Contents

Marketplace Agentic Architecture – Consolidated Overview

Authors: Joey Best-James (Senior System Architect), Micah Johnson (System Architect)
Started: January 15, 2026
Last Updated: January 19, 2026
Status: Discovery & Framework Development

> This is the **canonical document** for the Marketplace Agentic Architecture exploration. All other files in this directory are supporting detail.

Table of Contents

Ethos

"Tools recede, understanding remains."

Guiding Principles:

1. **Serve the humans in the loop** — Agents augment reviewers, not replace them
2. **Start internal, earn trust** — Shadow mode before autonomous decisions
3. **Version everything** — Agent configs, prompts, and logic are versioned like code
4. **Measure before optimizing** — Baseline accuracy before claiming improvement
5. **Fail gracefully** — When uncertain, escalate to humans

Team

Joey Best-James — Senior System Architect

- Led migration from Knack → Airtable (foundational system work)
- Airtable system design & automations (expert)
- Python development
- Built Partner/Experts matching system with versioned algorithms (v7)
- Guiding Micah through Webflow systems architecture
- Learning: Local agent development environments (Claude Code, Cursor)

Micah Johnson — System Architect

- Frontend systems (SvelteKit, Next.js, Designer Extensions)
- Cloudflare Workers architecture
- Agent SDK (Claude, Gemini Pro) + Modal for agent deployment
- IC MVP translation pipeline
- Built Response Classification agent (Zapier + GPT-5.1)
- Learning systems architecture from Joey

Goal

Primary Question: When do we use agents? When is it a function/rule/automation?

Target: - Identify where agents add value in the Marketplace system - Optimize existing agents (Response Classification) - Design what's next

What We Mean by "Agent"

An **agent** is: - LLM-powered reasoning that handles ambiguity - Operates with defined inputs/outputs - Has confidence scoring (knows when it's uncertain) - Versioned and auditable - Human-in-the-loop for high-stakes decisions

An agent is **NOT:** - A replacement for deterministic rules - Autonomous without oversight - A black box

---

Framework: When to Use What

Decision Matrix

Approach	Use When	Cost	Risk	Examples
Deterministic Rules	Logic is known, bounded, stable	Low	Low	File size limits, naming conventions
Weighted Algorithms	Multiple factors, tunable weights	Low	Low	Partner matching, priority scoring
Scripts/API Calls	Predictable I/O, no reasoning	Low	Low	Webhook delivery, data transforms
Automations	Event-driven, multi-step workflows	Low	Low	Status change → notification
AI Agents	Ambiguity, reasoning, judgment	Medium–High	Variable	Response classification, content analysis
Human Review	High stakes, edge cases, taste	High	Low	Final approval, rejection decisions

Decision Tree

```
Is the task deterministic with clear rules?
├── YES → Use rules/scripts/automations
└── NO → Does it require reasoning about ambiguous input?
    ├── NO → Use weighted algorithm (like Partner matching)
    └── YES → What are the stakes if wrong?
        ├── LOW → Agent can act autonomously
        ├── MEDIUM → Agent suggests, human confirms
        └── HIGH → Agent surfaces info, human decides
```

Current State

Volume (December 2025)

Metric	Value
Assets submitted	382
Templates	95%
Apps	5%
Published	37%
Pending/Rejected	63%
Stuck 5+ days	26 (7%)

Review Team

- 5–6 active reviewers
- Top 3 handle 71% of reviews
- Pablo handles Apps specifically

Existing Systems

System	Owner	Tech Stack	Status
Airtable Backend	Joey	Airtable + Automations	Production
Asset Dashboard	Micah	SvelteKit + Cloudflare	Production
App Form	Micah	Next.js + Vercel	Production
Template Validation	Micah	Designer Extension + CF Worker	Production
Bundle Scanner	Micah	Security rules	Experimental

Existing Agents

Agent	Purpose	Status	Owner
Response Classification	Reads creator responses, determines if status should change from “Changes Requested” to “Response to Review”	Zapier (paused)	Micah
Categorization agents	Running via Airtable automations	Production	Joey

