A SAAS BUSINESS FRAMEWORK

Beyond Bedlam

A Practical Guide to Designing, Building, and Scaling a SaaS Business

Brent Golemon

"The cessation of digging is only the first step. There are then an infinite number of sequential options to get out. But there are tools, and a framework."

To My Family

An entrepreneur's answer to prayer

— BG

Table of Contents

INTRODUCTION

Foreword

Introduction

PART I — STAKEHOLDER INPUT — Chapters 1–16

Ch. 1 — Sales Team

Ch. 2 — Operations Team

Ch. 3 — Customer Support Team

Ch. 4 — End User Operational Processes

Ch. 5 — Sales Process

Ch. 6 — Marketing / Public Relations

Ch. 7 — Customer Support Processes

Ch. 8 — Current Tools & Technology

Ch. 9 — Startup SaaS Business Model

Ch. 10 — The Case for Custom SaaS

Ch. 11 — Legacy Software Obstacles

Ch. 12 — Retooling Your SaaS Service

Ch. 13 — The Market

Ch. 14 — The Goals

Ch. 15 — Compliance

Ch. 16 — Service Features

PART II — VALUE ANALYSIS — Chapters 17–26

Ch. 17 — Key Accounts

Ch. 18 — Marketing

Ch. 19 — Sales Funnel

Ch. 20 — Packaging & Pricing

Ch. 21 — Legal & Contracts

Ch. 22 — Competition

Ch. 23 — IT Infrastructure

Ch. 24 — DevOps
Ch. 25 — Measurements (KPIs)
Ch. 26 — Goals

PART III — INFRASTRUCTURE ALIGNMENT — Chapters 27–42

Ch. 27 — DevOps
Ch. 28 — UI / UX
Ch. 29 — Hosting ★ New
Ch. 30 — Scalability
Ch. 31 — National v. International
Ch. 32 — Third Party APIs
Ch. 33 — Software Integrations
Ch. 34 — Development Timeline
Ch. 35 — Software Features
Ch. 36 — Quality Assurance (QA)
Ch. 37 — Bug Tracking
Ch. 38 — Redundancy & Disaster Recovery
Ch. 39 — Target Markets
Ch. 40 — Marketing Automation
Ch. 41 — Pricing Format
Ch. 42 — Cash Flow

PART IV — FRACTIONAL DECISIONS — Chapters 43–45

Ch. 43 — Sales Contracting
Ch. 44 — Support
Ch. 45 — Development Team

PART V — MEASURED IMPLEMENTATION — Chapters 46–52 ★ All New

Ch. 46 — Management Team
Ch. 47 — Cash Flow Analysis
Ch. 48 — Proof of Concept Roadmap
Ch. 49 — Technology Roadmap
Ch. 50 — Point Analysis
Ch. 51 — Sales & Marketing Plan
Ch. 52 — Implementation Timeline

Foreword

In 1993, I co-founded my first Software as a Service business — though nobody had yet invented that term. I was carrying a laptop from office to office in downtown Austin, plugging into fax machine phone lines to demo a legislative tracking platform over a 54Kb dial-up connection. I sold almost as much AOL in those days as I sold actual software subscriptions.

What followed was a nearly thirty-year entrepreneurial journey through the full emotional spectrum of building software businesses. I've been in management at several SaaS startups, stood at the helm while turning around a struggling platform, and led teams through the painful process of migrating legacy software to the cloud. Along the way I managed development, customer service, sales, and marketing teams; helped raise capital; and negotiated the terms of two international mergers.

None of these services are household names, but many grew into strong businesses generating millions in annual revenues. For every success, however, there is a string of failures. I have spent many nights wondering whether our service would come back online. I have dealt with the loss of customer data. I have watched revenues shrink due to poor packaging and pricing decisions. I negotiated with the FBI to protect client confidentiality. I have suffered through third-party development companies that missed deadlines, overcharged, and underdelivered. I have been part of marketing and sales efforts that were, in a word, disastrous.

Does any of this sound familiar? If so, this book is my attempt to tell you there is a light at the end of the tunnel — and it does not have to be a train.

What I've learned is that starting a SaaS service, or transitioning legacy software to a SaaS model, is more science than art. There are systematic steps to be taken. There is no quick fix, no silver bullet. What there is, is a process — one that demands dedication and honest, sometimes brutal, self-reflection, which will inevitably require some hard choices.

There are thousands of books on how to improve your business skills. This is not one of them. I did not write it to make you a better leader, a better recruiter, or a better closer. I wrote it to share a process that works for anyone charged with moving a business forward by leveraging Cloud technology. Think of it as a

guide, and therefore a quick read — a framework from which to navigate your own SaaS journey.

Throughout the book, I draw on my experience working with entrepreneurs, business owners, practice managers, and other leaders who have reached the same realization: their target market will leave them behind if they don't harness the power of the Internet to create a platform that empowers their people and serves those they exist to help.

I want to thank the thousands of mentors, co-workers, partners, and clients who have helped me better understand what it takes to conceive, design, and implement a successful SaaS business. No idea or organization is ever better than the people tasked with making it run — and I have been blessed to work alongside some remarkably talented people.

Finally, my deepest passion is coaching football. For over forty-five years I have been playing, officiating, or coaching the game in Texas. To understand Texas football coaches, you have to understand the power of story. Some coaches excel at player development, others at strategy or team management, but all of us are fluent in what I call "Coach Speak" — the art of using humor and story to drive a point home. I hope this book meets you where you are in your journey and brings some clarity to whatever bedlam you may be experiencing in your business.

FRONT MATTER

INTRODUCTION

Introduction: The First Rule of Holes

""The first rule of holes: when in one, quit digging.""

— Molly Ivins

I founded my first SaaS company with my high school best friend. He had graduated from the University of Texas with an Electrical Engineering degree and had been working in the family computer business since he was fifteen. We hatched the idea sitting in a hot tub at a mutual friend's party and would soon discover that inspiration born in hot tubs is far inferior to inspiration born in showers.

The early years were a relentless cycle of two steps forward, one step back. I often told people our business model was so good that even we couldn't kill it — like the plant your mother gives you when you leave for college, the one you never water that somehow survives the semester. She nurses it back to health over the summer and hands it back to you hoping you've learned something about botany. You haven't, of course. You proceed to water it with beer.

The turning point came in a conference room heated argument we'd been having for over an hour — one of those circular debates about an issue we deemed monumental at the time. A Board member from Lebanon named Tony had been sitting quietly through the whole affair. Finally, when it was clear we'd made no progress, I asked Tony for his opinion. His response became a watershed moment for me. He simply asked, "Brent — what is the first rule of holes?"

The first rule of holes is that when you're in one, you stop digging. This is sage advice I've carried with me and shared with fellow travelers throughout my years as a business owner, investor, coach, and consultant. I find myself returning to it often, asking which end of the shovel I'm holding.

When we are digging in our business, we rarely take time to plan. We are oblivious to the unnecessary holes we're creating, unaware that working harder and faster is only making things worse. As Benjamin Franklin put it, "If you fail to plan, you are planning to fail."

Stopping the digging is only the first step. It doesn't remove you from the hole you're already in. What it does is create the space for a sequential plan of escape — and no two paths will ever look the same. However, there are tools and a framework from which to choose the best route for each situation.

What is certain is that inaction is itself a form of action. Staying in the hole may feel safe in the short term, but it will eventually become your undoing. The Beyond Bedlam process is the construct for a roadmap out. It does not guarantee success, and it doesn't presume to answer every specific question. What it does is address the main issues to be considered on the journey, provide real examples, and suggest overarching principles to guide your thinking.

The journey is organized into five sequential steps: Stakeholder Input, Value Analysis, Infrastructure Alignment, Fractional Decisions, and Measured Implementation.

The components of each step are enumerated and illustrated with examples drawn from my own SaaS experiences. I've offered considerations for each topic — but they are general and by no means exhaustive. Like any process, this one is constantly being refined and adjusted to account for changes in technology and the marketplace.

Where possible, I've deliberately avoided endorsing specific technical solutions, since the shelf-life of any particular recommendation is short. Instead, I've focused on the underlying issues that must be confronted. For more current and specific guidance, I'm building that out through the blog at www.beyondbedlam.com — a format better suited to the pace of change in this industry.

PART I

Stakeholder Input

Chapters 1–16

Before a single line of code is written, the most valuable intelligence in the organization lives in the people closest to the problem. This section is about listening before building.

PART I — STAKEHOLDER INPUT

CHAPTER ONE

The Sales Team

""Stop selling. Start helping.""

— Zig Ziglar

I started selling my first SaaS service in 1993 by going office to office in downtown Austin. The term "SaaS" hadn't been coined yet. I was lugging a laptop and plugging it into fax machine phone lines at each prospect's desk to demo our product over a 54Kb dial-up connection. I sold almost as much AOL in those days as I sold legislative tracking subscriptions.

What I did learn was that salespeople knew the pulse of the market better than anyone. First-hand, you heard why a prospect would never buy this product — or why they just might. The criticisms and insights our sales team brought back built GalleryWatch into a company that would one day raise $8 million in investment capital and ultimately be acquired by a Fortune 500 company.

When assessing the pros and cons of a SaaS service's viability and refining its value proposition, it is crucial to get the perspective of your sales team. One of your first steps is to sit down with sales team members and flush out what problems the end user is actually trying to solve. What is their pain? How are they managing it currently? Where does the current solution fall short? And what would cause them to leave for something better?

It is critical to distinguish sales feedback from support feedback. Sales feedback reveals the pains of prospective buyers — it should never be conflated with customer support, which is a post-sale concern. One focuses on what will get a prospect to become a subscriber; the other is about delivering on the value you promised once they are.

A defining example occurred when GalleryWatch opened its Washington, D.C. office in 1997. I was interviewing candidates the same week a leading competitor — a Washington Post subsidiary — had folded. I immediately began interviewing their displaced sales, marketing, and customer support staff. That's when I met three-martini Arnie.

Arnie had run customer support and had previously been the lead salesman for our soon-defunct competitor. The most buttoned-up individual I had ever encountered — and I say that having spent a year in London, where rigidity was practically a cultural currency — Arnie's prim exterior concealed a very cunning mind. He helped me understand exactly what kind of sales team we needed to build.

His caution was against seasoned veterans who were too invested in the legacy technology to hear what the market was demanding. Instead, he recommended we seek people who genuinely embraced emerging technology and had connections with prospects who were apt to adopt SaaS-based solutions early.

In the end, GalleryWatch gained its foothold in the federal market because we listened to what prospective clients were telling our sales team. It turned out some of our least successful state-market features were the strongest draw at the federal level. We adjusted our marketing and our development roadmap accordingly.

The same dynamics apply when transitioning from on-premise software to the cloud. Those who will be selling your platform must be genuine advocates of it — or they will offer more excuses than constructive insights.

ROADMAP CONSIDERATIONS

- ◆ Are the people you are hiring to sell genuine advocates of your service?
- ◆ **Does your sales team sell the software** — or do they sell a solution to the prospect's actual problem?
- ◆ Are sales team members included in DevOps conversations about new features?
- ◆ **Consider pulling outside sales hires from your customer support team** — they already know the product and the customer.
- ◆ Train your customer support team to identify and act on upsell opportunities.

PART I — STAKEHOLDER INPUT

CHAPTER TWO

The Operations Team

""Through great input, you get great output.""

— RZA

Input-output analysis is a macroeconomic method that observes the interdependencies between sectors of an economy — how a shock to one industry ripples into others. It is important because it reveals the hidden connections that a surface-level view will always miss.

The same principle applies when building your SaaS roadmap. The operations team is one of the most valuable — and most overlooked — sources of input. End users often lack purchasing authority, but they carry enormous influence over the purchasing decision and, more critically, over whether the service survives in the field once it's deployed.

I worked with a lawn care company that had developed an internal legacy software to track crew locations, schedule jobs, manage task completion, and handle accounting. We were hired to migrate the system to a cloud-based SaaS platform. When I asked to speak with the operations team, the owner directed me toward office management. I pushed back. I wanted the people in the field.

The request created visible discomfort. After a pause, the owner explained that only crew managers used the application because they were the only staff issued mobile devices. I pointed out that the new application would run on any smartphone, meaning anyone on the crew could make updates. He had genuinely never considered that possibility. He then noted that most of the crew couldn't read and that many didn't speak fluent English.

What I heard from the field workers was revelatory. The existing workflow was actively obstructing the crew's ability to complete their work. One example: the company had a single hedge trimmer, and on certain routes, an entire crew had to wait for one task to finish before moving to the next property. The application had no way to split a job into separate tasks executable by different crew members at different times.

