PRE-BUILD ENGINEERING GUIDE

Before Anyone Builds Your AI System or App.

The six written blueprints — plus one automated AI & security release gate — to lock on paper before a single line of production code is written. Every document is shown filled in for one realistic system, with the prompt that drafts it and what to check before you sign off.

Want it built from real architecture, not a vibe?

Book a 15-minute intro call. Bring the AI product, client portal, or workflow you want built — and whatever plan you have so far. Every quote is tailored in writing after scoping.

Book 15-min call →

01 WHAT IS INSIDE

- All six pre-build documents, in the exact order they are written and approved.
- Each document filled in for one realistic example so you can see what principal-grade clarity looks like.
- A ready-to-run prompt for each document so you can draft your own pack before calling any engineer.
- Document 07: The AI Eval & Security Release Gate — how Sage Ideas proves the finished system works.
- A six-box pre-build readiness check to run before approving work.

02 HOW TO USE THIS GUIDE

Every example in this guide follows one illustrative build from start to finish: **Atlas Ops & Client Portal** — a B2B services & operations team that wants a grounded AI knowledge assistant, client onboarding portal, online agreement signing, Stripe billing, and an operator dashboard with strict tenant isolation.

Run the prompts in order in one AI session and paste each approved document into the next prompt. Replace anything highlighted in **[Lime Brackets]** with your real business details.

> **Your job is the deciding, not the coding.** Modern tooling makes building fast — which means the written plan is the only thing that prevents building the wrong system fast. An hour reading and signing off on these six documents is the cheapest point in the entire project to change your mind.

DOCUMENT 01 · PRD

The Product Requirements Document

What the system does, feature by feature, with verifiable "Done When" acceptance tests.
If a feature is not on this list, it is not in v1.

1 · PRODUCT REQUIREMENTS

EXAMPLE · V1

Atlas Ops & Client Portal

Product requirements · version 1

GOAL

Clients can review blueprints, e-sign agreements, pay invoices, and inspect project milestones in one private workspace — while internal staff answer questions via a grounded AI assistant without leaking cross-client data.

WHO USES IT

Prospective clients, signed clients, and internal operators (admin).

FEATURES & VERIFIABLE ACCEPTANCE GATES

FEATURE	WHO USES IT	DONE WHEN (TESTABLE GATE)
Tailored scope intake	Prospect	Brief is validated, saved to CRM pipeline, and operator is emailed within 10s.
Agreement & blueprint e-sign	Client	Client reviews document, submits typed name + consent; timestamp and hash stored.
Deposit & milestone billing	Client	Stripe Checkout webhook marks invoice paid idempotently and sends receipt.
Grounded AI assistant	Client / Ops	Answers cite retrieved project spans only and abstain when evidence is absent.
Operator cockpit	Admin	Pipeline, contracts, client infra links, and unpaid invoices visible on one screen.

EXPLICITLY NOT IN THIS VERSION

Custom ERP two-way sync, autonomous outbound emails without human approval, or public self-serve price tiers.

PROMPT · DOCUMENT 1 (PRD)

I want to build **[what you want built]** for my business, **[one line on what your business does]**. Help me write a Product Requirements Document. Ask me up to 10 clarifying questions first — one at a time — about who will use it, what each role must accomplish, and what "finished" means. Then output a one-page PRD with: a one-line goal, user roles, a features table (Feature, Who uses it, Done When), and an explicit "Not in v1" list. Use plain English and zero vague buzzwords.

Read it for: Anything you silently assumed was included, and whether every "Done When" test can actually be verified by you in under two minutes.

DOCUMENT 02 · TRD

The Technical Requirements Document

The stack, hosting, auth, backups, and external integrations decided once in writing — so nobody guesses halfway through the build.

2 · TECHNICAL REQUIREMENTS

EXAMPLE · V1

Atlas Ops & Client Portal

Technical architecture & stack decisions · version 1

AREA	CHOICE	WHY, IN PLAIN ENGLISH
Web App & API	Vite / Next.js + Edge Functions	Fast static public pages paired with isolated serverless APIs.
Database & Auth	PostgreSQL + Supabase RLS	Strict row-level isolation so one client can never query another.
Hosting & CDN	Vercel + Cloudflare	Zero-downtime deploys, instant rollback, global edge caching.
Payments	Stripe Checkout + Webhooks	Card data never touches our servers; webhooks verify signatures.
Transactional Email	Resend (verified domain)	Signed receipts, portal links, and suppression tracking from our domain.
AI & Eval Harness	Server-side LLM + CI Gate	API keys stay server-only; golden & red-team evals block regressions.
Backups & Logs	Daily PITR (30d) + Sentry	Point-in-time recovery plus structured error alerts with zero PII dumps.

