AI IMPLEMENTATION FRAMEWORK

A practical planning framework for moving an AI opportunity from discovery to controlled deployment.

INSIDE THIS RESOURCE

- Purpose
- Phase 1: Discover
- Planning posture
- Phase 2: Scope

HOW TO USE THE FRAMEWORK

Purpose

Use this framework to organize the work before, during, and after an AI pilot. It is not a formal Butler Agency methodology or a promise about engagement length. It is a practical planning structure for deciding what to build, what to test, what to govern, and what to avoid.

- Start with one workflow, not a broad AI transformation theme.
- Define the human owner, the system boundary, and the data boundary before prototype work begins.
- Treat the prototype as a learning instrument, not as proof that the production system is solved.
- Move forward only when exit criteria are clear enough for the business owner, technical owner, and risk owner to understand.

Planning posture

KEEP THE FRAME NARROW ENOUGH TO TEST

A useful AI implementation starts with an operationally specific workflow: the source systems, review path, exception handling, expected output, and failure mode are all visible.

PHASE 1: DISCOVER + PHASE 2: SCOPE

Phase 1: Discover

- Activities: observe the current workflow, interview the people doing the work, identify source systems, document handoffs, and list where judgment or exception handling appears.
- Deliverables: workflow map, data inventory, stakeholder list, risk notes, and candidate use cases ranked by operational value and implementation friction.
- Exit criteria: the team can explain the workflow without resorting to generic AI language, and the first candidate use case has a clear business owner.
- Watch-outs: do not accept a polished demo as workflow evidence; verify how the work actually happens on a normal day.

Phase 2: Scope

- Activities: choose the smallest useful workflow segment, define in-scope and out-of-scope tasks, set success metrics, and decide what data can be used in testing.
- Deliverables: pilot brief, acceptance criteria, data boundary, review plan, and a no-go list that defines what would stop the pilot.
- Exit criteria: the team knows what will be built, what will not be built, who will review output, and how success will be measured.
- Watch-outs: avoid scoping around a model capability alone. Scope around an operational result.

PHASE 3: BUILD

Prototype the workflow layer

Build work should connect the use case to the systems and review paths that make it real. Depending on the workflow, that may mean a simple retrieval prototype, a copilot interface, API integration, automation around an existing system, or deeper infrastructure work. Specific patterns to consider include Kubernetes-based deployment, multi-agent coordination, GPU-backed workloads, and private model routing for confidential data.

- Activities: select the implementation pattern, connect representative data, define prompts or orchestration logic, build the first interface or workflow handoff, and instrument the system for review.
- Deliverables: working prototype, integration notes, test dataset, human-review path, risk assumptions, and implementation backlog.
- Exit criteria: the prototype can run against representative examples and produce outputs that users can evaluate in context.
- Watch-outs: do not confuse a model response with an implemented workflow. The build phase is where data, systems, permissions, review, and user experience meet.

Vertical software considerations

WHEN THE WORKFLOW TOUCHES VERTICAL SOFTWARE

Treat product architecture, permissions, auditability, and integration boundaries as first-order design inputs, not cleanup tasks after the prototype works.

PHASE 4

PHASE 4: VALIDATE

Test usefulness, not novelty

- Activities: run representative cases, compare outputs against human baseline, test edge cases, document failure modes, and collect user feedback from the people who will own the workflow.
- Deliverables: validation report, error taxonomy, revised acceptance criteria, risk register updates, and a recommendation to stop, revise, pilot, or deploy.
- Exit criteria: the team understands where the system performs well, where it fails, what human review is required, and what must change before production use.
- Watch-outs: avoid validation sets that only include easy examples. Useful testing includes messy records, ambiguous requests, missing information, and handoff failures.

Measurement examples

- Cycle time reduction for a repeated workflow.
- Error reduction or fewer manual lookups.
- Improved consistency in summaries, routing, or review packets.
- User adoption and reviewer confidence, not only model accuracy.

PHASE 5: DEPLOY + PHASE 6: SCALE

Phase 5: Deploy

- Activities: prepare production configuration, confirm access controls, publish user guidance, define support ownership, and roll out to a controlled user group.
- Deliverables: deployment checklist, runbook, training notes, monitoring plan, fallback path, and support escalation path.
- Exit criteria: users know when to use the system, reviewers know what to check, and operators know how to monitor and respond.
- Watch-outs: do not launch without a fallback path. If the AI layer is unavailable or produces low-confidence output, the workflow still needs a working path.

Phase 6: Scale

- Activities: review usage, tune the workflow, add adjacent use cases only after the first path is stable, and revisit data, security, and support assumptions.
- Deliverables: scale plan, backlog, governance notes, updated documentation, and owner-approved expansion criteria.
- Exit criteria: expansion is based on observed operating value rather than enthusiasm alone.
- Watch-outs: scaling too early turns a contained pilot into distributed process debt.

COMMON FAILURE MODES

Where AI implementation work breaks down

- Unclear workflow owner: no one can make scope, acceptance, or rollout decisions.
- Data boundary drift: sensitive or unreliable data enters the pilot without a clear handling rule.
- Demo-first planning: the prototype looks impressive but does not connect to the actual work.
- Review path gap: outputs are generated, but no person or process owns validation.
- Integration underestimation: APIs, permissions, data schemas, and upstream changes are treated as minor details.
- Pilot sprawl: one workflow expands into several before the first path has been validated.
- Operational handoff gap: the build works once, but support, monitoring, documentation, and training are not owned.

Practical countermeasure

USE EXIT CRITERIA AS THE CONTROL POINT

Each phase should end with a decision: stop, revise, continue, or expand. If the team cannot make that decision clearly, the phase is not done.

TIMELINE TEMPLATES: 6-WEEK + 12-WEEK

6-week pilot planning example

- Week 1: discover workflow, confirm owner, document data boundary, and select one use case.
- Week 2: scope pilot, define acceptance criteria, prepare representative test examples, and identify review path.
- Weeks 3-4: build prototype, connect limited data or system inputs, and prepare review instrumentation.
- Week 5: validate with users, run edge cases, document failure modes, and revise acceptance criteria.
- Week 6: decide whether to stop, revise, pilot in a controlled setting, or plan production deployment.

12-week implementation planning example

- Weeks 1-2: discovery, workflow mapping, data inventory, stakeholder alignment, and candidate prioritization.
- Weeks 3-4: scope, risk review, test design, integration planning, and pilot brief approval.
- Weeks 5-7: prototype build, workflow integration, human-review design, and instrumentation.
- Weeks 8-9: validation, edge-case testing, user feedback, and revision.
- Weeks 10-11: controlled deployment planning, training, runbook, monitoring, and fallback path.
- Week 12: deployment decision, scale criteria, and backlog for adjacent workflow opportunities.