The solution was to break tasks apart, allow crew members to update the app using graphic-based icons to overcome the language barrier, and add geo-positioning to coordinate between crews. The result was nearly a 30% improvement in task completion time — which translated directly into additional revenue. It also enabled the company to offer previously jettisoned services, such as rose pruning, that had been impractical under the legacy platform.

ROADMAP CONSIDERATIONS

◆ **Seek to understand before being understood** — Steven Covey's principle is as relevant in software development as anywhere else.

◆ Listen in person or over video, where body language and hesitation reveal as much as words.

◆ **Share a clear vision of the new platform's capabilities upfront** — it expands the creative range of the feedback you receive.

◆ Interview stakeholders both as a group and individually. Group settings surface the "unspoken" issues that individuals might not raise alone.

◆ Filter ruthlessly. Use only the most pertinent feedback relative to the actual project scope.

◆ Manage expectations. Stakeholders can develop unrealistic hopes quickly. Dispel false assumptions early.

◆ Always end on a positive note. You will almost certainly need to return to these people for additional input.

PART I — STAKEHOLDER INPUT

CHAPTER THREE

The Customer Support Team

""It takes months to find a customer and seconds to lose one.""

— Vince Lombardi

We had a saying at my IT company: "Those who say the customer is always right have never worked in IT." All businesses have demanding clients, but none are more trying than the ones with a direct line to your personal cell phone — and no demographic perfects this art quite like family members who are also subscribers to your service.

In the nineties, I co-founded the first company to track legislative information on the Internet. My father, a longtime and well-respected lobbyist, became our first subscriber. His patronage was crucial in a niche market deeply skeptical of this new-fangled SaaS model. For over a year, things operated smoothly.

Then came the 5:00 a.m. phone call. My father was furious that he couldn't load a document in the portal. We worked through every troubleshooting step — including the dreaded "is the computer plugged in?" inquiry, which never carries an upside.

The staffer eventually conveyed the problem with the careful diplomacy of a man walking a very thin wire. The service was almost certainly working fine. The issue was the monitor. My father had been using the screen as a surface for his Post-it notes. Someone had replaced his actual monitor with a realistic-looking plastic display prop — the kind used for marketing — and put the sticky notes back. I suggested the young man quietly pilfer a monitor from a junior partner's office and take the knowledge of the plastic monitor to his grave.

Every customer support team is a library of these stories. And beneath the humor are genuine data points — buried in ticket systems, embedded in interviews with support staff, and latent in conversations with key accounts. Some of the best intelligence for designing or retooling a SaaS platform lives in your support operation, waiting to be mined.

The final shape of your SaaS service must include an efficient system for reporting, auditing, and resolving issues. It must also reflect a clear-eyed assessment of what you'll have to support in order to maintain a high level of customer satisfaction over time.

WAYS TO GATHER CUSTOMER INTELLIGENCE

- **Use existing data.** Ticket volume, category, and resolution time tell a story before you ask a single question.
- **Solicit structured customer feedback.** Surveys, NPS scores, and post-resolution follow-ups reveal patterns the individual ticket view misses.
- **Customer journey mapping. Walk the entire experience from awareness through support resolution** — from the customer's perspective, not yours.
- **Direct input from the service team.** Front-line support staff are often the most honest voice in the company about what is and isn't working.
- **Competitive analysis. Study what your competitors' customers complain about publicly** — those are your opportunities.
- **Social media listening.** What are people saying in community spaces, forums, and review platforms about your category?
- **Keyword research.** The questions people search for around your type of product are often the questions they never think to ask you directly.

PART I — STAKEHOLDER INPUT

CHAPTER FOUR

End User Operational Processes

""The customer is not always right, but they are always the customer.""

— Shep Hyken

Any organization, regardless of type or size, cannot survive without genuinely meeting the needs of its customers. There's a principle in marketing worth tattooing somewhere visible: customers come to a product based on what they want, but they stay based on what they need.

I worked with a company that was among the first to put newspaper clips on the Internet. Before their SaaS solution, employees would arrive early each morning to physically cut articles from print editions, paste them onto paper, and scan them into PDFs for client email delivery. The new SaaS platform made those clips searchable for the first time, presented them in both PDF and plain text, allowed for cut-and-paste access to article snippets, and offered organized archives with a host of other features.

We thought we had hit it out of the park. What more could a subscriber want? Quite a lot, it turned out. Because we had failed to ask how the subscribers were actually using the data. The majority of end users were not sitting at their desks during the day — they were out in the world, and they wanted a single email they could open during brief windows of downtime, in locations that often had no reliable Wi-Fi access.

The solution was straightforward: add a daily digest email that bundled each subscriber's search results. But the near-disaster was entirely preventable. We had simply failed to understand how users were interfacing with the service in the real world. The SaaS build cannot assume that end users will abandon their existing workflows simply because your platform is objectively better in other dimensions.

THE MOST COMMON CUSTOMER NEEDS

- **Functionality** — Your service must actually solve their stated problem.
- **Price** — Everyone has a budget. Don't build a Cadillac for a customer who needs a reliable pickup truck.
- **Convenience** — If it requires more steps than the current process, adoption will stall.
- **Experience** — Reducing time and complexity is often more valuable than adding features.
- **Reliability** — Be available. Every time. Without exception.
- **Performance** — Work exactly as advertised, at scale.
- **Compatibility** — Play well with the other software already in the customer's world.
- **Empathy** — Support staff who understand the user's context and frustration.
- **Transparency** — Communicate the good, the bad, and the ugly with equal clarity.
- **Control** — Make it easy to both onboard and, if necessary, leave.

PART I — STAKEHOLDER INPUT

CHAPTER FIVE

The Sales Process

""If you can't describe what you are doing as a process, then you don't know what you are doing.""

— Edward Deming

No two software sales processes are alike, and it is critical to understand how new software development will integrate — and potentially enhance — the existing sales workflow. Features and design decisions made early in the development process can either accelerate the sales cycle or silently obstruct it.

To illustrate this, I had a client offering a franchise service supported by several SaaS tools. In reviewing their organization, I discovered they had no phone number on their website, no company email address publicly listed, and — remarkably — no way to contact them at all until after you'd already purchased. This was not an oversight. This was the architecture.

The company operated in 22 countries and had built their entire sales process around forced self-qualification. Prospects had to navigate an automated funnel where their questions were answered through curated content. The friction was intentional: they wanted franchisees who would walk over broken glass to start a franchise.

For their development team, this meant everything had to be fully automated with zero human touchpoints. Every element had to integrate seamlessly with their CRM, marketing campaigns, and content management systems. There was no room for features that assumed manual oversight, because manual oversight simply didn't exist in their model.

That is an extreme example, but it illustrates the principle clearly: your SaaS strategy must account for how the service is ultimately sold — not how you imagine it will be sold. Free trials, partial feature access, usage-based onboarding, and other go-to-market tactics all need to be baked into the architecture from the beginning.

ROADMAP CONSIDERATIONS

- ◆ What CRM is being utilized by the sales team, and how will the platform integrate with it?
- ◆ What is the sales model? Hunter/Gatherer, Account Management, PLG (product-led growth), or something else entirely?
- ◆ What other key software is embedded in the sales process that the new platform must touch?
- ◆ **What is unique about the target market's decision-making process** — procurement cycles, gatekeepers, committee approvals?
- ◆ Are there regulatory or contractual constraints that affect how the product can be demonstrated or trialed?

PART I — STAKEHOLDER INPUT

CHAPTER SIX

Marketing & Public Relations

""People do not buy goods and services. They buy relationships, stories, and magic.""

— Seth Godin

One of the most frequently overlooked parties in SaaS development is the Influencer — the individual or group that isn't a paying customer but whose goodwill and endorsement can make or break the platform's early adoption. Understanding who these people are, and designing for them, is one of the highest-leverage decisions you can make early in the process.

When we started GalleryWatch, our goal was to give legislative professionals — lawyers, trade associations, lobbyists, and government affairs departments — access to data they had always wanted but found nearly impossible to obtain. At the time, committee substitutes were the Holy Grail. A committee substitute replaced the text of the original bill in committee, often substantially different and sometimes hundreds of pages longer. Politicos needed to see these substitutes in advance, but they weren't officially part of the public record and therefore weren't available through open records requests.

In our first session, GalleryWatch got doors slammed in our faces through official channels. So we tried something different: old-fashioned bartering. We developed a tool inside our SaaS platform that assisted committee clerks with their own workflows and made their jobs meaningfully easier. We offered it to them for free. And we brought donuts in the mornings and pizza in the evenings.

Word spread among the committee clerks faster than legislation. Soon they were using our SaaS service more than our paying subscribers were. The more access we gave them, the more cooperative they became with the committee substitutes. Our ability to provide this data led to a doubling of our user base in three months and secured our path to profitability.

ROADMAP CONSIDERATIONS

◆ **Who are the influencers in your target ecosystem** — the people whose adoption accelerates everyone else's?

◆ How can influencers be meaningfully incentivized to become advocates of your platform?

◆ Are there third-party applications or integrations that, if incorporated, would make you indispensable to this influencer group?

PART I — STAKEHOLDER INPUT

CHAPTER SEVEN

Customer Support Processes

""A good process produces good results.""

— Nick Saban

Customer needs analysis is a strong starting point for understanding your customer, particularly around common pain points. But understanding those pains is only half the work. The other half is ensuring the right processes and the right people are in place to address them — especially during the high-stress period of a SaaS transition or platform retooling.

Bubba was a twenty-five-year employee of a company I had been hired to move to the cloud. He had been the principal architect of the legacy system and, it became clear within days, the only human being on the planet who truly understood it. Bubba was the only one who could onboard a new client. Bubba was the only one who could touch the code. Bubba handled almost every tech support call, and most of the client base had his personal cell number. If you looked up "key man risk" in the business dictionary, you would see a photograph of Bubba looking tired.

To create a timeline for developing and implementing a new SaaS service required his involvement — which meant first liberating him from his current obligations. To "Free Bubba," we instituted a new tiered support structure. A premium "white glove" tier was introduced at four times the previous annual service charge and provided dedicated phone support and proactive monitoring. The existing tier was repositioned as email-only. Clients not on either plan were moved to a high hourly rate for ad hoc support. The result was that Bubba's calendar opened up, the company generated additional revenue from an already-installed base, and the support team grew to fill the operational gaps.

Having appropriate support processes and personnel in place is not a background concern — it is a foundational one. The complexity of your product will dictate the size and skill set of the team required to support it. Any plan that doesn't account for these costs is not really a plan.

BUILDING A PROACTIVE SUPPORT CULTURE

◆ **Offer consistent messaging.** When the whole company understands its goals, values, and product capabilities, that clarity flows directly to the customer experience.

◆ **Build structured onboarding.** A well-designed post-purchase experience enables your product to be useful from day one, not just eventually.

◆ **Nurture customer relationships proactively.** Retention work begins before any churn signal appears, not in response to one.

◆ **Build feedback loops into the support process.** Every problem is a disguised opportunity. A team that surfaces those signals systematically improves the product iteratively.

PART I — STAKEHOLDER INPUT

CHAPTER EIGHT

Current Tools & Technology

""You can't look at the competition and say you're going to do it better. You have to look at them and say you're going to do it differently.""

— Steve Jobs

As the world of apps and SaaS continues to evolve, cross-platform software integration has become the baseline expectation, not a premium feature. Gone are the days of the all-in-one platform that aspires to meet every need.

Copying a competitor is not a strategy for survival. Your solution ultimately needs a unique value proposition. But it is equally unrealistic to assume you must reinvent every wheel to remain competitive. In decades of SaaS work, I have never been part of a development effort where some element of the build wasn't drawn from an open-source framework or third-party foundation.

A real estate mortgage lending company engaged us to replace a series of complex, interconnected spreadsheets and Word documents that served as their internal workflow engine. As we worked through their deeply intricate workflow, we realized the process was an accretion of disappointments — layers of workarounds built on top of SaaS products that had never fully met their needs.

The breakthrough came when we discovered their lead analyst's process was actually an improvement upon a specific SaaS product currently in the market. We trialed that product, along with several others in the competitive landscape, and suddenly the conceptual language he had been speaking became clear. We weren't starting from zero — we were mapping improvements from a known baseline. The final solution exceeded the client's initial expectations with only marginal additional development cost.

ROADMAP CONSIDERATIONS

◆ What software in the marketplace already accomplishes comparable tasks? Have you actually used it?

◆ How will the software you are designing compete with or integrate alongside current solutions in your customer's stack?

◆ Could a lightweight middleware SaaS solution, connecting existing platforms, provide more value with less risk than a full build?

CHAPTER NINE

Startup SaaS Business Model

There are essentially four types of SaaS roadmaps: Startup, Legacy, Service, and Transition. There is certainly overlap in how you approach the planning process for each, but every type carries its own set of nuances when preparing an action plan.