CONNECTS TO (EXISTING TOOLS)

Cal.com scheduling, Stripe billing, Resend email, and GitHub CI release gates.

OPEN DECISIONS BEFORE CODE STARTS

Will clients sign in via email + magic link only, or email + password with optional SSO?

PROMPT · DOCUMENT 2 (TRD)

Here is my approved PRD: **[paste Document 1]**. Draft a Technical Requirements Document. Specify the app framework, database, hosting, authentication, payment handling, email, backups, observability, and how it connects to my current tools: **[list your existing tools — e.g. Stripe, HubSpot, Google Workspace, Cal.com]**. Output a table (Area, Choice, Plain-English Why) and end with open decisions that must be locked before code starts.

Read it for: Every tool you already pay for appearing in "Connects To", zero vendor lock-in surprises, and a plain-English "Why" you actually understand.

DOCUMENT 03 · APP FLOW

App Flow & Failure Paths

Every screen a user touches, what every button does, and — most importantly — where they land when something breaks. Zero dead ends.

3 · USER JOURNEYS & RECOVERY

EXAMPLE · V1

Atlas Ops & Client Portal

Happy paths & unhappy paths · version 1

MAIN ENTRY PATH

Public Studio & How I Work	→	Tailored Scope Brief / Call → Private Client Portal Active

CORE CLIENT JOURNEYS

Open Blueprint / Contract Link	→	Review Sections + Consent → Signed Receipt + Timestamp Saved
Approve Delivered Milestone	→	Stripe Checkout (PCI Isolated) → Invoice Paid + Receipt Emailed
Ask Portal AI Assistant	→	Verify HMAC Session + Scope → Grounded Answer Citing Facts

WHEN SOMETHING GOES WRONG (FAILURE RECOVERY)

Payment Declined / Timeout	→	Return to Invoice · Clear Reason → Retry Card or Request Wire Link
Forwarded Portal URL (No Session)	→	Fail Closed (HTTP 401) → Prompt Client Email Verification
AI Upstream Error / Low Evidence	→	Abstain — Never Hallucinate → 1-Click Route to Operator Inbox

PROMPT · DOCUMENT 3 (APP FLOW & FAULTS)

Using this PRD: **[paste Document 1]**, map every user journey per role. List every screen, primary action, and next state. Then explicitly map the unhappy paths: failed payment, expired/forged login link, missing form fields, double submission, and AI provider outage. Every error state must end in a clear recovery action with zero dead ends.

Read it for: Dead ends. Walk through every red box as an impatient customer on a phone. If an error leaves them stranded, fix the flow on paper now.

DOCUMENT 04 · DESIGN BRIEF

The Design Brief & Token Contract

Exact colors, typography, spacing, accessibility contrast, and tone rules locked up front so screen ten looks as disciplined as screen one.

4 · DESIGN SYSTEM CONTRACT

EXAMPLE · V1

Sage Ideas Architectural System

Design brief · version 1

LOCKED BRAND PALETTE

Obsidian Ink	Signal Lime	ArchOS Cyan	Editorial Paper
#090C10 · Canvas	#DBFF27 · Primary CTA	#0ED3CF · Systems	#F0ECE1 · Copy

TYPOGRAPHY & INTERFACE RULES

ELEMENT	NON-NEGOTIABLE RULE
Typography	Bricolage display headlines, Plus Jakarta Sans body (≥15px), monospace technical labels (≥12px).
Geometry	Sharp architectural corners (≤4px radius), hairline 1px borders, zero blurry drop-shadow bloat.
Primary Actions	One Signal Lime primary CTA per view; minimum 44×44px touch targets on mobile.
Accessibility	WCAG AA contrast (≥4.5:1 text), keyboard navigable, respects prefers-reduced-motion.
Voice & Honesty	Direct engineering prose. Real vs. simulated explicitly labelled. Zero invented social proof.

PROMPT · DOCUMENT 4 (DESIGN BRIEF)

Write a one-page Design Brief for **[what is being built]**. My brand colors are **[hex codes or brand feel]**, my fonts are **[font names]**, and my primary users are **[audience + desktop vs. mobile split]**. Specify exact color roles, type scale, button hierarchy, form rules, mobile breakpoints, accessibility contrast, and microcopy tone so an engineer never has to guess.