Response Classification Agent Details **Platform:** Zapier
Model: GPT-5.1 (temperature: 0.7)
Zap Name: “Zendesk Response Sync”

Flow:

```
Airtable (Zendesk Messages updated)
    → Zendesk (get latest comment)
    → ChatGPT (classify email)
    → Filter (only "Ready for re-review")
    → Airtable (update Asset Version status to "□ Response to Review")
```

Classification Framework: Uses Heidegger’s phenomenological distinction: - **Zuhandenheit (Ready-to-hand):** Work is complete, creator’s concern orients toward review → “Ready for re-review” - **Vorhandenheit (Present-at-hand):** Work remains incomplete, creator still engaged with modifications → “Still working on it”

Current Status: Nodes 2-5 paused

Related Systems (Joey)

System	Owner	Status	Notes
Partner/Experts Matching	Joey	Production	Algorithmic matching with 17 variables (v7)
Algorithms Table	Joey	Production	v1-v7 tracked, pattern for agent config versioning

System	Owner	Status	Notes
Expert Matching Algorithm	Joey	Production	Matches partners with users based on provided details

Why these matter for the Agentic Layer: – Joey led the Knack → Airtable migration (deep knowledge of data model) – The Algorithms table pattern (versioned configs) directly applies to agent prompts – Expert matching demonstrates weighted variable approach → useful for smart routing – Joey's existing Python + Airtable integration patterns are the foundation

---

Agent Opportunities

Prioritized List

Priority	Opportunity	Current State	Proposed	Value
P1	Response Classification optimization	Working, accuracy unknown	Add confidence scoring + escalation	Medium
P1	Security pre-scan	Manual Bundle Scanner	Auto-run on all submissions	High
P1	Validation → Review correlation	No tracking	Learn what validation issues predict rejections	High
P2	Duplicate detection	None	Agent compares to existing assets	High
P2	Smart routing	Manual/round-robin	Route by asset type + reviewer expertise	High
P2	Consistency check	None	Surface similar past reviews	High
P3	Rejection email generation	Templates	Contextual draft generation	Medium
P3	Auto-fix suggestions	None	Validation app suggests fixes	Medium
P3	Review timeline prediction	None	Dashboard shows estimated time	Low

By Lifecycle Stage

```
SUBMISSION         QUEUE              REVIEW             FEEDBACK
    │                 │                  │                   │
    ▼                 ▼                  ▼                   ▼
┌─────────┐     ┌──────────┐      ┌───────────┐      ┌────────────┐
│Duplicate│     │  Smart   │      │ Security  │      │  Response  │
│Detection│     │ Routing  │      │ Pre-Scan  │      │ Classific. │
└─────────┘     └──────────┘      └───────────┘      └────────────┘

┌─────────┐     ┌──────────┐      ┌───────────┐      ┌────────────┐
│Pre-valid│     │ Priority │      │Consistency│      │ Rejection  │
│Results  │     │ Scoring  │      │   Check   │      │  Drafts    │
└─────────┘     └──────────┘      └───────────┘      └────────────┘
```

Research Insights

Source: Perplexity Deep Research, January 2026

Industry Benchmarks

Finding	Source
60% automation rate achievable with proper calibration	Shopify
Multi-layer architecture: automated → human → appeals	Etsy
Graduated enforcement: warnings before suspension	Etsy

Confidence Calibration (Critical Finding)

Raw model confidence is poorly calibrated. A model reporting 90% confidence may only be 70% accurate.

Solution: Multi-tier thresholds

Confidence	Action
>85%	Auto-approve
70-85%	Quick human confirmation
<70%	Deep review with reasoning

Category-specific thresholds: - High-consequence (fraud, policy): require >90% - Low-consequence (grammar, tone): can auto-approve at >75%

Prompt Versioning = Joey's Algorithms Pattern

Industry best practice **matches what Joey already built:** - Version prompts like code (v1, v2, v3…) - Store configs as data (not hardcoded) - A/B test versions before full rollout - Track which version processed which records

Airtable + LLM Integration

Recommended stack (aligns with Joey’s expertise):

Airtable (data) → Automation (trigger) → Python/pyAirtable → Claude API → Update record