Startup is by far the most common. I have been part of several successful ones — GalleryWatch.com, NewspaperClips.com — and an honest assortment of failures, including DinnerConnect.com, Ukumi, and others I've mercifully blocked from memory.

Startups are the hardest to map and the most demanding to execute. The reason is simple: most of the initial inputs are, to use the technical term, Wild Ass Guesses. What do we sell? Who will buy it? How do we reach them? How do we price it without losing our shirts on development and support costs?

Saaspreneur (noun, pl. saas·pre·neurs)

The perfect blend of arrogance and ignorance. The arrogance to think you can design and sell software online, and the ignorance of not yet grasping how difficult it truly is.

If that definition dissuaded you, consider yourself fortunate — you're not a Saaspreneur. For the rest of you knuckleheads: welcome to the tribe. I pray you learn from the mistakes scattered throughout these pages before you make your own variations of them.

STARTUP SURVIVAL PRINCIPLES

- ◆ Narrowly define your target market before writing a single line of code. "Everyone" is not a market.
- ◆ Design what the customer wants, not what impresses your development team.
- ◆ **Price high and discount when necessary** — it is always easier to come down than to go up.
- ◆ Fanatical customer service starts at the top. If the CEO doesn't model it, no one below them will sustain it.

PART I — STAKEHOLDER INPUT

CHAPTER TEN

The Case for Custom SaaS

""Engagement is when your customer is realizing value from your SaaS.""

— Lincoln Murphy

There is a persistent misconception in the market that SaaS platforms cannot be meaningfully customized — that you take what's available and learn to work around its limitations. The reality is the opposite. One of the greatest advantages of cloud-based architecture is the ability to build precisely what your business needs, at a cost that would have been prohibitive a decade ago.

A property management client had been subcontracting environmental researchers to manage ecological assessments on property under management. They already had a client-facing CMS portal, but they needed to expand it to handle contractor assignment, data collection, reporting, and billing for these researchers. The government required these studies to be completed on schedule, or their clients would forfeit tens of thousands of dollars in tax benefits.

What made the SaaS roadmap so compelling to their management team was the clarity of the cost-savings case. When you can show that a custom solution eliminates a specific labor cost, closes a compliance risk, and generates a new revenue line, the conversation moves from "can we afford this?" to "why didn't we do this earlier?"

The problem with off-the-shelf commercial SaaS is that one size never fully fits all. Core and proprietary business processes are often best delivered through a customized solution — one designed around how the business actually operates, not around how a software vendor imagines businesses should operate.

ADVANTAGES OF CUSTOM SAAS DEVELOPMENT

- **No installation overhead.** Cloud-native solutions require no on-premise hardware or IT provisioning.
- **Purpose-built features.** Include exactly what users need and nothing they don't.
- **No hardware dependency.** The infrastructure is managed, scaled, and secured at the platform level.
- **Centralized updates. New features and bug fixes deploy to every user simultaneously** — no version fragmentation.

PART I — STAKEHOLDER INPUT

CHAPTER ELEVEN

Legacy Software Obstacles

""Let's go invent tomorrow rather than worrying about yesterday.""

— Steve Jobs

Legacy SaaS migration involves taking proprietary software running on a local network and moving it to the cloud. The path to success is littered with obstacles — most of which reside not in the code itself, but in migrating the people and the culture alongside the software.

In one legacy point-of-sale migration, we needed the SaaS platform to interface with peripheral devices already in use at the client's locations. The development team was based in India. Many of the devices were old, boutique, or both — and couldn't be sourced in India. We shipped physical hardware to the development team. It was held in Indian customs for over four months while bureaucrats debated the correct valuation of equipment with no comparable reference point. I learned that government bureaucracy is universal.

As Mike Tyson famously observed, "Everyone has a plan until they get punched in the mouth." I am continually surprised when clients are surprised by the difficulties that emerge in a legacy migration. The transition almost always reveals additional features to add and legacy functions to rethink — both of which extend the timeline.

UNIQUE CONSIDERATIONS FOR LEGACY MIGRATION

- **Hardware audit.** Identify all current hardware that is not plug-and-play with a cloud-native solution. Budget accordingly.
- **Third-party software inventory.** Assess every piece of software currently used by end users, including legacy tools that may themselves be out of support.
- **Network architecture review.** Even though the SaaS solution is cloud-based, certain elements may still depend on the local network. Identify these dependencies early.

PART I — STAKEHOLDER INPUT

CHAPTER TWELVE

Retooling Your SaaS Service

""Sometimes the simplest things are the most difficult to achieve.""

— Keanu Reeves

I was sitting in a sushi restaurant in Düsseldorf, Germany, with the owners of an international SaaS service operating in eight countries. The acquisition ink had been dry for six weeks. I knew the hardest SaaS roadmap I had ever assembled was still ahead: merging two companies across half a world, running different technology stacks, with the shared goal of retaining enough of both subscriber bases to justify the deal.

What I had that I didn't have the first time around was a plan. And SaaS transitions require a laser-like focus on existing customers. Their core needs must be met through the transition or they will not make the switch — and they will tell everyone in their professional circle why.

The phone call came not long after dinner. A prominent association lobbyist and 16-year subscriber — nearing 80 years old, formidable as a force of nature — called to inform me she did not like Germans, and that if anything changed about how the site worked or what she received in her daily delivery, she would take her business elsewhere. I relayed her comments to my partners. They were immediately concerned, rightly noting that her demands were probably not achievable and that she had been, by any standard, quite rude. I told them she was the least of our concerns — she was a kitten compared to some of our other subscribers.

Eighteen months later we had retained over 85% of the subscriber base and grown revenues by more than 20%.

ROADMAP CONSIDERATIONS FOR MERGERS & RETOOLING

◆ **Infrastructure.** What adaptations are needed to preserve the key value propositions that prevent subscriber churn?

◆ **Product offering.** Will the retooling cause you to lose market share, expand it, or shift it? Each path requires different planning.

◆ **Growth model.** How will the change maximize bottom-line growth over a 12–24 month horizon?

PART I — STAKEHOLDER INPUT

CHAPTER THIRTEEN

The Market

""Everyone is not your customer.""

— Seth Godin

You cannot be all things to all people. Let your competition try to serve everyone — and watch them fail from a comfortable distance. One of the most powerful benefits of defining a precise target audience is that it clarifies your company's positioning and makes every downstream decision — from product design to marketing to pricing — vastly easier to execute.

I learned the real power of targeted markets when we opened our D.C. office. I had been lobbying our Board to enter Florida and California instead. Three weeks later my wife and I loaded two dogs into an SUV and headed to Washington with no product built for the market, no operational team, and very limited knowledge of how the federal legislative ecosystem worked.

Government agencies? We didn't have the financial stamina for federal procurement cycles. Trade associations? Almost everything we'd built for state-level associations had no relevance at the national level. Lobbyists? We discovered they didn't actually track legislation the way our platform assumed. Who was left? Legal librarians.

Legal librarians were often responsible for comparing different versions of legislation — a laborious process at the federal level where omnibus bills ran hundreds of pages. Our bill comparison tool handled this analysis automatically. We had come to D.C. to sell a robust legislative tracking service. We actually sold a $5,000 bill comparison tool with a robust set of bundled tracking features. The distinction matters — and it only emerged because we stopped assuming we knew the market and started actually listening to it. Eleven months and twenty-two days after we arrived, my wife and the dogs were able to drive back to God's country.

BEST PRACTICES FOR AUDIENCE DEVELOPMENT

◆ **Focus on the motives behind the behaviors** — not just demographics, but psychographics and situational pressures.

◆ Build personas that are fictional but realistic. A persona without a problem to solve is just a description.

◆ Choose one primary persona per product or service. Serve that one person exceptionally well before expanding.

◆ **Tell your persona's story from their perspective** — not from yours.

PART I — STAKEHOLDER INPUT

CHAPTER FOURTEEN

The Goals

""Tell less and ask more.""

— Unknown

I remind myself regularly that my advice is usually not as good as I think it is. It is very easy for consultants to parachute into a situation, misread the actual problem, and then turn around with solutions that address the wrong question entirely.

One of the more humbling experiences of my life was attending the University of Texas School of Law. Before you stop reading, know this: I am the second-best kind of lawyer. A non-practicing one. The best kind is, of course, a dead one.

The moment I knew I would never practice came in the second semester of my first year. The visiting professor — a co-author of our textbook — was teaching through Socratic dialogue. He called on Barry, a student so quiet he would never have spoken voluntarily. The exchange went back and forth for nearly ten minutes. At the end, the professor asked Barry why the class should side with him. Barry replied that his argument had been made by the visiting professor himself — in a research paper cited in the footnotes of the very reading we'd been assigned.

Let that sink in. We were assigned nearly 400 pages of reading per day, and Barry had not only read the footnotes but had tracked down and read the source paper referenced within them. The professor was visiting unexpectedly. Barry had not raised his hand. Barry set a standard of preparation I was not up to meeting. But his example permanently redirected my focus. I stopped trying to have the answers and started learning to ask better questions.

When building a SaaS roadmap, you must know the goals — and not just the ones written in the executive summary. We must start with the end state in order to prepare the path.

GOAL-SETTING FRAMEWORK

◆ **Define strategic, tactical, and operational goals as distinct layers** — don't let them bleed into each other.

◆ Choose objectives based on your unique strategy, not on what your industry conventionally pursues.

◆ Consider goals across all four perspectives: financial, customer, internal process, and learning and growth.

◆ Use the "Verb + Adjective + Noun" format to force precision. A goal should describe an action, a quality, and an object.

◆ **Write "strategic objective statements" that clarify intent** — not just what you want to achieve, but why it matters.

PART I — STAKEHOLDER INPUT

CHAPTER FIFTEEN

Compliance

""Red tape will often get in your way. That is why I often carry scissors.""

— Richard Branson

Throughout my years of bringing SaaS services online, I have encountered a remarkable diversity of state, federal, and international compliance standards that had to be embedded into the architecture from day one. GDPR, HIPAA, PCI-DSS, SOX — these are not afterthoughts to bolt on at the end of development. They are structural requirements that shape the platform.

In one engagement, the client was building an online sales platform for electronic cigarettes. At the time, e-cigarettes were an unregulated industry and rapidly becoming a popular alternative to traditional tobacco. The regulatory landscape was a minefield. Big tobacco wanted the competition suppressed. Government stakeholders saw diminishing tax revenues and were positioning to act. Citizen activists had genuine public health concerns.

One key design decision was whether to build in age verification — even though it wasn't yet a legal requirement. We believed it would be legislated. The client agreed, both on principle and on strategy. We looked at other industries where minors were already prohibited from purchasing online — tobacco, gambling, alcohol — and mined their compliance architectures for lessons.

The importance of regulatory compliance for software systems has never been greater. Regulations are increasingly shaping both functional and non-functional requirements for legacy and new systems alike. Mistakes carry real financial penalties and potential legal entanglements — at best, these slow growth; at worst, they end a business.

FIVE COMPLIANCE STANDARDS EVERY SAAS BUILDER SHOULD KNOW

◆ **GDPR** — General Data Protection Regulation. Governs data collection, processing, and privacy for EU residents regardless of where the company is based.

◆ **PCI-DSS** — Payment Card Industry Data Security Standard. Required for any platform processing credit card transactions.

◆ **HIPAA** — Health Insurance Portability and Accountability Act. Mandatory for any SaaS handling protected health information.

◆ **SOX** — Sarbanes-Oxley. Applies to financial reporting and internal controls for publicly traded companies and their vendors.

◆ **GLBA** — Gramm-Leach-Bliley Act. Governs how financial institutions handle private consumer information.

PART I — STAKEHOLDER INPUT

CHAPTER SIXTEEN

Service Features

""The measure of who we are is what we do with what we have.""

— Vince Lombardi

An analysis of the service's design and feature set is the final stage of the discovery process — and deliberately so. Understanding the actual software to be designed, redesigned, or transitioned can only begin after all other due diligence has been collected and analyzed. The UI, UX, features, and functionality must be evaluated against the backdrop of budget, stakeholders, resources, business model, compliance requirements, target market, and strategic goals. In that order.

The success or failure of a SaaS service is never the product of the technical brilliance of its design. SaaS functionality works when it aligns with the team supporting and selling it — and it fails, reliably, when the development team embarks on blind feature initiatives disconnected from real user need.

Less is more. Developer creep — the tendency for features to multiply beyond any strategic intention — is the bane of SaaS products. When we migrated MediaTracking to a new SaaS platform, we eliminated more than 50% of the landing pages and countless features that nobody knew existed or knew how to use. The majority of customer support content was eliminated because the remaining features were almost entirely intuitive.

