AZROTIC · FREE SAMPLE

AI-NATIVE SDLC

BEYOND THE DEMO GUIDEBOOK

The step-by-step playbook to build **real, production-ready apps with AI**, even if you have **never written code**: from first idea to live, monitored software.

19 phases | 40 copy-paste prompts | 9 quality gates | 30-point readiness checklist

Authored by the Azrotic Team

Azrotic AI & Tech Solution · azrotic.com

Contents

★ Start here 4

★ The 19-phase journey 5

★ Fast track: your first project 6

★ Who does what 7

★ Folder structure 8

STAGE A · PLAN IT RIGHT

00 Discovery — Understand the business problem 10

01 Requirement Engineering — Write the business requirements 13

02 Product Definition — Shape requirements into a product 17

03 Architecture Engineering — Design before you build 19

04 Architecture Freeze — Lock the design: it becomes law 24

05 Development Constitution — Write the supreme law for the AI 26

06 Master Implementation Guide — Give the AI its working context 28

07 Roadmap Generation — Break the system into milestones 32

08 Task Breakdown — Turn features into atomic tasks 34

09 ADR Framework — Record every important decision 36

STAGE B · BUILD IT RIGHT

10 Governance Layer — Verify before every implementation 40

11 Task Execution — Implement exactly one task 42

12 Validation Gates — Pass every gate before merging 46

13 Execution Reports — Document what was built 49

STAGE C · PROVE AND SHIP IT

14 Milestone Review — Step back at each milestone 52

15 Local and Staging Validation — Prove it works in a clean room 54

16 Dogfooding — Use it on real projects, in stages 58

17 Release and Marketplace — Ship with everything in place 61

STAGE D · RUN IT

18 Continuous Governance — Operate, maintain and improve 66

TOOLKIT

T T1 · Templates 69

T T2 · Production-readiness checklist 71

T T3 · Quick reference 73

T T4 · Glossary 75

T About Azrotic 75

Start here

This guidebook takes you from an idea to production-ready, industry-ready software in **19 phases**, in a fixed order. Each phase is a short, practical page: do the steps, paste the prompts, save the files, tick **Done when**, pass the gate. Then turn the page.

The usual way

Idea → Code → Fix → Deploy

Speed for a day, chaos for a month: invented architecture, undocumented decisions, nobody sure it works.

The AI-native way (this guidebook)

Discover → Define → Design → Freeze → Rules → Plan → Build → Prove → Ship → Run

AI drafts and implements one approved task at a time. People decide on evidence.

How every phase page works

1 · The hero

Phase number, goal and where you are on the 19-step track.

2 · At a glance

What you need, the exact files you produce, who owns it.

3 · Do this + prompts

Numbered steps and copy-paste prompts with yellow placeholders.

4 · Done when + gate

A checklist, mistakes to avoid, and who signs the gate.

What you need

- **An AI assistant** for planning prompts (any capable chat assistant).
- **An agentic coding tool** for build prompts: Claude Code, Cursor, Codex, Antigravity or similar.
- **A code host with CI** (for example GitHub with Actions) so checks run on every change.

- **A Technical Approver**: an experienced engineer (part-time or external is fine) who signs the Architecture Freeze, High-risk changes and the Production Readiness Review.
- Non-technical owners can run everything else. Without a Technical Approver, do not call the result production-ready.

Prompt placeholders look like `{{NAME}}`. Replace every one; write "unknown" rather than guessing. All 40 prompts are also in the starter kit's `prompts/` folder as text files.

THE ONE RULE

Never let AI write production code until requirements, architecture, governance, roadmap, task breakdown and decision records are approved and baselined. Code is the final artifact, not the starting point.

The 19-phase journey

Phases run top to bottom. Phases 10 to 13 repeat for every task; Phase 14 repeats for every milestone; Phase 18 never stops while the product is live. Amber tags mark the nine quality gates.

