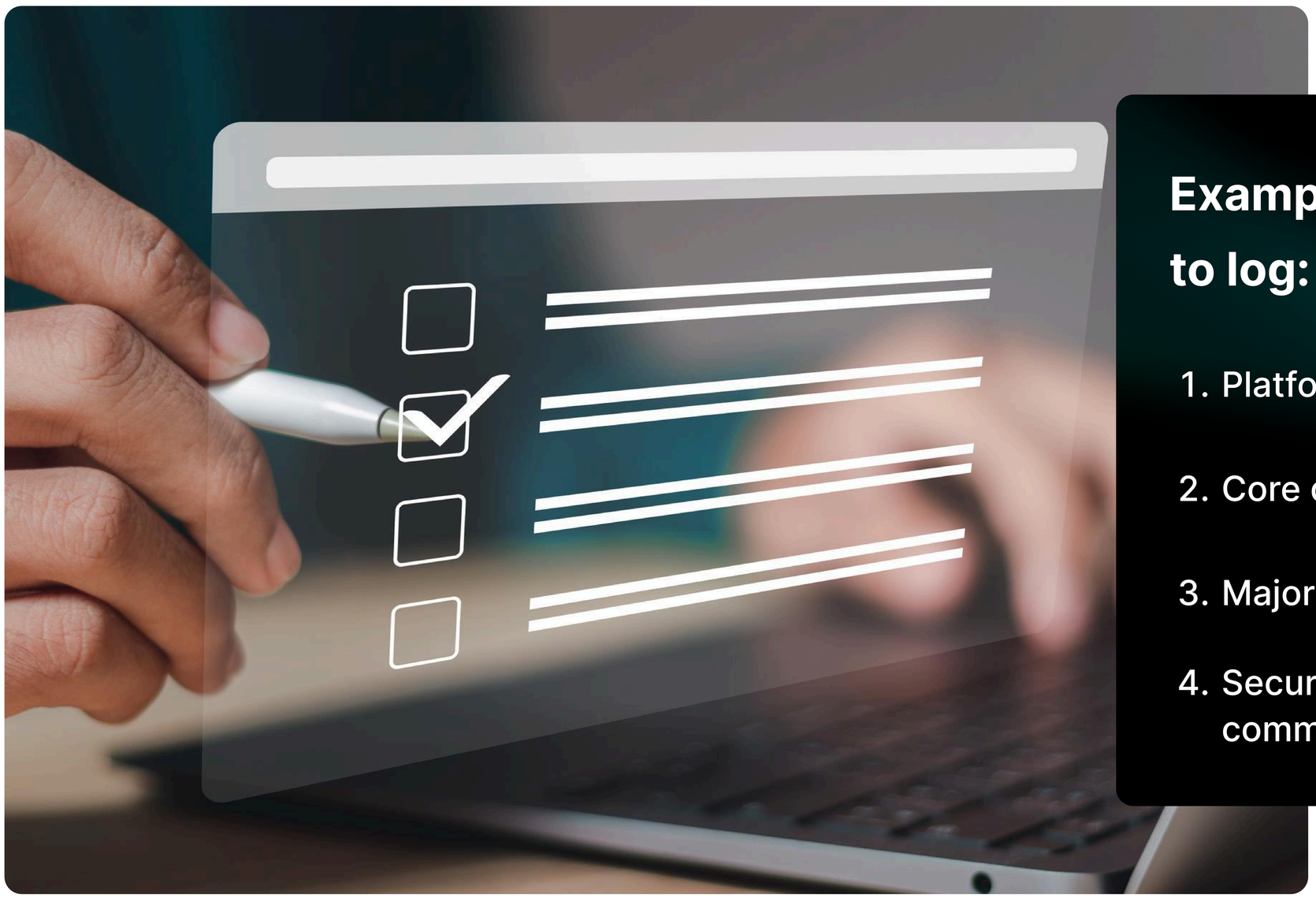
If a metric does not change a decision, remove it.

Decision Log Template

Use this log only for decisions that are hard or expensive to reverse.

Do not log the everyday delivery decisions.