Moving to SaaS is a strategic opportunity to realign the product with the value propositions your end users actually care about. Part of that alignment includes a plan for future features. The POS software we migrated had no integration with accounting software, yet over 90% of the install base used QuickBooks. The roadmap included a plan for building that API bridge — not as an afterthought, but as a defined development milestone.

FEATURE-DRIVEN DEVELOPMENT FRAMEWORK

- ◆ **Develop the overall model.** What is the complete conceptual picture of the service?
- ◆ **Create the feature list.** Prioritized, validated against user need, and tied to business goals.
- ◆ **Plan by feature.** Break the roadmap into feature-driven milestones, not just time-based sprints.
- ◆ **Design by feature.** Let each feature's requirements drive the design decisions for that component.
- ◆ **Build by feature. Develop, test, and ship one complete feature at a time** — not partial implementations across many.

PART II

Value Analysis

Chapters 17–26

With stakeholder input in hand, the work shifts to interrogating the value the service can deliver — and to whom. This section maps the commercial, legal, and competitive landscape before any architecture is committed.

PART II — VALUE ANALYSIS

CHAPTER SEVENTEEN

Key Accounts

""There are no traffic jams along the extra mile.""

— Roger Staubach

When companies are transitioning a legacy platform, adding a new SaaS layer, or retooling an existing service, identifying key accounts is a critical early step. Key accounts are those who constitute a large percentage of current revenue, those who carry meaningful influence in the market, or those willing to be early adopters of something new.

In one engagement, we needed to transition four hundred clients from an aging SaaS service to an entirely new platform. Our first attempt was simply to send test logins to key accounts with minimal documentation. The response was immediate and discouraging. Users reported dissatisfaction with the new UI, complained about lost functionality, and disliked changes to the UX. Worse, we received almost no actionable feedback — just a general sense of rejection.

We pivoted. In follow-up focus groups, we showed users three-to-five minute video demos highlighting specific areas where the new UI saved time and expanded functionality. Then we gave them the test accounts. The shift in reception was dramatic. Users came in with more confidence, engaged more constructively, and provided feedback that led directly to meaningful product improvements before the general rollout.

Key accounts also provide your most honest signal on pricing and packaging. They will tell you — sometimes bluntly — which features of the new service they would pay more for and which ones they wouldn't pay for at all. That intelligence is priceless, and it is almost never available through a survey.

ROADMAP CONSIDERATIONS

- **Prioritize transparency over early efficiency** — share the roadmap, even when it's incomplete.
- **Build a client community around the development process** — make them feel like co-architects.
- Test early and often, with structured feedback loops rather than open-ended requests.
- Put a checkpoint system in place so key account feedback is formally reviewed and actioned at defined intervals.

PART II — VALUE ANALYSIS

CHAPTER EIGHTEEN

Marketing

""The secret of getting ahead is getting started.""

— Mark Twain

Although the marketing tools you implement may cover most of your needs, there are almost always gaps. Your SaaS roadmap should plan for connecting third-party marketing tools, enabling them to communicate with each other, and filling those gaps before they become expensive mid-launch discoveries.

In 1999, our legislative tracking service entered the federal market with a significant brand problem: nobody in Washington had ever heard of us. What we did know was that once a prospect walked through a live demo, they converted at approximately one-in-three. The challenge was generating enough qualified demos.

We engaged a local marketing firm with a goal of 300 demos in six months. Their suggestion was to send each prospect their own login to the system. Experience told us this wouldn't work unless those accounts were loaded with a dashboard tailored specifically to that prospect's industry. So we adapted the SaaS platform to give the marketing team a simple interface for creating custom dashboards by vertical — and integrated it directly with the CRM. The result was a measurably shorter sales cycle and management metrics that hadn't existed before.

ROADMAP CONSIDERATIONS

- ◆ **Native integrations** — Built into the software or available in an app marketplace. Simplest to activate; limited to what the tool natively supports.
- ◆ **Third-party API integrations** — Some software provides its own API that allows direct connection to your platform.
- ◆ **Third-party connectors** — Tools like Zapier bridge apps that lack native or direct API integration.
- ◆ **Custom API integrations** — Built by your development team. Most expensive, most flexible, most tailored.

PART II — VALUE ANALYSIS

CHAPTER NINETEEN

Sales Funnel

""Focus on being productive instead of busy.""

— Tim Ferriss

"I want to know who the SOB is that just sent me anthrax," wrote the prospective subscriber to our SaaS service. This is not the optimal introduction to a business relationship one hopes will end with the prospect spending $5,000 or more annually.

The truth was I had authorized sending him a voodoo doll — a small, political-themed figurine with Republican and Democrat signage and several stick pins, accompanied by a vial of white powder around its neck. Our tagline read: "Our legislative tracking service is so cutting edge you'll think it's magic." The problem was that the week we began mailing these dolls was the same week a letter containing anthrax was discovered in the U.S. House of Representatives mail room.

The campaign ultimately won a prestigious national marketing award. It generated almost no subscribers. I felt like the financier of a film that won an Academy Award and bombed at the box office. Looking back, the campaign had no measurable call to action, no CRM integration, no coordinated follow-up sequence, and no stated value proposition.

A well-constructed sales funnel is the architecture that connects awareness to revenue. Every element of your SaaS platform should be evaluated against the question of how it supports or accelerates that funnel.

ROADMAP CONSIDERATIONS

- ◆ Could the platform offer a scaled-down free version that demonstrates core value before requiring payment?
- ◆ Could users pay based on usage rather than a fixed subscription, lowering the barrier to initial commitment?
- ◆ Can elements of the platform be linked to social channels or a mobile app to expand the top of the funnel?

PART II — VALUE ANALYSIS

CHAPTER TWENTY

Packaging & Pricing

""The single most important decision in evaluating a business is pricing power.""

— Warren Buffett

Packaging and pricing are among the most consequential decisions in a SaaS business — and among the most frequently made too early, with too little data.

At NewspaperClips.com, the legacy market had always priced based on the number of keywords monitored daily. Our SaaS platform eliminated the sorting labor entirely, but we retained the keyword-based pricing model — and used it as a competitive advantage. We could offer double the keywords at the same price as legacy competitors with zero increase in our own cost of goods sold.

In another service, we discovered that an archive feature had an extremely small technology and support footprint. It was quietly one of our most profitable features. Initially, it was included in all four pricing tiers. The result was a disproportionate concentration of users on the lowest tier. When we moved the archive feature to the third tier at $199 per month, more than 50% of the starter-tier users upgraded. The feature hadn't changed. The packaging had.

A good pricing strategy is not simply about setting the lowest competitive price. It is about aligning your price to the genuine value you deliver — and then structuring packages in a way that makes the most profitable tier feel like the most logical choice.

ROADMAP CONSIDERATIONS

- **Premium pricing** — Set above the market average; signals quality and exclusivity.
- **Cost-plus pricing** — Start from COGS and add a defined margin. Simple, but often leaves value on the table.
- **Package pricing** — Bundle features into defined tiers to simplify the buying decision.
- **Tiered pricing** — Different price points for different levels of access, usage, or user counts.
- **Value-based pricing** — Price to the value delivered to the customer, not to the cost of delivering it.

PART II — VALUE ANALYSIS

CHAPTER TWENTY-ONE

Legal & Contracts

""It's not the will to win, but the will to prepare to win that makes the difference.""

— Paul "Bear" Bryant

I spent significant time in the belly of the beast that some call law school. I escaped with a degree by discovering early on that the grade distribution created a natural equilibrium, and that the entire grade rested on two or three exams. The best outlines were produced by Law Review students, who were selective in their social calendar. I ingratiated myself through hosting house parties. The conversation was cerebral and only grew more so when alcohol entered the picture, but my guests bent over backwards to help me navigate the experience.

Law is a degree. You do not need it for most of the legal matters you will encounter in business. But I do encourage everyone to understand the basics of contract law, because we live and operate in a world built on agreements.

Strong fences make good neighbors. Good contracts make great partners. Handshake deals end poorly the moment one party believes they are no longer benefiting from the arrangement.

Evergreen clauses are essential in SaaS. Automatic renewal provisions will not prevent anyone from purchasing your service, but they will protect your recurring revenue from clients who simply forget to renew.

Mutual NDAs matter. Never enter an NDA where both parties aren't treated symmetrically.

Most companies spend too much on legal fees. There are many excellent online services that handle corporate formation, employee contracting, SaaS service agreements, and NDAs at a fraction of what a law firm charges.

Guaranteed minimums should be avoided. I have never seen a guaranteed minimum work in favor of the party doing the guaranteeing. Not once.

Keep your terms and conditions separate from your sales agreement. The sales agreement should reference terms and conditions as posted on your website.

When engaging a law firm, remember you're paying for everyone in the room. You usually don't need more than one person present.

PART II — VALUE ANALYSIS

CHAPTER TWENTY-TWO

Competition

""Keep your friends close; keep your enemies closer.""

— Sun Tzu

Burrelle's Press Clipping Bureau was founded in New York City in 1888. For its first century, the service required employees to cut clippings from local newspapers, make copies, and hand-deliver them to clients. In 1984, the fax machine transformed the industry. In 1994, delivery moved to email. By 2007, more than 50,000 publications were being tracked through an online platform. Today, the number of sources is effectively infinite.

What that history illustrates is the dramatic compression of the innovation cycle. What once took a century now happens in five years. Your competitors are moving. You cannot afford to watch from a distance.

That said, your competitors' innovation should never be the sole driver of your own development decisions. For years, our sales team lobbied for features our media monitoring service didn't offer. We chose not to build most of them. In the end, we didn't lose subscribers because of those decisions. Meanwhile, several competitors went out of business trying to maintain feature-laden offerings that diluted their value proposition.

When we entered the D.C. market, I made a point of meeting the heads of every company we might compete with. I developed strong relationships with most of them and turned several potential adversaries into referral sources. The ultimate outcome was a partnership or two — and eventually the sale of our company.

ROADMAP CONSIDERATIONS

- ◆ Get to know your competition's management. It is almost always worth the investment.
- ◆ Understand the single most compelling value proposition of each major competitor.
- ◆ Your competition should inform your thinking, but your own client-driven innovation should drive your growth.

PART II — VALUE ANALYSIS

CHAPTER TWENTY-THREE

IT Infrastructure

""Hardware eventually fails. Software eventually works.""

— Unknown

"This fan is not going to cut it," said my co-founder and CTO. We were in the kitchen of a shared business office in Austin, Texas, where our servers were residing. The landlord controlled the AC and set it to turn off an hour after 8 p.m. In an Austin summer, keeping that room below 80 degrees was becoming impossible. So we did the only thing we could afford: we cut the window sills to fit two window AC units. Problem solved until the condensation shorted out the power and tripped the breakers.

What I discovered in those early years was that nobody questioned our infrastructure. Through years of selling SaaS services, I never had a prospect ask to see our servers, examine our backup strategy, or review our redundancy plan.

Too many technology companies over-invest capital and internal resources in tech infrastructure because most are founded by technicians rather than by sales and marketing people. When evaluating the tech stack, apply the lens of the end user. The hardware must deliver the value proposition, but beyond that threshold, every dollar spent on infrastructure gold-plating is a dollar not spent on growth.

ROADMAP CONSIDERATIONS

- **Ensure the technology aligns with corporate goals and budget** — not the other way around.
- Identify specific needs that must be addressed in moving to a cloud-native environment.
- Prioritize infrastructure investments by business impact, not technical elegance.

PART II — VALUE ANALYSIS

CHAPTER TWENTY-FOUR

DevOps

""It's not the Xs and Os. It's the Jimmys and Joes.""

— Unknown Football Coach

In coaching, there's an old saying: "The last one holding the chalk wins." When diagramming plays on a whiteboard, whoever has the marker last can always redraw the defense or offense to counter the previous coach's scheme. But ultimately, it's not the Xs and Os that make a play work. It's the Jimmys and Joes. The same is true in software development.

We once needed to transfer a legacy platform from FoxPro to a modern architecture. The principal architect of the legacy system was unavailable. We found a freelancer with FoxPro experience who could help us document the existing code and serve as a bridge. In another instance, the legacy system's gatekeeper attempted to charge an exorbitant fee for the migration. We hired a different freelancer, rewrote the interface code, and achieved the same outcome at one-tenth the cost.

DevOps needs a team that can deliver a stable and reliable service — one with tight collaboration between development and IT, a strong QA function, and processes that minimize the lag between a bug being identified and a fix reaching the customer.

ROADMAP CONSIDERATIONS

- Who are the key individuals required to achieve the development goals?
- **List and prioritize the major technical obstacles** — don't allow them to be discovered mid-sprint.
- Identify where outside resources can fill gaps in the current team, domestically and internationally.