---

System Visualization

Current State + Agentic Layer

```
┌──────────────────────────────────────────────────────────────────────────────────────────
│                                    CREATOR LAYER
├──────────────────────────────────────────────────────────────────────────────────────────
│
│   ┌──────────────────────┐    ┌──────────────────────┐    ┌──────────────────────┐     │
│   │   Asset Dashboard    │    │       App Form       │    │  Template Validation │     │
│   │   (SvelteKit/CF)     │    │   (Next.js/Vercel)   │    │  (Designer Extension)│     │
│   │                      │    │                      │    │                      │     │
│   │  □  View assets      │    │  □  Submit apps      │    │  □  Validate before  │     │
│   │  □  Review progress  │    │  □  File uploads     │    │     submission       │     │
│   │  □  Timeline predict │    │  □  Pre-validation   │    │  □  Auto-fix suggest │     │
│   └──────────┬───────────┘    └──────────┬───────────┘    └──────────┬───────────┘     │
│              │                           │                           │                 │
└──────────────┼───────────────────────────┼───────────────────────────┼─────────────────┘
               │                           │                           │
               ▼                           ▼                           ▼
┌──────────────────────────────────────────────────────────────────────────────────────────
│                                    DATA LAYER (Airtable)
├──────────────────────────────────────────────────────────────────────────────────────────
│
│   ┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐     │
│   │  □  Assets      │  │  □  Versions    │  │  □  Zendesk Msg │  │  □  Creators    │     │
│   │                 │  │                 │  │                 │  │                 │     │
│   │  Status, Type   │  │  Review state   │  │  Responses      │  │  History        │     │
│   └────────┬────────┘  └────────┬────────┘  └────────┬────────┘  └─────────────────┘     │
│            │                    │                    │                                   │
│            └────────────────────┼────────────────────┘                                   │
│                                 │                                                        │
│                    ┌────────────┴─────────────┐                                          │
│                    │  AUTOMATIONS (Triggers)  │                                          │
│                    │  □  Status changes       │                                          │
│                    │  □  New submissions      │                                          │
│                    │  □  Response received    │                                          │
│                    └────────────┬─────────────┘                                          │
│                                 │                                                        │
└─────────────────────────────────┼──────────────────────────────────────────────────────────
                                  │
                                  │ webhook / trigger
```

```
                         ▼
    ┌─────────────────────────────────────────────────────
    │                AGENTIC LAYER (Proposed)
    ├─────────────────────────────────────────────────────
    │
    │  ┌─────────────────────────────────────────────────┐
    │  │          AGENT SERVICE (Python/CF Worker)       │
    │  │                                                 │
    │  │  ┌─────────────┐ ┌─────────────┐ ┌─────────────┐│  │
    │  │  │  Response   │ │  Security   │ │   Smart     ││  │
    │  │  │Classification│ │  Pre-Scan   │ │  Routing    ││  │
    │  │  │             │ │             │ │             ││  │
    │  │  │□  IN ZAPIER │ │□  PLANNED   │ │□  PLANNED   ││  │
    │  │  │(to migrate) │ │             │ │             ││  │
    │  │  └──────┬──────┘ └──────┬──────┘ └──────┬──────┘│  │
    │  │         │               │               │       │  │
    │  │         └───────────────┼───────────────┘       │  │
    │  │                         │                       │  │
    │  │              ┌──────────┴──────────┐            │  │
    │  │              │     LLM CALLS       │            │  │
    │  │              │  Claude / GPT-5.1   │            │  │
    │  │              │                     │            │  │
    │  │              │ Input: context + prompt          │  │
    │  │              │ Output: decision +  │            │  │
    │  │              │         confidence +│            │  │
    │  │              │         reasoning   │            │  │
    │  │              └──────────┬──────────┘            │  │
    │  │                         │                       │  │
    │  └─────────────────────────┼───────────────────────┘  │
    │                            │                          │
    │  ┌─────────────────────────┼───────────────────────┐  │
    │  │           AGENT CONFIGS (Airtable)              │  │
    │  │                         │                       │  │
    │  │  ┌─────────────┐ ┌──────┴──────┐ ┌─────────────┐│  │
    │  │  │ □  Prompts  │ │ □ Thresholds│ │ □ Agent Logs││  │
    │  │  │ (versioned) │ │ (confidence)│ │ (decisions) ││  │
    │  │  │             │ │             │ │             ││  │
    │  │  │ v1, v2, v3...│ │ high: 0.85  │ │ input, output││  │
    │  │  │ □  PLANNED  │ │ low:  0.70  │ │ accuracy    ││  │
    │  │  │ (like Joey's│ │ □  PLANNED  │ │ □  PLANNED  ││  │
    │  │  │  Algorithms)│ │             │ │             ││  │
    │  │  └─────────────┘ └─────────────┘ └─────────────┘│  │
    │  │                                                 │  │
    │  └─────────────────────────────────────────────────┘  │
    │
    └─────────────────────────────────────────────────────
                         │
```

```
                            │ pyAirtable writeback
                            ▼
┌──────────────────────────────────────────────────────────────
│                      REVIEW LAYER
├──────────────────────────────────────────────────────────────
│
│  ┌──────────────────────────────────────────────────────┐
│  │            Airtable Interface (Reviewers)            │
│  │                                                      │
│  │  ┌───────────┐  ┌───────────┐  ┌───────────┐        │  │
│  │  │ □  Review │  │ □  Agent  │  │ □  Feedback│       │  │
│  │  │    Queue  │  │ Suggestions│ │ → Zendesk │        │  │
│  │  │           │  │           │  │           │        │  │
│  │  │ □  Exists │  │ □  PLANNED│  │ □  Exists │        │  │
│  │  └───────────┘  └───────────┘  └───────────┘        │  │
│  │                                                      │
│  └──────────────────────────────────────────────────────┘
│
└──────────────────────────────────────────────────────────────
```

```
LEGEND:  □  Exists    □  Exists (needs migration)    □  Planned
```