DATE	DECISION	CONTEXT	TRADEOFF	OWNER



Examples of decisions to log:

1. Platform selection
2. Core data model
3. Major integrations
4. Security or compliance commitments

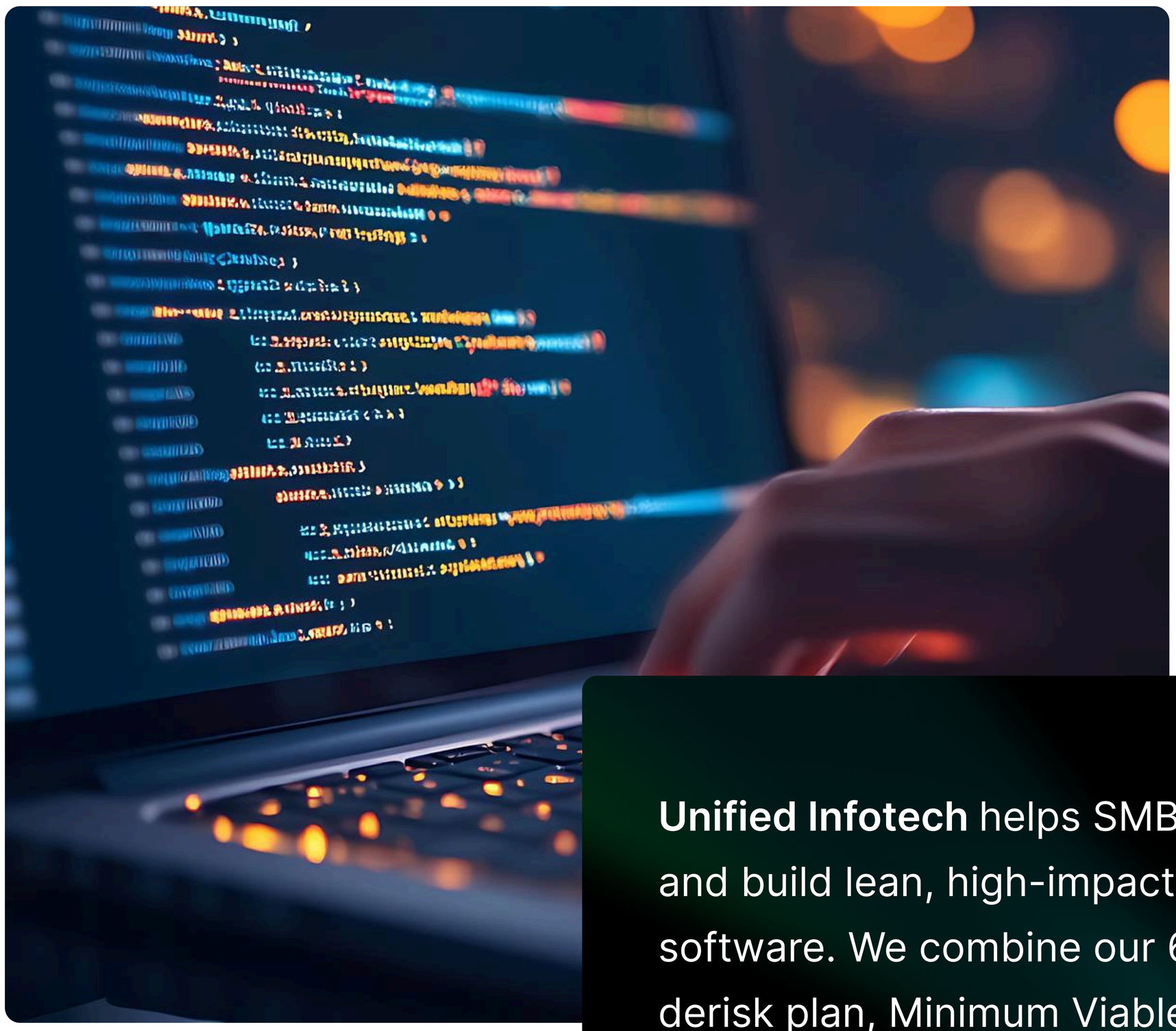
Contract Checklist to Prevent Scope Creep

Use this checklist before signing any software development contract.

CLAUSE OR RULE	WHAT TO INCLUDE
Paid discovery is a separate phase	Deliverables: backlog, risk register, estimates, roadmap, architecture guardrails
Milestones are tied to working software	Each milestone requires a demo and acceptance criteria
Change control is explicit	Scope changes include impact on timeline and budget before approval
Access and ownership are clear	Repos, environments, credentials, and IP ownership defined
Handover is contractually required	Runbooks, docs, onboarding sessions, and code quality expectations
Termination is defined	Clear exit terms, partial payments, and handover obligations
Support model is defined	Post launch SLA expectations and incident handling
Security responsibilities are explicit	Secure coding, penetration testing responsibility, and vulnerability management