STAGE A
Plan it right

- 00 Discovery — GATE · Scope sign-off
- 01 Requirement Engineering
- 02 Product Definition — GATE · Requirements baseline
- 03 Architecture Engineering
- 04 Architecture Freeze — GATE · Architecture Freeze
- 05 Development Constitution
- 06 Master Implementation Guide
- 07 Roadmap Generation
- 08 Task Breakdown — GATE · Plan ready
- 09 ADR Framework

STAGE B
Build it right

- 10 Governance Layer — GATE · Governance (every task)
- 11 Task Execution
- 12 Validation Gates — GATE · Validation (every pull request)
- 13 Execution Reports

STAGE C
Prove and ship it

- 14 Milestone Review
- 15 Local and Staging Validation — GATE · Production Readiness Review
- 16 Dogfooding — GATE · Rollout verification (each stage)
- 17 Release and Marketplace — GATE · Release approval and client acceptance

STAGE D
Run it

- 18 Continuous Governance

THE FLOW IN ONE LINE

Business Idea → Discovery → Requirements → Architecture → Freeze → Constitution → Implementation Guide → Roadmap → Task Breakdown → ADR → Governance → AI Execution → Validation → Reports → Reviews → Staging Validation → Dogfooding → Release → Continuous Governance

Fast track: your first project in 8 moves

Never done this before? Follow these eight moves in order. Each one unlocks the next.

01 Run discovery
What problem, for whom, how will you measure success, which risk tier? Phase 0 · prompt 0.1

02 Write the requirements
BRD, PRD, FRS, NFR with the AI; you review every line. Phases 1–2 · prompts 1.1–2.1

03 Design and freeze the architecture
Start from the simple default, record decisions as ADRs, lock it. Phases 3–4 · prompts 3.1–4.1

04 Set the rules and context
Constitution, AGENTS.md, Master Implementation Guide, tests, CI, monitoring. Phases 5–6 · prompts 5.1–6.4

05 Plan the work
Roadmap with M0 Foundation first, then atomic tasks with IDs; set up ADRs. Phases 7–9 · prompts 7.1–9.2

06 Run the execution prompts
First run once, then continuous: one task, tests first, CI, human merge. Phases 10–13 · prompts 10.1–13.1

07 Prove it
Milestone reviews in staging, security, load, accessibility, drills, readiness review. Phases 14–15 · prompts 14.1–15.4

08 Ship and run it
Staged rollout, release, client handover, then continuous governance. Phases 16–18 · prompts 16.1–18.2

THE ONE MISTAKE TO AVOID

Asking the AI to "just build the feature" before the documents exist. If the architecture is not frozen and the task is not in TASK_BREAKDOWN.md, the answer is always: stop and finish the planning first.

Folder structure

Create this structure on day one (the starter kit includes it). The yellow tag shows the phase that creates each file. Every file has one job, so nothing is written twice.

```
my-app/
├── AGENTS.md                      rules every AI agent auto-loads                  P5
├── CLAUDE.md                      one line: @AGENTS.md (Claude Code)               P5
├── README.md                      what it is, how to run it                        P6
├── CHANGELOG.md                   release notes                                    P17
├── AI_DEVELOPMENT_CONSTITUTION.md supreme law                                      P5
├── MASTER_IMPLEMENTATION_GUIDE.md how code is built here                           P6
├── FINAL_ARCHITECTURE_FREEZE.md   locked design, changes only by ADR               P4
├── IMPLEMENTATION_ROADMAP_MVP.md  milestones, M0 Foundation first                  P7
├── TASK_BREAKDOWN.md              task cards + the only task status                P8
├── EXECUTION_TRACKER.md           AI memory between sessions                       P11
├── IMPLEMENTATION_REVIEW.md       milestone and rollout reviews                    P14
├── docs/
│   ├── discovery/                 BUSINESS_DISCOVERY.md                            P0
│   ├── requirements/              BRD PRD FRS NFR .md, interviews/                 P1-2
│   ├── architecture/              HLD DATA_MODEL SECURITY_PRIVACY .md              P3
│   ├── quality/                   TEST_STRATEGY.md                                 P6
│   ├── operations/                DEVOPS_RELEASE RUNBOOK .md, postmortems/         P6 P17 P18
│   ├── release/                   PRODUCTION_READINESS_REVIEW HANDOVER .md, guides/ P15 P17
│   └── RISK_AND_DEBT_REGISTER.md  risks and tech debt, each owned                  P1
├── ADR/                           README.md (index), TEMPLATE.md, ADR-0001-*.md    P3 P9
├── reports/                       <task-id>.md, one per task                       P11-13
├── prompts/                       the 40 prompts in this guidebook                 all
├── api/                           openapi.yaml (API contract)                      P3
├── db/migrations/                 expand, migrate, contract                        P11
├── infra/                         environments as code                             M0
├── src/ tests/                    application code; unit, integration, e2e         P11
├── .github/                       ci.yml, pull_request_template.md, CODEOWNERS     P6
└── .env.example                   config names, never values                       P6
```