Data Flow: Response Classification (Current → Proposed)

```
CURRENT (Zapier):
┌─────────┐    ┌─────────┐    ┌─────────┐    ┌─────────┐    ┌─────────┐
│ Airtable│───▶│ Zapier  │───▶│ Zendesk │───▶│ GPT-5.1 │───▶│ Airtable│
│ trigger │    │ ($$)    │    │  lookup │    │ classify│    │  update │
└─────────┘    └─────────┘    └─────────┘    └─────────┘    └─────────┘

PROPOSED (Repo-based):
┌─────────┐    ┌─────────┐    ┌─────────┐    ┌─────────┐
│ Airtable│───▶│  Python │───▶│  Claude │───▶│ Airtable│
│ webhook │    │ Service │    │  API    │    │  update │
│         │    │ (in repo)│   │         │    │(pyAirtable)
└─────────┘    └─────────┘    └─────────┘    └─────────┘
                    │
                    ▼
               ┌─────────┐
               │  Agent  │
               │  Logs   │
               │ (audit) │
               └─────────┘
```

Progress Tracker

```
PHASE 1: Foundation                 PHASE 2: Optimize           PHASE 3: Expand
───────────────────                 ───────────────             ───────────────
[████████░░░░░░░░░░░░] 40%          [░░░░░░░░░░░░░░░░░░░░] 0%   [░░░░░░░░░░░░░░░░░░
```

☐ System context documented
☐ Agent audit complete
☐ Volume analysis done
☐ Research complete
○ Agent Configs table
○ Agent Logs table
○ Migrate Response Classification

○ Confidence scoring
○ Multi-tier thresholds
○ Escalation paths
○ Accuracy baseline
○ Human override tracking

○ Security pre-scan
○ Duplicate detecti
○ Smart routing
○ Consistency check

Tooling Discussion: Where Should Logic Live?