PART II — VALUE ANALYSIS

CHAPTER TWENTY-FIVE

Measurements (KPIs)

""If you can't measure it, you can't improve it.""

— Peter Drucker

In 2002, I was part of a SaaS service that created one of the first tools to mine data from online news sources. What we had not accounted for was the rate at which online publications modified their structure, breaking the scripts we had written to harvest their content. As the source list grew from hundreds to thousands, the publications being missed exploded. We had failed to build any mechanism to automatically detect when a source went offline.

The result was that our own subscribers were alerting us to coverage gaps before we discovered them ourselves. The solution was a monitoring and alerting system that allowed one person to track over 20,000 sources in real time. What had been a reactive embarrassment became a proactive operational advantage.

KPIs must be defined for every meaningful dimension of the business — not just the obvious ones. In one SaaS service, sales team members had been creating "free trials" and, in a remarkable percentage of cases, simply never ending them. These were perpetual free trials — a polite term for non-paying subscribers. One salesperson actually requested commission on non-paying accounts. These things do not fix themselves.

ROADMAP CONSIDERATIONS

- **Identify mission-critical service components** — the ones whose failure affects the customer most severely.
- Ensure real-time monitoring and measurement of those components.
- Automate as much of the monitoring and alerting as possible. Manual checks will eventually fail.

PART II — VALUE ANALYSIS

CHAPTER TWENTY-SIX

Goals

""Setting the goal is not the main thing. It is deciding how you will go about achieving it and staying with that plan.""

— Tom Landry

If we want to change our business circumstances, we must follow a sequence that most organizations get backwards. First, define the goals. Second, define who you must become to achieve them. Third, identify what you must do to get there. Most businesses believe they must first obtain the resources, then do the necessary work, and eventually become who they want to be. The order matters enormously.

At GalleryWatch, we needed to become the bipartisan, trusted partner of legislative and committee staff. To become that, we hired non-political staff, marketed equally to organizations across the full political spectrum, and took deliberate care never to offer political commentary of any kind. These were not PR decisions — they were identity decisions.

The result was that within a year, we were invited into the Speaker of the House's office to discuss the speed at which public record information should be made available to the public. That conversation was proof of concept for everything we had committed to being.

ROADMAP CONSIDERATIONS

- ◆ **Who do you need to become?** Define the identity and character of the organization that can achieve these goals.
- ◆ **What disciplines must you execute on?** These are the specific, repeatable behaviors that bring the goals closer over time.
- ◆ **What is it you want to achieve? State this with measurable specificity** — not as a direction, but as a destination.

PART III

Infrastructure Alignment

Chapters 27–42

The architecture of a SaaS service is the result of dozens of decisions made before the first sprint. This section ensures those decisions are made deliberately — not by default.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER TWENTY-SEVEN

DevOps

""DevOps is not a goal, but a never-ending process of continual improvement.""

— Jez Humble

In October 1996, we had thirty of the most influential Texas politicos in a hotel ballroom, ready to witness a live demonstration of our new legislative SaaS service. We had been working twenty-hour days for nearly a year to reach this moment.

The night before, my development partner delivered the news: the login system was down. The reason was extraordinary. Our lead developer had gotten into a dispute with another developer over a code architecture decision. Instead of resolving the conflict or escalating it, he had unilaterally deleted all 30,000 lines of code and started rewriting from scratch.

Some team members wanted to fire him immediately. But he was the only person who knew the system well enough to rebuild it in time, and he was already rewriting. There was no test server — he was coding directly into the production environment on no sleep. We had no version control, no staging environment, and no meaningful debugging tools.

The system came back online thirty minutes before the meeting. The demo was a success. But the entire crisis was the direct result of failed DevOps practices: no version control, no formal code review process, no communication protocols between developers, and no separation between production and testing environments. We had survived through luck and sheer talent, which are not a sustainable infrastructure strategy.

ROADMAP CONSIDERATIONS

- ◆ Establish and enforce formal development processes: version control, staging environments, code review, and deployment pipelines.
- ◆ Define IT infrastructure and security standards before the first line of code is written.
- ◆ **Determine hosting parameters** — cloud provider, architecture, and failover strategy — as part of the development plan, not after it.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER TWENTY-EIGHT

UI / UX

""When the UX doesn't consider all users, shouldn't it be called 'Some User Experience' — or SUX?""

— Billy Gregory

There are many considerations in designing the optimal user interface and user experience, but the paramount one is simple: it must meet the needs of the end user — not the needs the product manager imagines the end user has.

When working with a point-of-sale software company, the legacy UI was firmly rooted in the 1990s — because it had been designed in the 1990s. But the install base was over 400 independently operated stores across the country, and the users had spent years developing keyboard-based workflows around this interface. For many, the ability to complete a transaction without ever touching a mouse was not a quirk — it was a professional skill.

Adding mouse-driven navigation would have slowed transaction time and frustrated a user base that had never asked for it. The printing output had to remain compatible with legacy label printers, cash drawers, and handheld scanners.

After weighing all of these considerations, the decision was made to preserve the legacy UI and UX. Protecting the existing revenue base was more important than attracting new revenue. This is not always the right call, but it was the right call for this client. What is always the right call is making the decision deliberately, with all of these tensions named and evaluated — not by default.

ROADMAP CONSIDERATIONS

- **Optimize for speed and efficiency above all else** — every unnecessary interaction is a tax on the user.
- Design for scalability; the UI must accommodate new features without requiring a full redesign.
- **Treat mobile as primary. The mobile web is not a secondary consideration** — it is often the primary one.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER TWENTY-NINE

Hosting

★ NEW CONTENT

""Uptime is everything. Clients don't want to hear a server is down; they just expect things to work.""

— John Burks

Not long after we moved our legislative tracking service to a dedicated server environment, we experienced what I can only describe as the quietest catastrophe in the history of our company. No alerts. No alarms. No frantic phone calls. The service had simply gone offline on a Sunday morning during legislative session — the single worst possible moment — and none of our monitoring tools caught it. The first alert came from a subscriber.

Our hosting decision had been made primarily on price. We had selected a shared hosting environment because it was affordable and someone on the team had a good experience with it years earlier. What we had not done was match our hosting infrastructure to the demands of the service we were actually running — or the expectations of the market we were serving.

Hosting is not a commodity decision. It is a foundational architectural decision that determines the ceiling of your service's reliability, performance, scalability, and security. Your hosting strategy should be derived from your SaaS service's requirements — not from what happens to be on sale.

Understanding Your Hosting Options

Shared hosting is appropriate for development environments and very early-stage prototypes. It is almost never appropriate for a production SaaS service serving paying customers.

Virtual Private Servers (VPS) offer dedicated resources within a shared physical infrastructure. They are a viable option for early-stage SaaS products with modest, predictable traffic — but they require manual scaling.

Cloud hosting — through providers like AWS, GCP, and Azure — is the current standard for production SaaS. It offers elastic scaling, global regional deployment, managed services, and a pay-for-what-you-use billing model that aligns infrastructure costs with revenue growth.

Managed cloud platforms such as Heroku, Render, and Railway offer a layer of abstraction above raw cloud infrastructure. They are faster to deploy and easier to manage, at the cost of some configurability and, at scale, some cost efficiency.

> *The question to ask is not 'what hosting can we afford right now?' The question is 'what hosting will this service require at 3×, 10×, and 50× current load — and how do we get there without a crisis?'*

Your customers don't think in terms of "99.9% uptime." They think in terms of whether the service worked when they needed it. A three-nines SLA permits 8.7 hours of downtime per year. For a service used during business hours in a mission-critical workflow, that can be catastrophic. Build your hosting architecture to the standard your customers actually require, not to the standard that saves the most money.

ROADMAP CONSIDERATIONS

- ◆ **Match the hosting environment to your actual production workload, compliance requirements, and uptime SLA** — not to what is cheapest at launch.
- ◆ **Understand the elastic scaling capabilities of your chosen hosting model before you commit** — you will need them.
- ◆ Ensure your hosting architecture accounts for all applicable compliance requirements: HIPAA, GDPR, PCI-DSS, SOX.
- ◆ Define your uptime SLA target based on what your customers actually need, then build backward from that requirement.
- ◆ **Plan for geographic distribution if you are serving users in multiple regions** — latency and data residency both matter.
- ◆ Build monitoring and alerting into the hosting layer from day one. The first alert about a production outage should never come from a subscriber.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER THIRTY

Scalability

""Scalability is nothing without sustainability.""

— Kaushal Soni

When we launched our initial online newspaper clipping service, we were serving subscribers in Texas and Florida, monitoring a few hundred online publications, and serving a user base under a hundred subscribers. A year later, the top 100 U.S. publications were in our network, and our subscriber base had grown to just over 200. The growth had been controlled, steady, and entirely manageable from an infrastructure standpoint.

Then, in the fourth year, we launched a new service feature. It was popular. It required significantly more time on the portal and generated substantially more database requests per session. We had not factored in the infrastructure cost implications of those new usage patterns. The site began to slow at peak times and, for some users, became unusable.

Looking back, the problem was entirely foreseeable. We had growth projections. We had usage data. We had sales team forecasts. What we had not done was translate any of that information into infrastructure planning. Today, load testing tools are affordable, accessible, and effective. Regardless of which tools you use, the failure to model growth scenarios before they become live crises is an organizational failure, not a technical one.

ROADMAP CONSIDERATIONS

- ◆ **Model your infrastructure requirements at 3×, 10×, and 50× current load before launch** — not after you start hitting limits.
- ◆ Evaluate load balancing options appropriate to your architecture and traffic profile.
- ◆ Conduct load and performance testing as a standard part of every major feature release, not just at launch.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER THIRTY-ONE

National v. International

""Absorb what is useful, discard what is not, add what is uniquely your own.""

— Bruce Lee

In 2018, the European Union's General Data Protection Regulation came into force and fundamentally changed the calculus for any SaaS service with a connection to EU residents or their data. I was consulting with a U.S.-based SaaS company that had proxy servers in Europe. Those servers suddenly carried potential legal liability they had never anticipated.

The European subscriber base constituted less than 1% of gross revenues. As a small company, the cost and operational complexity of navigating an untested and aggressive new regulation wasn't justifiable against that revenue. The servers were decommissioned. It was not an ideal outcome — but it was the right business decision given the circumstances.

Whether you are building for a single domestic market or planning for international expansion, the decision must be made explicitly in the roadmap. Several of the earlier analyses — target market, compliance, IT infrastructure — are directly affected by whether the scope is national or international.

ROADMAP CONSIDERATIONS

- **Identify all relevant international data protection and privacy regulations before designing the infrastructure** — GDPR, CCPA, and equivalents.
- Design the infrastructure and architecture to handle a globally dispersed user base, even if you're not serving one yet.
- Adapt UI/UX to align with the expectations and usability patterns of the target audience in each market, not just translate the existing interface.
- Offer payment options that accommodate regional preferences and currency requirements.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER THIRTY-TWO

Third Party APIs

""When building a team, I first search for people who love to win. If I can't find those, I search for people who hate to lose.""

— Ross Perot

As a SaaS service scales, it inevitably accumulates client requests for capabilities that live outside the platform's core. The service then faces three choices: do nothing, build the capability in-house, or acquire it by integrating a third-party solution via API.

The Media Tracking service had been designed before social media existed as a content category. For years, our subscribers had told us they wouldn't pay for social media monitoring. Then came the tipping point — and seemingly overnight, those same subscribers were demanding exactly what they'd said they didn't want.

We partnered with vendors who sold us a data feed. The first vendor was acquired; we had 90 days to find a replacement. The second vendor provided a raw data firehose — more control, significantly higher infrastructure costs. That vendor was then acquired by a direct competitor and we were shut out again. Our final solution was a fully hosted, white-labeled platform from a third party, where we handed off users via URL. Most subscribers eventually abandoned it.

That journey — from harvesting and owning our social content to outsourcing every element of it — was never profitable at any stage and consistently distracted talent from more valuable initiatives. The lesson isn't that third-party APIs are wrong. The lesson is that API dependencies create risks that must be modeled explicitly.

ROADMAP CONSIDERATIONS

◆ **Evaluate whether the API's functionality fully meets your integration requirements** — and what happens if it partially doesn't.

◆ Review the quality of API documentation. Poor documentation is a reliable predictor of poor support.

◆ Confirm protocol compatibility with your current technology stack.

◆ Analyze the pricing structure, licensing terms, and what would happen to those terms in a vendor acquisition scenario.

◆ Investigate the vendor's versioning and compatibility policies. How much notice will you receive before a breaking change?

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER THIRTY-THREE

Software Integrations

""I'm not for integration and I'm not against it.""