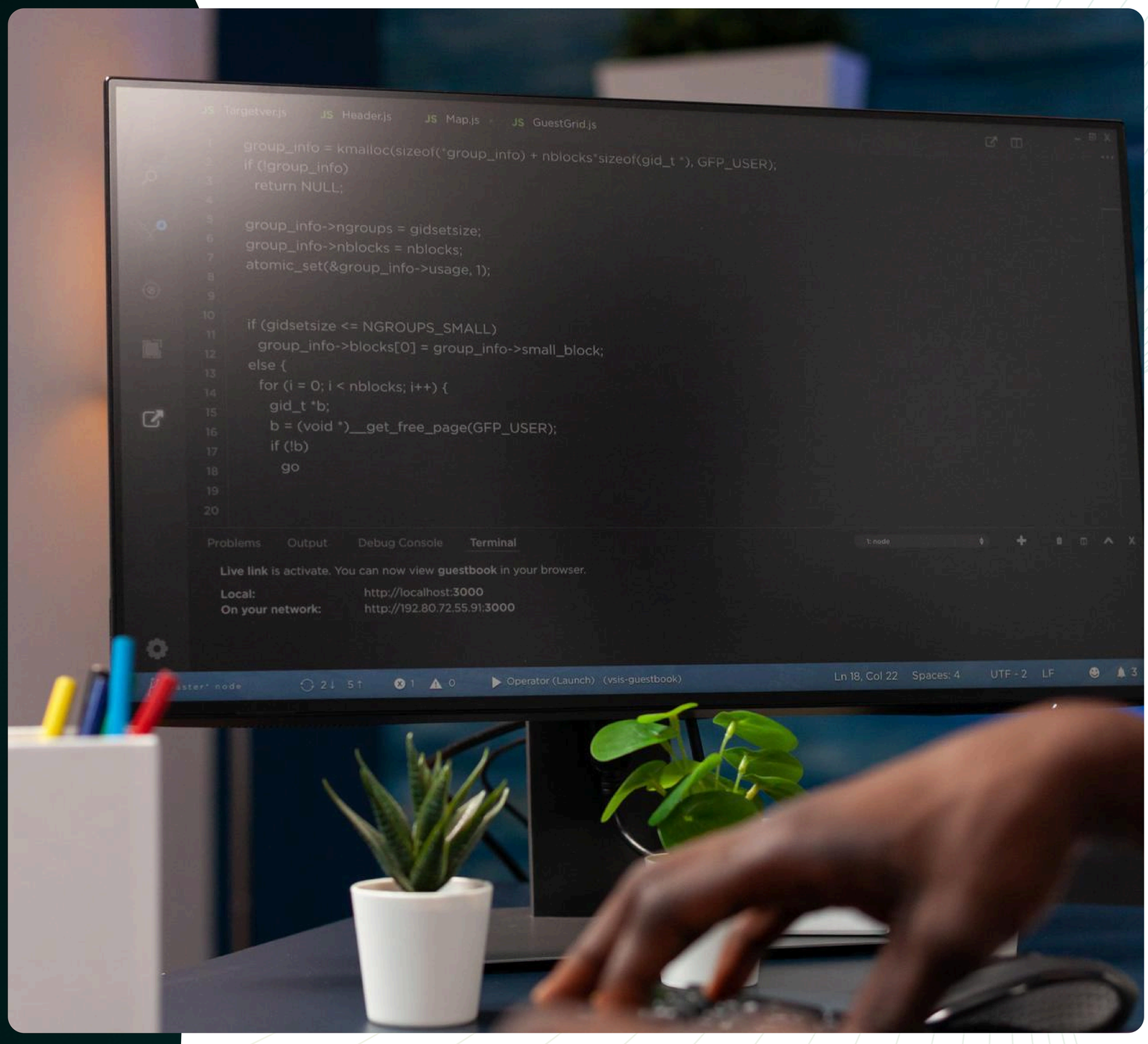
If a clause feels heavy, it usually exists because a previous project failed without it.

How Unified Infotech Supports SMB CTOs with Custom Software Development



Unified Infotech helps SMBs design and build lean, high-impact custom software. We combine our 6-week derisk plan, Minimum Viable Governance, and modular architecture to ensure your project stays on track. Our focus is measurable business outcomes, not experimentation theatre

Next Step



Most software projects do not fail because of poor engineering. They fail because reality shows up late.

This kit exists to surface reality early, keep decisions reversible, and prevent complexity before proof. Use it before you commit.

THANK YOU