From FigJam (Joey's notes): "Shifting existing scripting / zaps / etc. to agentic approaches"
From meeting: "Potential GitHub repo setup (private or personal repos as a starting point)"

Current State

Tool	Used For	Limitation
Zapier	Response Classification agent	Expensive at scale
Airtable Automations	Categorization agents, workflows	Count limits
Cloudflare Workers	Dashboard, Validation app	Requires deployment

The Question

Should agent logic live in: 1. **UI-based tools** (Zapier, Airtable Automations) – faster iteration, no deployment 2. **Repos** (Python scripts, Cloudflare Workers) – versioned, testable, cheaper at scale

Tradeoff Analysis

Dimension	UI Tools (Zapier/Airtable)	Repo-Based (Python/CF Workers)
Iteration speed	Fast (no deploy)	Medium (deploy required)
Cost at scale	High (Zapier pricing)	Low (compute only)
Versioning	Limited	Full Git history
Testing	Manual	Automated
Observability	Platform logs	Custom (but flexible)
Team access	Easy (UI)	Requires Git knowledge
Reliability	Platform-dependent	Self-managed

Joey's Direction (from FigJam)

Joey already identified **shifting from Zapier/scripts to agentic approaches** as a direction. This aligns with codifying logic in repos because: – Agents need versioned prompts (Git provides this) – Agents need testing (repos enable CI/CD) – Agents need observability (custom dashboards)

Proposed Path

Hybrid approach:

1. **Airtable remains the data layer** – Joey's expertise, existing workflows
2. **Airtable Automations handle triggers** – detect events, call external services
3. **Python service (repo) handles intelligence** – LLM calls, confidence scoring, routing logic
4. **Results write back to Airtable** – via pyAirtable

```
Airtable (data + trigger)
    → Webhook to Python service (in repo)
    → Claude/GPT API call
    → pyAirtable writes result back
```

Benefits: - Airtable automation count used minimally (just triggers) - Zapier eliminated (cost savings) - Logic in repo (versioned, testable) - Joey can work in Python (his expertise) - Micah can host on Cloudflare Workers or Modal (his expertise)

Hosting Options for Python Agents

Platform	Pros	Cons	Best For
Modal	Sub-second cold starts, Python-native, scale-to-zero, no YAML	Newer platform	AI workloads, agents
Cloudflare Workers	Fast, global, existing expertise	Python support limited	Lightweight, edge
Render	Simple, Git-deploy	Cold starts	Background jobs
Vercel	Existing (App Form)	Timeout limits	Web-facing

Micah's current approach: Using Modal with Claude Agent SDK and Gemini Pro for composable agent development.

Decision for Joey

Option	Pros	Cons
A: Keep Zapier	No migration work	Expensive, not versioned
B: Move to Airtable Scripts	All in one platform	Automation count limits, less flexible
C: Move to Python repo	Versioned, cheap, testable	Requires deployment setup
D: Hybrid (Airtable trigger → Repo logic)	Best of both	Initial setup complexity

Recommendation: Option D (Hybrid) aligns with Joey's stated direction and both team members' expertise.

Design Philosophy: Composable Agents

Core Principles

1. **Composability over monoliths** – Agents are modular, pluggable into specific system areas
2. **Migration is expected** – Build with refactoring in mind, not permanent architecture
3. **IC/DE empowerment** – Enable individual contributors to create agentic items
4. **Managed pipeline** – Agentic creations flow through validation before production
5. **Find the fit** – Identify where agents serve best vs. where rules/automation suffice

The Composable Agent Model

```
┌──────────────────────────────────────────────────────────────────────────────────
│                              COMPOSABLE AGENT ARCHITECTURE
├──────────────────────────────────────────────────────────────────────────────────
│
│   ┌─────────────────────────────────────────────────────────────────────────┐
│   │                            AGENT REGISTRY                               │
│   │                                                                         │
│   │   ┌──────────────┐  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐ │   │
│   │   │  Response    │  │  Security    │  │   Routing    │  │ [Your Agent] │ │   │
│   │   │  Classifier  │  │  Scanner     │  │   Agent      │  │              │ │   │
│   │   │              │  │              │  │              │  │  IC/DE can   │ │   │
│   │   │  v2.1        │  │  v1.0        │  │ v0.1 (draft) │  │  create new  │ │   │
│   │   └──────┬───────┘  └──────┬───────┘  └──────┬───────┘  └──────────────┘ │   │
│   │          │                 │                 │                           │   │
│   └──────────┼─────────────────┼─────────────────┼───────────────────────────┘   │
│              │                 │                 │
│              ▼                 ▼                 ▼
│   ┌─────────────────────────────────────────────────────────────────────────┐
│   │                  AGENT RUNTIME (Modal / CF Workers)                     │
│   │                                                                         │
│   │   • Claude Agent SDK                     • Scale to zero when idle      │
│   │   • Gemini Pro                           • Sub-second cold starts       │
│   │   • Versioned prompts from repo          • Logs to Agent Logs table     │
│   │   • pyAirtable for data access           • Confidence scoring           │
│   │                                                                         │
│   └─────────────────────────────────────────────────────────────────────────┘
│
└──────────────────────────────────────────────────────────────────────────────────
```