— Richard Pryor

In redesigning a SaaS point-of-sale solution, users had made clear that the reporting functionality in the legacy platform was inadequate across every dimension. Reports didn't reliably export to spreadsheets. When they did, there was no control over the output format. Reports couldn't be amended within the platform.

We evaluated several third-party reporting applications and selected a vendor whose product was well-received. The integration worked well until the vendor released a major upgrade that required a license per-subscriber to the SaaS platform. What had been a $400 annual line item became over $145,000 per year.

The solution was to adapt an open-source reporting library to meet subscriber needs. The outcome was comparable and, in some respects, more maintainable. But the lesson was clear: any third-party integration that sits in the critical path of your product's value proposition needs a viable exit strategy. The best exit strategy is wholly owned and licensed code.

ROADMAP CONSIDERATIONS

- **Assess the compatibility of systems to be integrated** — languages, databases, and dependency chains.
- Model data mapping, transformation, and validation requirements before beginning integration work.
- Evaluate API availability and suitability for your architecture.
- For every critical integration, define the fallback plan. What do you do if the vendor changes pricing, gets acquired, or discontinues the product?

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER THIRTY-FOUR

Development Timeline

""Failure is not fatal, but failure to change might be.""

— John Wooden

I was serving as liaison between an outsourced development team and a client who wanted to retool their SaaS platform. The project was structured into twelve sprints of two to three weeks each. By the end of sprint three, we had produced meaningful work — but received almost no client input. I could feel the development team's frustration rising.

I asked the owner to sit down with me and walk through the application. Within fifteen minutes, he was visibly frustrated. The new UI didn't reflect what he had envisioned. Several features were missing. He wanted to know why work had started without what he called an "all-encompassing design specification."

I left that meeting feeling better about our timeline, not worse — because the owner's reaction confirmed exactly what I expected. There is no such thing as a complete design spec. Clients never provide sufficient detail upfront, and they never adequately test what they're given until it's in front of them. What I was practicing — and what the timeline needed to accommodate — was value-driven development: responding to change rather than following a static plan.

ROADMAP CONSIDERATIONS

- ◆ Is testing completed within the same development sprint as the corresponding code? If not, defects accumulate faster than they're resolved.
- ◆ Is documentation minimal, purposeful, and secondary to working code? Documentation that no one reads consumes time that could build features.
- ◆ Does the plan allow the software to evolve in response to changes in business conditions and user feedback? If the roadmap is immutable, it's already wrong.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER THIRTY-FIVE

Software Features

""It's not a bug. It's a feature.""

— Unknown

It was a standard weekly development team meeting at GalleryWatch.com when one of our developers told me he'd been working on something in his off time. He had written a script to harvest public arrest records and cross-reference them against our database of legislators and their staff. He explained this could be used to "influence" members of the legislature. I suggested we call it "The Godfather," recommended he bring it up in the team meeting, and watched as it became one of our more spirited discussions.

Feature creep is the tendency for product requirements to expand beyond what was originally scoped — driven by good intentions, creative minds, and the irresistible logic that "someone would use this." The product phenomenon typically results from designing for what sounds interesting rather than for what users demonstrably need.

Additional features, bundled with price discounting, are not the path to differentiation. The companies that win in SaaS are not the ones with the most features — they are the ones with the most coherent, reliable, and satisfying customer experience around a well-defined set of capabilities.

ROADMAP CONSIDERATIONS

- ◆ Does this feature directly align with the documented business goals?
- ◆ **Will it serve the needs of the broad user base** — or only a narrow subset?
- ◆ Have you done user research to understand which existing features could be removed before adding new ones?

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER THIRTY-SIX

Quality Assurance

""I couldn't repair your brakes, so I made your horn louder.""

— Tater, Disney's Cars

Microsoft taught me an important lesson in the 1990s. They would release development tools and server operating systems for free. New versions shipped roughly quarterly, mostly containing bug fixes rather than new features. What Microsoft demonstrated was that software doesn't need to be perfect to win. It needs to iterate faster than the competition and patch problems before users lose patience.

In early 2000, we launched a feature to our legislative tracking service that sent text-message updates to mobile phones — flip phones and BlackBerries with severe character limits. We invented a vocabulary of abbreviations for legislative actions. Users provided feedback within the first few weeks, and many of our initial abbreviations were changed based on their suggestions. The users were the quality assurance team, and they were remarkably good at it.

The best QA is your user base. Failing to put something in front of them early — out of fear it isn't ready — will cost you more time in the long run than the feedback would have.

ROADMAP CONSIDERATIONS

- ◆ Is your service in a regulated industry? If so, your QA process may need to be more formal and less agile than you'd prefer.
- ◆ Have you determined whether load, stress, and penetration testing are appropriate for your platform and your risk profile?
- ◆ **Do you have meaningful incentives** — for users and internal staff — to actively test and report bugs?

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER THIRTY-SEVEN

Bug Tracking

""Success consists of going from failure to failure without loss of enthusiasm.""

— Winston Churchill

Bug tracking plays a foundational role in agile SaaS development. In iterative sprint-based environments, where code is being written, reviewed, tested, and deployed in compressed cycles, an underpowered bug tracking system is not merely an inconvenience — it is a structural liability.

In one consulting engagement, I was brought in to help accelerate software development and improve customer satisfaction. The organization's issue management process consisted entirely of email chains. The results were predictable: dissatisfied customers, issues falling through the cracks, and development cycles consumed by rediscovering problems that had already been reported.

After implementing a formal bug tracking system, we identified that more than sixty percent of customer satisfaction issues were tied to the integration with a single third-party software vendor. Furthermore, those issues were consuming more than eighty percent of development time. That insight was completely invisible before the system existed. It changed the entire strategic conversation — and ultimately led to replacing the integration altogether.

ROADMAP CONSIDERATIONS

- ◆ **Bug tracking empowers teams to prioritize and address defects efficiently** — particularly in fast-moving sprint cycles.
- ◆ **A centralized system ensures that bugs are documented, owned, and tracked to resolution** — not lost in email threads.
- ◆ Over time, bug data reveals patterns that drive strategic decisions about architecture, vendor relationships, and product priorities.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER THIRTY-EIGHT

Redundancy & Disaster Recovery

""Our disaster recovery plan goes something like this: Help!… Help!""

— Dilbert (Scott Adams)

In the early 2000s, a virus found its way onto the network of a SaaS company I was working with. In an estimated ninety minutes, the virus had erased or corrupted nearly 40% of our source code and over 30% of our archived data.

Restoring the archived data from backups was manageable. The source code was a different problem. The virus had compromised both our production server and our test server — which had been serving as the production backup. Both machines were on the same network.

What saved us was that developers around the world had various segments of the source code saved on their local machines from prior work sessions. We spent over a week piecing the codebase back together from those fragments. The service was partially restored after a week. Full functionality returned in another two weeks. Complete restoration of data mining capabilities took over two months.

The lessons were direct: more backups, stored in more locations, with no two storage systems on the same network or under the same logical control. And any backup plan that has never been tested is, by definition, untested.

ROADMAP CONSIDERATIONS

- ◆ Define a disaster recovery strategy that explicitly addresses both recovery time objective (RTO) and recovery point objective (RPO).
- ◆ **Assess whether your business model can withstand the slower recovery of a cold backup** — or whether you need hot standby infrastructure.
- ◆ Quantify your cost of downtime. That number should drive your redundancy investment, not the other way around.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER THIRTY-NINE

Target Markets

""Don't find customers for your products. Find products for your customers.""

— Seth Godin

I worked with a law firm that also operated a real estate and property management company serving high-net-worth clients. They were not the only firm in this niche, but they had something the others didn't: an internal SaaS platform that made client communication, document management, and multi-stakeholder coordination dramatically more efficient.

Working with NewspaperClips.com, we built a feature that allowed trade associations to share curated news clips with their membership via email. In 2002, this was genuinely cutting-edge. The key was identifying the right market vertical before building the feature, not after. We then recognized that state agencies had similar internal distribution needs. Those two verticals became our focus, and we deliberately declined to pursue PR firms and advertising agencies.

Our subscribers experienced us as a service that uniquely understood their needs, which was only true because we had committed to understanding them specifically.

ROADMAP CONSIDERATIONS

- ◆ Commit to a narrow SaaS target market before launch. Broadening it later is always possible; discovering you had no market at all is not recoverable.
- ◆ Find the specific need within the target market where your service is not just good, but irreplaceable.
- ◆ Map the demographic, geographic, and psychographic characteristics of your target market in detail.
- ◆ Assess the accessibility and loyalty potential of the target market. Some markets are easy to reach but churn fast; others are hard to reach but stay forever.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER FORTY

Marketing Automation

""Automation is solving the problem once and then putting it on autopilot.""

— Michael Hyatt

I worked with a non-profit SaaS service that licensed its platform to users in 22 countries at a starting price of $15,000 for the first year. What made the service remarkable was not the product — it was the sales process. From the moment a prospect expressed interest, they entered a fully automated funnel. Not a single phone call was made until a purchase was imminent.

Did it work? Each year, more than 2,000 prospects entered the funnel. Over 120 ultimately purchased the service. The entire operation was run by one part-time employee. The economics of that model — driven entirely by automation — are difficult to replicate without it.

Whatever marketing method you adopt for your SaaS service, automation is not optional — it is the lever that makes your efforts scalable. Upgrades, free trial management, limited account expirations, renewal sequences, upsell campaigns — all of these should be supported by the platform itself, not managed manually.

ROADMAP CONSIDERATIONS

- ◆ Define clear marketing goals and segment your target audience before building any automation workflows.
- ◆ Select a CRM with a robust, well-documented API that supports the automation layers you plan to build.
- ◆ Choose automation tools that integrate natively with your CRM and your SaaS platform.
- ◆ Test automated campaigns before launching them at scale. A broken automation can be more damaging than no automation.
- ◆ Monitor campaign performance continuously and optimize based on actual conversion data.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER FORTY-ONE

Pricing Format

""Nowadays people know the price of everything and the value of nothing.""

— Oscar Wilde

During a merger between two SaaS companies, the pricing question became a significant point of friction. The acquiring company's product was priced at more than double the acquired company's equivalent offering. Even though the two services operated in different countries, price parity would eventually be required — and the gap was too wide to bridge by simply splitting the difference.

In a separate scenario, a SaaS service was replacing a legacy on-premise software whose one-time purchase price was approximately eighteen times what the SaaS competitors were charging monthly. Getting existing customers to shift from "I already paid for this" to "I subscribe to this" required a carefully staged pricing strategy with compelling incentives for early adopters.

For the merger, we modeled projected client retention at different price points and used the resulting churn analysis to establish a staged price increase over two years. For the on-premise-to-SaaS migration, early adopters received the deepest discounts — locked in for three years — with the discount declining sharply for those who waited.

ROADMAP CONSIDERATIONS

- ◆ **Understand the psychological factors influencing how customers perceive price changes** — particularly in transition scenarios.
- ◆ Communicate the pricing strategy to customers with full transparency about the value they receive at each tier.
- ◆ Evaluate all relevant pricing models: one-time, subscription, tiered, and usage-based. Each carries different cash flow implications.

PART III — INFRASTRUCTURE ALIGNMENT

CHAPTER FORTY-TWO

Cash Flow

""Never take your eyes off the cash flow because it's the lifeblood of business.""

— Sir Richard Branson

One of our client's primary goals was to fund their SaaS transition using revenue from the existing install base rather than external capital. Conversations with the customer support team revealed that most clients were not properly backing up their data in the legacy system. Cloud backup was a natural solution. A partnership with a backup vendor allowed us to transition a significant number of clients to a better backup practice while simultaneously giving the company intelligence about data volumes that would eventually need to be hosted in the SaaS environment.

Separately, clients had been asking for digital pricing display capability — a way to update kiosk-format pricing in real time. This feature could be broken out and sold independently, well before the full SaaS platform launched. And the company's support offering had been priced as email-only when it was actually delivering white-glove service. Repricing that support tier added six-figure annual revenue.

Cash flow modeling must account for the full economic picture: MRR, upsell opportunities, churn, contraction, cost of goods sold, and the timing of infrastructure expenditures relative to revenue recognition. The model is not a budget. It is a decision-making instrument.

ROADMAP CONSIDERATIONS

- ◆ Calculate net operating cash flow from core business operations.
- ◆ Account for capital expenditures tied to infrastructure, development, and growth investments.
- ◆ Project future cash flows from historical data, incorporating seasonality and growth assumptions.
- ◆ Calculate the operating cash flow-to-revenue ratio and benchmark it against industry standards.
- ◆ **Account for non-cash items** — depreciation, amortization, and deferred revenue — that affect reported earnings but not actual cash position.