Always required

Every file above, at a depth that matches your risk tier. Testing, security, operations and release documents are not optional in production software.

Only when they apply

`api/` if you expose an API · `docs/release/guides/` if you have end users or client admins · `postmortems/` after incidents · marketplace files if you distribute through a store.

STAGE A · PHASES 0–9

A

Plan it right

From idea to signed requirements, a frozen architecture, the rules for AI and a task-level plan.

00	**Discovery** Understand the business problem	GATE
01	**Requirement Engineering** Write the business requirements	
02	**Product Definition** Shape requirements into a product	GATE
03	**Architecture Engineering** Design before you build	
04	**Architecture Freeze** Lock the design: it becomes law	GATE
05	**Development Constitution** Write the supreme law for the AI	
06	**Master Implementation Guide** Give the AI its working context	
07	**Roadmap Generation** Break the system into milestones	
08	**Task Breakdown** Turn features into atomic tasks	GATE
09	**ADR Framework** Record every important decision	

PHASE 0 · DISCOVERY

00 Understand the business problem

Get crystal clear on why this software should exist, for whom, and how you will measure success. Every later document builds on this.



YOU NEED

- Your idea or the client's request
- Known budget, deadline and constraints
- The people who will decide

YOU PRODUCE

- `docs/discovery/BUSINESS_DISCOVERY.md (vision, problem, stakeholders, goals, scope, risk tier)`

WHO OWNS IT

You (Product Owner) answer and approve. The AI drafts.

Do this

1. Create the project folder from the starter kit (see Folder Structure).
2. Paste prompt 0.1. Answer its questions with real facts; write 'unknown' instead of guessing.
3. Validate the need: talk to 5 to 10 people who have the problem, or run a manual pilot.
4. Set the risk tier with the four questions below. The highest answer wins.
5. Run prompt 1.3 on the draft and resolve every High issue.
6. Sign off the scope.

Pick your risk tier (highest answer wins)

Question	Tier 1	Tier 2	Tier 3
Who uses it?	Internal staff	External users or customers	The public at scale, or public sector
What data?	Staff identity only	Personal data (names, emails, accounts)	Health, payment-card, children's, biometric
Cost of a one-day outage?	Inconvenience	Revenue or reputation	Safety, legal or major financial harm
Regulated?	No	Privacy law applies	Sector rules or audits (get qualified legal advice)

The tier sets how deep every later control goes: testing, security baseline, environments, backups, review. Rigor is mandatory; over-engineering is not.

DONE WHEN

- [ ] Problem stated without naming a solution
- [ ] Every goal has a number or observable signal
- [ ] Out of scope listed, each with a reason
- [ ] Risk tier chosen with reasons
- [ ] Product Owner and Technical Approver named

AVOID

- × Writing 'an app like X' instead of a problem
- × Choosing a lower tier to save effort
- × Building before anyone has confirmed the problem

QUALITY GATE
Scope sign-off

Pass when: All Done-when items true; P1.3 shows no High issues
Signed by: Product Owner (and client sponsor)