Read it for: Specific hex codes, font names, and concrete rules. If the brief could describe any generic SaaS template on earth, tighten it until it sounds unmistakably like your brand.

DOCUMENT 05 · SCHEMA & SECURITY

Backend Schema & Access Matrix

Exactly what data is stored, how records link together, and who is allowed to read, edit, or delete every row. This document protects your clients.

5 · DATA MODEL & OWASP ACCESS BOUNDARIES

EXAMPLE · V1

Atlas Ops & Client Portal

Plain-English schema & isolation rules · version 1

RECORD	WHAT IS STORED	WHO CAN READ / WRITE (RLS)
Prospect	Name, email, company, scope brief, stage	Operator only (service-role; zero public read).
Proposal	Line items, tailored fee, deposit/balance, status	Operator; assigned client via opaque publicId only after approval.
Agreement	Kind (NDA/Blueprint/SOW/MSA), sections, signed hash	Operator; assigned client via sent link; drafts hidden.
Project & Milestones	Status, deliverables, sign-off name & timestamp	Operator; verified client session (HMAC + email match).
Invoice	Invoice number, Stripe payment intent ID, paid_at	Operator; verified client. Card numbers NEVER stored.
Audit Log	Actor, action, target ID, timestamp, IP hash	Append-only by server; Operator read-only.

NON-NEGOTIABLE SECURITY RULES

Every table enables PostgreSQL Row-Level Security (deny-all default for anonymous callers). Admin endpoints require constant-time token or allowlisted JWT verification. Payment card numbers never touch our database. Private portal & API responses send `Cache-Control: no-store`.

PROMPT · DOCUMENT 5 (DATA SCHEMA & ACCESS)

Using Document 1 **[paste PRD]** and Document 2 **[paste TRD]**, write a plain-English Backend Schema & Access Matrix. For every table/entity, list what fields are stored, foreign-key relationships, and exact Read/Write/Delete permissions per role. Flag all PII, credentials, or payment data — and list what we must NEVER store.

The one document you never skip: A visual glitch can be fixed in twenty minutes. One client seeing another client's data — or an unauthenticated endpoint leaking billing details — destroys trust permanently.

DOCUMENT 06 · IMPLEMENTATION PLAN

The Staged Implementation Plan

Build order sequenced riskiest-part-first, with an inspectable client sign-off gate at every stage before the next stage begins.

6 · STAGED DELIVERY & SIGN-OFF

EXAMPLE · V1

Atlas Ops & Client Portal

Five vertical slices · riskiest integration proven early

STAGE	WHAT SHIPS	YOU SIGN OFF WHEN (REAL TEST)
1. Auth, RLS & Schema	Database tables, strict RLS, login & fail-closed APIs	Automated test proves Client A gets 401/403 on Client B's records.
2. Payments & Contracts	Stripe Checkout webhook + e-sign agreement receipt	Test checkout marks invoice paid idempotently and emails signed PDF.
3. Core AI & Eval Gate	Grounded RAG retrieval + 18-case golden/red-team suite	CI blocks merge when a bad prompt drops faithfulness below floor.
4. Client & Admin UI	Desktop + mobile responsive portal & operator cockpit	You run one complete client lifecycle end-to-end on phone & desktop.
5. Hardening & Launch	Accessibility audit, load check, rollback test, handoff	0 critical axe/security issues, live health green, runbook delivered.

WHY RISKIEST PART FIRST

Auth boundaries, Stripe webhooks, and AI retrieval accuracy are verified in Stages 1–3 with real automated tests — before cosmetic polish is layered on top.

PROMPT · DOCUMENT 6 (IMPLEMENTATION PLAN)

Here are Documents 1–5 for **[what is being built]**: **[paste Documents 1–5]**. Break delivery into 4–6 sequential vertical stages. Put the highest-risk integration first and explain why. For each stage, list: what ships, what inputs you need from me, how I will personally test it, and the sign-off criterion before the next stage begins.

Read it for: A concrete test you can run yourself at every milestone. If you cannot test progress until the final week, surprises are guaranteed.

DOCUMENT 07 · THE SAGE IDEAS STANDARD

The AI Eval & Security Release Gate

Most web agencies stop at Document 6. For AI systems and production portals, Sage Ideas adds one non-negotiable 7th artifact: an automated gate that can actually say no.