PART IV

Fractional Decisions

Chapters 43–45

Not everything has to be built in-house or staffed locally. This section examines where outsourcing your talent — in sales, support, and development — accelerates the roadmap rather than compromising it.

PART IV — FRACTIONAL DECISIONS

CHAPTER FORTY-THREE

Sales Contracting

""Master your strengths. Outsource your weaknesses.""

— Ryan Khan

It continues to baffle me that so many businesses never seriously consider outsourcing their sales function, in whole or in part. The consistent objection is that you need to be physically present with your sales team to ensure quality. In practice, most businesses struggle to find quality people at all — and the founder or owner ends up being the de facto primary salesperson indefinitely.

In the early 2000s, I assisted a company whose President was convinced that only he could bring on new subscribers. I entered into an arrangement at a low base with high commission, then turned around and outsourced list generation to a team in India and lead generation to a team in the Philippines. I handled only warm and hot leads for closing. The result was a 400% increase in annual sales.

The question is never "can we afford to outsource sales?" The question is "can we afford not to?"

ROADMAP CONSIDERATIONS

- ◆ Ensure the outsourced sales team has a thorough understanding of the product, target market, unique value proposition, and sales process before they make a single contact.
- ◆ **If operating across cultural or language boundaries, invest in cultural alignment** — not just translation.
- ◆ Maintain clear oversight structures, reporting mechanisms, and performance metrics from day one.
- ◆ Review the outsourcing agreement carefully, particularly around IP, data handling, and exclusivity provisions.

PART IV — FRACTIONAL DECISIONS

CHAPTER FORTY-FOUR

Support

""Do what you do best. Outsource the rest.""

— Peter Drucker

One of our consulting clients was supporting an install base of over 200 customers with one full-time tech support employee and a part-time office manager. The predictable result was a deeply frustrated install base and an employee on the edge of burnout.

The root cause was an inability to find qualified support personnel locally. The owner had a geography-bound hiring assumption that was limiting his talent pool to a small region. Within a month, the company had hired a full-time customer support manager located in Pakistan and two technical support employees in the Philippines and Jamaica. The quality of support improved, response times dropped, and the full-time employee finally had bandwidth to tackle higher-value work.

Contracting platforms like Fiverr, Upwork, and Toptal make this kind of distributed team both practical and accountable — they handle payment, time tracking, and contractor management within their platforms.

ROADMAP CONSIDERATIONS

- ◆ Compare the total cost of an outsourced support model against in-house hiring, including benefits, training, turnover, and management overhead.
- ◆ Assess the outsourcing provider's experience with support for businesses similar to yours in complexity and market.
- ◆ Ensure the team can support in the language or languages your customers expect.
- ◆ Define SLAs with the provider: response time, first-contact resolution rate, escalation protocols, and availability windows.
- ◆ Establish clear communication channels and regular reporting cadences.

PART IV — FRACTIONAL DECISIONS

CHAPTER FORTY-FIVE

Development Team

""ChatGPT is scary good.""

— Elon Musk

I have spent the past three decades leveraging the skills of developers in India, Romania, Pakistan, and other countries throughout the world. It has been a genuine equalizer — allowing our teams to compete with multinational software platforms on a fraction of the budget. But the next generation of development won't be offshored. It will be outsourced to artificial intelligence.

The development skills that will carry the most value going forward are not deep database architecture expertise or advanced programming proficiency alone. The highest-value skill will be the ability to work effectively alongside AI — to prompt it well, to evaluate its output critically, and to edit the code it produces into something production-grade.

Our teams have already begun using AI for load testing, quality assurance, and third-party integration work. The time savings are significant and the quality, particularly for well-scoped tasks, is remarkably good. I foresee a future where the development team is smaller, primarily responsible for code maintenance and architectural oversight, while AI handles the bulk of new feature development.

ROADMAP CONSIDERATIONS

- **AI-powered automation can meaningfully streamline development tasks** — but it still requires human oversight, review, and judgment.
- AI analytics applied to application logs, user interactions, and system performance can surface insights that were previously invisible.
- AI can automate significant portions of repetitive coding work, compressing time-to-market on well-defined features.
- **AI security techniques** — behavioral analysis, anomaly detection — can detect and respond to threats more quickly than manual monitoring.

PART V

Measured Implementation

Chapters 46–52 ★ All New Content

The work of the first four phases culminates here. Measured Implementation is where the roadmap becomes reality — where plans are tested against actual conditions, adjusted against real data, and driven by a team with the clarity and alignment to see it through.

PART V — MEASURED IMPLEMENTATION

CHAPTER FORTY-SIX

Management Team

★ NEW CONTENT

""Talent wins games, but teamwork and intelligence win championships.""

— Michael Jordan

We have now worked through four complete phases of the Beyond Bedlam process: listening deeply to stakeholders, analyzing the genuine value the service can deliver, aligning the infrastructure with the vision, and making smart fractional decisions about where to build capacity and where to borrow it. All of that analysis culminates here, in Measured Implementation — and the first question to answer before anything else is, who is going to lead this?

I learned a hard lesson early in my SaaS career about the cost of misaligned leadership. We had an extraordinary product and a development team that could build anything. What we lacked was a management structure that could translate business objectives into development priorities — and a leadership team honest enough to say what they didn't know. The result was a series of technically impressive releases that missed the market entirely, because nobody in a leadership role had both the authority and the accountability to connect the product vision to the customer reality.

Assembling the right management team for a SaaS implementation is not simply a staffing exercise. It is a strategic act. The people in key roles will determine the pace of execution, the quality of decisions, and the resilience of the organization when — not if — things go sideways.

The Essential Roles

Every SaaS implementation, regardless of scale, requires clear ownership across four domains: product vision, technical execution, revenue generation, and operational sustainability. These don't always map to four separate people, particularly in early-stage organizations, but they must be explicitly owned by someone.

The **product owner** is responsible for translating business strategy into a product roadmap and ensuring the development team is building the right things in the right order. This person must be deeply conversant in both the market's needs and the platform's capabilities.

The **technical lead** owns the architecture, the development processes, the quality standards, and the platform's reliability. The best technical leads are not the ones who say yes to everything. They are the ones who know which tradeoffs are acceptable and which ones are catastrophic.

The **revenue lead** — whether a VP of Sales, a Head of Growth, or a founder who owns the go-to-market — is responsible for connecting the product to the people who will pay for it. In SaaS, this role is inseparable from the product roadmap.

The **operational lead** keeps the organization running — managing the financial model, the team, the vendor relationships, and the processes that prevent the business from becoming its own obstacle.

"The best management teams are not the ones that avoid conflict. They are the ones that resolve it quickly, without losing the institutional memory of what was decided and why."

Alignment Over Talent

I have watched talented management teams destroy companies through misalignment, and I have watched average teams build remarkable ones through genuine shared commitment to the same goals. Talent is necessary but not sufficient. Alignment — on mission, on priorities, on how decisions get made — is the multiplier.

In practice, this means that before the first sprint is planned or the first hire is made, the management team needs to have an honest conversation about decision rights. Who has final authority on the product roadmap? Who can commit the company to a vendor relationship? Who can authorize a scope change? These conversations are uncomfortable, but their absence is far more costly than the discomfort of having them.

ROADMAP CONSIDERATIONS

◆ **Explicitly map the four core ownership domains** — product vision, technical execution, revenue generation, and operations — to specific individuals before implementation begins.

◆ Identify gaps between required capabilities and current team composition. These gaps are risks, and they need mitigation plans.

◆ Define decision rights in writing. Who can commit on what, and up to what threshold?

◆ **Establish a regular management cadence** — weekly operational, monthly strategic — and protect those meetings from the tyranny of the urgent.

◆ **Define what success looks like at 30, 90, and 180 days** — in measurable terms that every member of the management team can recite from memory.

PART V — MEASURED IMPLEMENTATION

CHAPTER FORTY-SEVEN

Cash Flow Analysis

★ NEW CONTENT

""Revenue is vanity, profit is sanity, cash flow is reality.""

— Unknown

Every decision in the Beyond Bedlam process carries a price tag. The hosting architecture decision costs something. The development timeline has a payroll attached to it. The marketing automation strategy requires tooling. Before the first line of code is committed to a production environment, you need to know exactly what this initiative costs, when it will generate revenue, and how long the organization can sustain the gap between the two.

I have seen more SaaS initiatives fail from cash starvation than from technical failure or product-market mismatch. The platform was good. The market was real. The team was capable. But nobody had modeled the cash flow with the discipline and honesty the exercise requires — which meant nobody knew how close they were to the edge until they were already over it.

Cash flow analysis in a SaaS context is not the same as a P&L; projection. Profitability and cash flow can diverge significantly, particularly in a subscription business where revenue recognition and cash receipt are often separated in time. Understanding this distinction is not an accounting technicality — it is a survival skill.

The SaaS Cash Flow Model

Monthly Recurring Revenue (MRR) is the heartbeat of the model. But MRR at any given moment is the net result of new bookings, expansion revenue from upgrades and upsells, contraction revenue from downgrades, and lost revenue from churned subscribers.

The **Customer Acquisition Cost (CAC)** is the fully-loaded cost of bringing one new subscriber aboard: marketing spend, sales commissions, onboarding labor, and any trial-period revenue concession. If you don't know your CAC, you cannot know whether your business model is viable.

The **Customer Lifetime Value (LTV)** is the total revenue a subscriber will generate, discounted to present value, net of the ongoing cost to serve them. The ratio of LTV to CAC is one of the most reliable indicators of business model health in a SaaS company. A ratio below 3:1 means the business is spending too much to acquire customers relative to what they return.

The **Churn rate** is arguably the most important single number in a SaaS cash flow model. A business with 5% monthly churn has a very different economic reality than one with 1% monthly churn, even if every other metric is identical.

The **cash runway** is the number of months the organization can sustain operations at its current burn rate before it runs out of capital. Knowing this number — precisely, not approximately — is not optional.

Build three scenarios: conservative, base, and optimistic. The conservative scenario should be built on assumptions you would be comfortable defending to a skeptical investor at 2:00 a.m. If the conservative scenario is still viable, you have something worth building.

CASH FLOW ANALYSIS CHECKLIST

- ◆ Calculate MRR across all subscriber tiers, net of discounts and trial-period concessions.
- ◆ **Project bookings, expansion, contraction, and churn separately** — not as a single 'net revenue' number.
- ◆ Model CAC fully-loaded, including all marketing and sales costs allocated to subscriber acquisition.
- ◆ Calculate LTV at your target churn rate and benchmark the LTV:CAC ratio against industry standards for your vertical.
- ◆ Model your infrastructure and hosting costs as a function of subscriber count, not as a fixed annual line item.
- ◆ Calculate cash runway under conservative, base, and optimistic scenarios.
- ◆ **Identify the revenue threshold at which the business reaches cash flow breakeven** — and the date that threshold is projected to be reached under each scenario.

PART V — MEASURED IMPLEMENTATION

CHAPTER FORTY-EIGHT

Proof of Concept Roadmap

★ NEW CONTENT

""The best way to predict the future is to create it.""

— Peter Drucker

If the cash flow analysis is the reality check of the Beyond Bedlam process, the Proof of Concept roadmap is the test that determines whether the reality is survivable. A Proof of Concept is not a prototype. It is not a mock-up or a design specification. A Proof of Concept is the smallest possible working version of the most critical unvalidated assumption in your business model — built, deployed, and measured against real conditions, with real users, before you commit to the full development investment.

I have been part of two major SaaS launches where a genuine POC would have changed everything. In one case, the platform was built to an elaborate spec over eighteen months before anyone outside the company used it. When users finally got their hands on it, they revealed a core workflow assumption that had been wrong from the beginning. The cost to rebuild the affected architecture was significant. A six-week POC with five paying customers could have surfaced that assumption in sprint three.

The goal of a POC is not to build something impressive. It is to find out what you don't know before you've committed everything to a direction that assumes you know it.

What to Prove and What to Defer

Not every element of a SaaS platform requires POC validation. Many technical decisions are well-understood and don't need to be proven through a real-world test. What the POC must validate are the assumptions that, if wrong, would materially change the business model.

These are typically: the core user workflow, the value proposition, the acquisition mechanism, and the support model. If you can validate these four assumptions with a POC before full development investment, you have dramatically reduced the risk profile of the full initiative.

Structuring the POC

The POC should be scoped to the single most critical unvalidated assumption — not every feature you eventually want to build. A well-scoped POC answers one question. It is built in the shortest time possible with the fewest resources necessary to get real signal. It is measured against specific, pre-defined success criteria. And it has a defined decision point: at the end of the POC, you will make one of three decisions. Proceed with confidence. Pivot with specific adjustments. Stop entirely.