PROMPT 0.1 Discovery interview and Business Discovery document

COPY · PASTE

USE WHEN	First thing on any new product or client request.
YOU GIVE	The idea in your own words, known constraints, and who decides.
SAVES TO	`docs/discovery/BUSINESS_DISCOVERY.md`
VERIFY	Every goal has a number. Out of scope is not empty. Approvers are named. Inferences are marked ASSUMPTION.

```
ROLE
You are a senior product analyst. You are concrete, careful and
sceptical of assumptions, including mine.

GOAL
Write docs/discovery/BUSINESS_DISCOVERY.md: the vision, problem,
stakeholders, success metrics and scope for this product.

INPUTS
Idea or request: {{IDEA}}
Known constraints (budget, deadline, team, legal): {{CONSTRAINTS}}
Decision-makers: {{STAKEHOLDERS}}

STEPS
1. Ask me up to 10 questions, most important first. Then stop
   and wait for my answers. Do not draft yet.
2. After I answer, write these sections:
   Vision (one paragraph)
   Problem statement: who has it, how they cope today, its cost
   Users and personas: 2 to 4, with goals and frustrations
   Business goals and success metrics, each measurable
   Scope for release 1, and Out of scope with a reason each
   Stakeholders and approvers: Product Owner, Technical
   Approver, client sponsor
   Constraints, Assumptions (each testable), Risks
   Validation plan: how we confirm the need before building
   Risk tier: answer the four questions, highest answer wins:
     Users? internal=1, external=2, public at scale=3
     Data? staff only=1, personal=2, health/payment/children=3
     One-day outage? inconvenience=1, revenue=2, serious harm=3
     Regulated? no=1, privacy law=2, sector rules=3
3. End with OPEN QUESTIONS.

RULES
- Use only facts I gave you. Label anything inferred ASSUMPTION.
- Never invent numbers, market data, competitors or quotes.
- Describe the problem, not a solution. No technology choices.
```

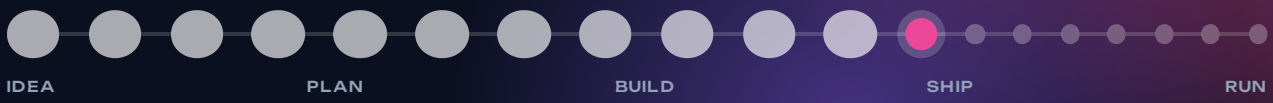
```
OUTPUT
Markdown for docs/discovery/BUSINESS_DISCOVERY.md, then OPEN
QUESTIONS.
```

11

PHASE 11 · TASK EXECUTION

Implement exactly one task

Now, and only now, the AI writes code. One task at a time, tests first, in its own branch and pull request, then it stops. A person reviews and merges.



YOU NEED

- Everything from Phases 0 to 10 signed off
- An agentic coding tool (Claude Code, Cursor, Codex, Antigravity or similar)

YOU PRODUCE

- `One branch and one pull request per task`
- `reports/<task-id>.md`
- `EXECUTION_TRACKER.md updated`

WHO OWNS IT

AI implements and reports. A human Reviewer merges; the Technical Approver reviews High-risk tasks.

Do this

1. First time only: paste prompt 11.1 (bootstrap). It creates the tracker and delivers the first task.
2. To build a task you choose: paste prompt 11.2 with its ID.
3. Every session after that: paste prompt 11.3. It picks the next Ready task, delivers it, and stops.
4. Review each pull request (Phase 13), merge it, and repeat.

How every task travels (one in, one out)

1. Governance check
2. ADR if triggered (Proposed, STOP)
3. Branch
4. Tests first
5. Implement one task
6. Run gates (max 3 attempts)
7. Report + tracker in same branch
8. Pull request
9. Human review and merge

Situation	Use
Brand-new project, empty tracker	**11.1** First run (once)
You want one specific task built	**11.2** Single task
Every session after that	**11.3** Continuous (next Ready task, then stop)

EXECUTION_TRACKER.md: the AI's memory between sessions

```
Current Milestone:  M3          Current Epic: M3-E5
Current Feature:    M3-E5-F3    Current Task: M3-E5-F3-T2
Completed Tasks:    M3-E5-F2-T1 ... T4, M3-E5-F3-T1
Blocked Tasks:      M3-E4-F2-T3  (waiting for ADR-0012 decision)
Pending Human Decisions: ADR-0012 queue strategy (Proposed)
Last Merged PR:     #48  M3-E5-F3-T1
Known Risks:        rate limit on legacy API (R-007)
Next Task:          M3-E5-F3-T3
```

DEFINITION OF DONE (THE BACK GATE)

Every criterion has a test that ran in CI · all required checks green · human approval (Technical Approver if High risk) · contracts, `.env.example` and docs updated · logs and metrics on new paths · migration reversible · report written · status Done · new debt logged.