7 · PROOF-DRIVEN RELEASE CONTRACT

PRINCIPAL STANDARD

No Fake Green — Evidence Stays With Every Claim

Automated verification run on every commit before production deploy

GATE LAYER	WHAT IT TESTS AUTOMATICALLY	FAILURE ACTION
Golden Eval Dataset	Real representative user questions scored against grounded sources & rubrics.	Blocks merge if pass rate falls below ratcheted floor.
Adversarial Probes	Prompt injection, jailbreak, PII extraction, and cross-tenant query attempts.	Immediate CI failure; zero silent regression.
OWASP ASVS Checks	Constant-time auth, rate limiting, input schemas, no-store headers, SSRF/XSS guards.	Fail-closed 401/400/429 unit regression suite.
Visual & A11y Gate	Playwright desktop/tablet/mobile screenshots, zero clipped media, 0 axe violations.	Blocks release until layout & contrast pass.
Machine Scorecard	Signed JSON receipt written only when 100% of gates exit green.	A failing run writes nothing.

PROMPT · DOCUMENT 7 (AI EVAL & SECURITY GATE)

Based on Documents 1–6 for **[what is being built]**, design a CI Release Gate specification. List 12 golden test cases, 6 adversarial/abuse probes (including prompt injection and cross-tenant access), minimum pass thresholds, and the exact OWASP ASVS checks that must pass before any deploy reaches production.

Why teams hire a Principal Architect: Anyone can wire an LLM API in an afternoon. Keeping it accurate, private, fast, and regression-free six months after launch is the real engineering job.

08 · PRE-BUILD SCORECARD

The Check Before Building Starts

Ask your builder for each of the six documents by name. Tick a box only when you have actually read it, understood it, and agreed to it in writing.

- [ ] 1. A Product Requirements list with verifiable "Done When" gates and explicit exclusions
- [ ] 2. A Technical Stack document fixing tools, hosting, auth, backups, and existing integrations
- [ ] 3. An App Flow map covering every screen and every error/failure recovery path
- [ ] 4. A Design Brief with our real hex colors, typography, mobile rules, and accessibility floor
- [ ] 5. A Backend Schema & Access Matrix proving strict client data isolation (RLS)
- [ ] 6. A Staged Implementation Plan putting the riskiest part first with sign-off gates

SCORE INTERPRETATION

INTERACTIVE VERSION AT SAGEIDEAS.DEV/HOW-I-WORK

BOXES TICKED	WHAT IT MEANS
6 of 6	**Green light.** There is a real engineering plan. Building can start safely.
3 to 5	**Pause and close the gaps.** Writing the missing blueprints takes hours; rebuilding later takes weeks.
0 to 2	**Stop before spending a dollar.** Nobody has decided what is being built yet.

Run this interactively online: Visit **sageideas.dev/how-i-work** to score your current readiness live, generate starter prompts for your specific build, or send your scope directly to Jason Teixeira for a straight read on fit.

09 · NEXT STEP

What To Do Next

Use this guide whether you build in-house, hire another team, or partner directly with Sage Ideas Studio.

1. **Ask for all six blueprints by name.** Any serious principal engineer can draft them cleanly before coding; a builder who refuses is telling you how the project will go.
2. **Read and challenge every assumption before signing off.** Changing one sentence in a PRD costs nothing. Changing a database schema after launch costs weeks.
3. **Approve stage by stage against real tests.** Use the implementation plan and automated release gate to verify each deliverable as it lands — never all at the end.

KEEP THIS RULE

When modern tools make coding fast, the written blueprint is the only thing that slows work down in the right place. **Decide on paper, then build** — and you only pay for the system once.

The building is the fast part. The architecture is what protects your investment.

Want a second pair of principal engineering eyes on your six blueprints — or want Sage Ideas to architect, build, and harden the system end-to-end? Every engagement is scoped and quoted in writing after a 15-minute intro call.

Book 15-min call →

BOOK: sageideas.dev/book · PIPELINE: sageideas.dev/how-i-work · EMAIL: hello@sageideas.dev

EXPLORE THE SAGE IDEAS ECOSYSTEM — LIVE PROOF

sageideas.dev	Principal Studio, 6-Blueprint Delivery Pipeline, Interactive Labs & Verified Proof Ledger.
archos.sageideas.dev	ArchOS Studio — 525 interactive systems architecture blueprints & 520 technical deep-dives.
crucible.sageideas.dev	CRUCIBLE OS — Course 0 (44 server lessons), 74 WASM/SQL/DSA drills & 12 Firecracker VMs.