POC ROADMAP CHECKLIST

- ◆ **Identify the single most critical unvalidated assumption in the business model** — the one whose failure would fundamentally change the investment decision.
- ◆ Define the minimum feature set required to test that specific assumption. Resist the temptation to add features that aren't needed for the test.
- ◆ **Recruit a small cohort of real target users** — ideally 3 to 10 — who match the primary persona definition. Internal users and friends are not substitutes.
- ◆ Define success criteria in advance, in measurable terms. 'Users seemed to like it' is not a success criterion.
- ◆ Set a fixed time and budget boundary for the POC. If it can't be completed within those constraints, the POC scope is too large.
- ◆ **Schedule a formal decision-point review at the end of the POC. Proceed, pivot, or stop** — based on what the data says, not on what you hoped to hear.

PART V — MEASURED IMPLEMENTATION

CHAPTER FORTY-NINE

Technology Roadmap

★ NEW CONTENT

""Give me six hours to chop down a tree and I will spend the first four sharpening the axe.""

— Abraham Lincoln

The technology roadmap is the operational translation of everything the Beyond Bedlam process has built. Every infrastructure alignment decision, every DevOps standard, every hosting determination, every third-party API evaluation — all of it must now be assembled into a coherent, sequenced plan that a development team can actually execute. The technology roadmap is not a wish list. It is a construction schedule.

I have seen technology roadmaps that were works of art — beautifully formatted, color-coded by workstream, richly annotated with dependencies. And I have seen those same roadmaps become expensive fiction by the end of sprint two, because they were built on assumptions about team capacity, vendor reliability, and integration complexity that nobody had actually stress-tested. A technology roadmap is only as valuable as the quality of its assumptions.

The Hierarchy of Technology Decisions

At the **foundational level** are the platform architecture decisions: cloud provider, database technology, application framework, and authentication model. These decisions are expensive to reverse because they affect every other technical component built on top of them. They deserve disproportionate deliberation upfront.

At the **structural level** are the integration decisions: which third-party APIs will be incorporated, how data will flow between systems, and what the security architecture will look like across the full stack. These decisions are moderately reversible.

At the **surface level** are the feature and UI decisions, which should be treated as hypotheses to be tested rather than specifications to be built to.

Phasing the Roadmap

The technology roadmap should be divided into clear phases, each tied to a specific business milestone rather than an arbitrary calendar date. Phase one should deliver the minimum viable infrastructure required to support the POC or the first paying subscriber — nothing more. Phase two should deliver the features and platform stability required to support the first commercial launch. Phase three and beyond should be treated as directional — the specific scope will be informed by what is learned in the earlier phases.

Phase	Focus	Success Criteria	Duration
Phase 0	Foundation & Architecture	Core platform operational; first test user can complete primary workflow	4–8 weeks
Phase 1	POC / MVP	Target users validated on core value proposition; pricing signal confirmed	6–12 weeks
Phase 2	Commercial Launch	First paying subscriber; support processes operational	8–16 weeks
Phase 3+	Growth & Scale	Defined by POC and launch learnings	Ongoing

ROADMAP CONSIDERATIONS

- ◆ Document all foundational architecture decisions with the rationale for each. Future team members will need this context.
- ◆ Map all integration dependencies and identify the critical path: which integrations must be functional before the first subscriber can use the product?
- ◆ Identify the top three technical risks in the plan and define explicit mitigation strategies for each.
- ◆ Ensure the roadmap phases are tied to business milestones, not just calendar dates.
- ◆ **Build a process for roadmap updates** — how often will it be reviewed, who can change it, and how will those changes be communicated?
- ◆ Separate the 'stable' tier of the roadmap (foundational architecture) from the 'flexible' tier (features and UI) in your planning documentation.

PART V — MEASURED IMPLEMENTATION

CHAPTER FIFTY

Point Analysis

★ NEW CONTENT

""Not everything that counts can be counted, and not everything that can be counted counts.""

— William Bruce Cameron

Throughout the entire Beyond Bedlam process — from the first stakeholder conversation through the technology roadmap — you have been accumulating intelligence. You now have more information about this initiative than anyone who hasn't done this work. Point Analysis is the mechanism for translating all of that accumulated intelligence into a defensible, prioritized decision framework.

It is the answer to the question that always arises at the beginning of implementation: with everything we know, what do we build first, what do we build second, and what do we not build at all?

The Scoring Framework

Point Analysis begins with a complete inventory of everything that could be built, changed, or integrated as part of the SaaS initiative. Each item on the list is then scored against a consistent set of criteria. I typically use five dimensions, each scored on a 1 to 10 scale:

Strategic alignment — How directly does this item advance the organization's stated goals? A feature that drives subscriber acquisition for a startup scores higher than a reporting enhancement.

Revenue impact — What is the projected effect on MRR? This includes both direct revenue generation and indirect revenue protection.

Development cost — How much time, talent, and money is required? For this dimension, the scoring is inverted — high cost scores low.

Risk of not doing it — What is the cost of deferring this item? Some features, if absent at launch, will be a constant friction point that generates negative feedback and elevated churn.

Time sensitivity — Is there an external deadline, competitive pressure, or market window that makes the timing of this item consequential?

The total score across these five dimensions creates a priority ranking that the whole team can see, discuss, and refine. The value of Point Analysis is not the score itself — it is the conversation it generates. When a development team and a management team look at the same scored list together, they will frequently disagree on the scores. Those disagreements are the analysis working as designed. The goal is not unanimity — it is shared understanding.

ROADMAP CONSIDERATIONS

- ◆ **Build a complete inventory of all potential development items** — features, integrations, infrastructure improvements, compliance requirements — before scoring begins.
- ◆ Establish the scoring criteria and the weighting of each dimension before scoring any individual item.
- ◆ Score each item as a team, not individually. The discussion that surfaces disagreements is the most valuable part of the exercise.
- ◆ Review the resulting priority list as a group. Do the top-ranked items match everyone's intuition about what should be built first?
- ◆ Use the Point Analysis output to drive the first sprint's scope.
- ◆ **Revisit the analysis at the beginning of each major phase** — the priority landscape will shift as the product evolves and the market responds.

PART V — MEASURED IMPLEMENTATION

CHAPTER FIFTY-ONE

Sales & Marketing Plan

★ NEW CONTENT

""Build something 100 people love, not something 1 million people kind of like.""

— Paul Graham

In Chapter Thirteen, we identified the target market. In Chapter Nineteen, we looked at the sales funnel. In Chapter Twenty, we worked through packaging and pricing. In Chapter Forty, we examined marketing automation. All of that analysis now needs to be synthesized into a single, coherent Sales and Marketing Plan — a document that describes, in operational terms, exactly how you will connect the product to the market.

Most SaaS organizations skip this step. They have individual plans — a marketing budget, a sales quota, a content calendar — but no integrated framework that connects all of those activities to a shared set of goals, metrics, and decision triggers. The Sales and Marketing Plan is the connector.

Defining the Go-to-Market Motion

A **sales-led motion** places relationship-driven selling at the center of acquisition. It works well when the product is complex, when the buyer needs education to understand the value, or when the annual contract value is high enough to justify the sales investment. It scales slowly and is expensive per acquisition, but it generates the richest customer intelligence available to the product team.

A **marketing-led motion** uses content, paid media, events, and brand to attract prospects who self-educate and then raise their hand to engage with sales. It scales faster than sales-led but requires consistent content investment and patience.

A **product-led motion** uses the product itself as the primary acquisition vehicle — through free trials, freemium models, or viral mechanics built into the product experience. It is the most scalable motion and the most capital-efficient — but it requires a product experience polished enough to sell itself.

The First 100 Subscribers

Before any of the above motions can operate at scale, the plan must account for the most important number in early SaaS growth: the first 100 paying subscribers. The path to 100 is almost never the same as the path to 1,000, and treating them as equivalent is a mistake that leads to premature scaling of channels that haven't been validated. The first 100 subscribers typically come from the founder's personal and professional network, from direct outreach to known prospects in the target market, or from early adopter programs built around the POC cohort.

SALES & MARKETING PLAN COMPONENTS

- **Define the primary go-to-market motion** — sales-led, marketing-led, or product-led — and the rationale for that choice given the product, the market, and the available resources.
- Describe the path to the first 100 subscribers: specific acquisition channels, specific prospect lists or communities, specific outreach strategies.
- **Define the lead qualification criteria** — what does a qualified prospect look like, and at what point do they enter the sales process?
- Map the sales process from first contact to closed subscription, including the handoffs between marketing and sales functions.
- Define the onboarding experience for new subscribers: who is responsible, what does it include, and what does success look like at 30, 60, and 90 days?
- Set MRR targets for months 1, 3, 6, and 12. Tie those targets to specific acquisition channels and activities so progress can be diagnosed, not just measured.

PART V — MEASURED IMPLEMENTATION

CHAPTER FIFTY-TWO

Implementation Timeline

★ NEW CONTENT

""A goal without a timeline is just a dream.""

— Robert Herjavec

We have arrived at the final chapter of the Beyond Bedlam process, and I want to begin with the most important thing I can tell you about implementation timelines: they will be wrong. Not because you've done poor planning. Not because your team is insufficiently talented. They will be wrong because the act of building a SaaS service generates information that didn't exist when the timeline was created, and that information changes things.

I have been part of SaaS implementations that ran exactly on schedule — and those are almost always the ones where someone in a position of authority was suppressing the signals that the timeline needed to change. The projects that succeeded were the ones where the timeline was treated as a living document, updated with discipline and transparency, and where the team felt safe saying "we underestimated this" before it became a crisis rather than after.

The Master Timeline Structure

The Implementation Timeline is the synthesis of every workstream that has emerged from the Beyond Bedlam process. It integrates the technology roadmap phases, the Sales and Marketing Plan milestones, the cash flow model's revenue and burn assumptions, and the management team's decision checkpoints into a single view of what needs to happen in what sequence.

At the **quarterly milestone** level, the timeline describes what the business will have accomplished at the end of each quarter: where the product will be, what the MRR will be, what the team structure will look like.

At the **monthly deliverable** level, it describes the specific outputs required in each month to stay on track for the quarterly milestone. If a monthly deliverable slips, the impact on the quarterly milestone must

be explicitly assessed and communicated — not absorbed silently.

At the **sprint task** level, it describes the two-week increments of development work that produce the monthly deliverables. Work that cannot be traced to a milestone is, by definition, lower priority than work that can.

The 30-60-90-Day Framework

For organizations beginning their implementation, the most valuable use of the timeline is as a 90-day plan with maximum specificity. The first 30 days should be planned at the task level — every significant action, assigned to a specific person with a specific due date. The second 30 days should be planned at the deliverable level. The final 30 days should be planned at the milestone level. This approach is not an admission that the later periods are less important. It is an acknowledgment that planning in the highest resolution for work that is far enough away to be fundamentally uncertain is an expensive form of fiction writing.

"The bedlam is not in the business. It never was. It is in the space between where you are and where you need to be — and in the willingness to stop digging long enough to draw a map."

A Final Word

Thirty years ago, I plugged a laptop into a fax machine phone line in a downtown Austin office building and tried to sell legislative tracking software over a 54Kb dial-up connection. I had no framework. I had no roadmap. I had ambition, a business partner who knew how to build things I had no idea how to build, and the arrogance — Saaspreneur arrogance — to believe it would work.

It did work, eventually, in ways neither of us had planned and through paths neither of us had anticipated. We survived by listening more than we spoke, by adjusting faster than our competitors, and by never confusing activity for progress.

The Beyond Bedlam process is the distillation of everything I wish we had known in that hotel ballroom in October 1996, when thirty Texas politicos were waiting to see a platform that our lead developer had deleted at 11 p.m. the night before. It is the framework I have applied, in various forms, to every SaaS engagement since. It is not a guarantee of success. Nothing is. But it is a map — and a map is something.

Quit digging. Draw a map. Then start building.

IMPLEMENTATION TIMELINE CHECKLIST

- ◆ Build the master timeline at three levels: quarterly milestones, monthly deliverables, and sprint tasks. Ensure each level traces to the level above it.
- ◆ Plan the first 30 days at task-level specificity. Plan days 31–60 at deliverable-level. Plan days 61–90 at milestone-level.
- ◆ **Define and document the formal triggers for timeline review** — and the process for conducting that review and communicating any revisions.
- ◆ Integrate the technology roadmap phases, the Sales and Marketing Plan milestones, and the cash flow model assumptions into a single unified view.
- ◆ **Establish a weekly team sync cadence** — brief, disciplined, focused on blockers and timeline status.
- ◆ Assign a single owner to the master timeline. Someone must be responsible for its accuracy and currency at all times.
- ◆ At each quarterly milestone, conduct a formal retrospective: what did we learn that changes the plan for the next quarter?