DONE WHEN

- [ ] Task's pull request merged with all checks green
- [ ] Every acceptance criterion has a test that ran in CI
- [ ] Report written, tracker and task status updated
- [ ] Agent stopped

AVOID

- × Asking the AI to 'just build the feature'
- × Letting the agent commit to main or merge
- × Starting a second task before the first is merged

PROMPT 11.1 **First run: bootstrap a fresh project (use once)** COPY · PASTE

USE WHEN	Once, when EXECUTION_TRACKER.md is empty or absent and Phases 0 to 10 are signed off.
YOU GIVE	An agentic coding tool with access to the repository.
SAVES TO	`EXECUTION_TRACKER.md, the first task's branch, pull request and reports/<task-id>.md`
VERIFY	The tracker exists, one pull request is open for the first task, CI ran, and the agent stopped.

```
Begin execution for the first time.
ONE-TIME SETUP. Run only when EXECUTION_TRACKER.md is empty or
absent. AGENTS.md and AI_DEVELOPMENT_CONSTITUTION.md override
everything below.

1. Read the governing set:
   - FINAL_ARCHITECTURE_FREEZE.md
   - AI_DEVELOPMENT_CONSTITUTION.md
   - MASTER_IMPLEMENTATION_GUIDE.md
   - IMPLEMENTATION_ROADMAP_MVP.md
   - TASK_BREAKDOWN.md
   - ADR/README.md (the ADR index)
2. Confirm readiness:
   - Every governing document exists and records its sign-off
   - TASK_BREAKDOWN.md has uniquely-IDed tasks with dependencies
   - M0 Foundation is the first milestone
   - If anything is missing, STOP and list the gap
3. Create EXECUTION_TRACKER.md:
   - Current Milestone / Epic / Feature / Task
   - Completed Tasks: none     Blocked Tasks: none
   - Pending Human Decisions: none
   - Last Merged PR: none
   - Next Task: the first task (e.g. M0-E1-F1-T1)
4. Select the FIRST executable task: lowest Milestone, Epic,
   Feature, Task with status Ready and no unmet dependency.
5. Run the 10-point governance check (prompt 10.1). Any READY
   item NO, or BLOCKER YES: STOP and report.
6. If an ADR is needed: draft it with Status Proposed (prompt
   9.2) and STOP. Never mark an ADR Accepted.
7. Create branch task/<task-id> from main.
8. Write tests first: one or more per acceptance criterion,
   named with its ID. Run them and confirm they fail.
9. Implement exactly this one task, following
   MASTER_IMPLEMENTATION_GUIDE.md. No unrelated refactoring.
10. Run every check command in AGENTS.md (lint, typecheck,
    tests, build, security, dependency, docs, performance).
```

AI can build a demo in an afternoon. Production is a different promise.

This guidebook is the governed, step-by-step process for building software with AI coding agents (Claude Code, Cursor, Codex, Antigravity) that is tested, secure, deployable, maintainable and ready to hand to a client. AI drafts and implements. People decide on evidence.

- Your framework's 19 phases, in order, each with steps, prompts, files and a **Done when** checklist
- 40 copy-paste prompts, including first-run, single-task and continuous AI execution
- A complete project folder structure, plus a starter kit you can copy
- 9 quality gates and a 30-point production-readiness checklist scaled by risk
- Security, testing, CI/CD, monitoring, rollback, backups and client handover built in

No method can guarantee bug-free or perfectly secure software. This one makes the important work visible, ordered and checkable, and tells you exactly when an experienced engineer must sign off.

Authored by the Azrotic Team