IC/DE Agent Pipeline

ICs and DEs can create agents that improve the system through a managed pipeline:

```
┌─────────────┐    ┌─────────────┐    ┌─────────────┐    ┌─────────────┐    ┌─────────────┐
│  IC/DE      │───▶│  Shadow     │───▶│  Review     │───▶│  Canary     │───▶│ Production  │
│  Creates    │    │  Mode       │    │  & Approve  │    │  (10%)      │    │  (100%)     │
│  Agent      │    │  (no act)   │    │  (Joey)     │    │             │    │             │
└─────────────┘    └─────────────┘    └─────────────┘    └─────────────┘    └─────────────┘
                          │                                     │
                          ▼                                     ▼
                   ┌─────────────┐                       ┌─────────────┐
                   │  Accuracy   │                       │  Monitor    │
                   │  Baseline   │                       │  & Alert    │
                   └─────────────┘                       └─────────────┘
```

Example: A DE creates a “Duplicate Detection” agent using AI Studio: 1. Agent runs in shadow mode (logs decisions but doesn’t act) 2. After 2 weeks, accuracy measured against human decisions 3. Joey reviews, approves promotion 4. Canary deployment to 10% of submissions 5. If metrics hold, full production

Agent-Assisted Development Workflow

Joey and Micah will work **with** agents to build **the** agents. This meta-layer is part of the composable philosophy.

```
┌──────────────────────────────────────────────────────────────────────────────────────────
│                             DEVELOPMENT ENVIRONMENT
├──────────────────────────────────────────────────────────────────────────────────────────
│
│    ┌────────────────────────────────────────────────────────────────────────────────┐
│    │                           LOCAL DEVELOPMENT                                      │
│    │                                                                                  │
│    │    ┌────────────────┐   ┌────────────────┐   ┌────────────────┐               │   │
│    │    │  Claude Code   │   │     Cursor     │   │   Gemini CLI   │               │   │
│    │    │  (Terminal)    │   │     (IDE)      │   │   (Terminal)   │               │   │
│    │    │                │   │                │   │                │               │   │
│    │    │  • bd (beads)  │   │  • MCP tools   │   │  • Research    │               │   │
│    │    │  • Git ops     │   │  • Perplexity  │   │  • Code gen    │               │   │
│    │    │  • Code review │   │  • Browser     │   │  • Analysis    │               │   │
│    │    └────────────────┘   └────────────────┘   └────────────────┘               │   │
│    │                                                                                  │
│    │    ┌──────────────────────────────────────────────────────────────────────────┐ │
│    │    │                      SHARED CONTEXT                                      │ │
│    │    │                                                                          │ │
│    │    │   specs/webflow-marketplace/    ← This directory                         │ │
│    │    │   packages/agent-sdk/           ← Agent patterns                         │ │
│    │    │   .claude/ settings             ← Agent configuration                    │ │
│    │    │   AGENTS.md                     ← Workflow principles                    │ │
│    │    │                                                                          │ │
│    │    └──────────────────────────────────────────────────────────────────────────┘ │
│    │                                                                                  │
│   │                                                                   │
│   └───────────────────────────────────────────────────────────────────┘
│
│
│   ┌───────────────────────────────────────────────────────────────────┐
│   │                       JOEY'S LEARNING PATH                        │
│   │                                                                   │
│   │  Current:                          Building toward:              │   │
│   │  • Airtable automations            • Claude Code for agent development │   │
│   │  • Python scripts                  • Local testing with Modal    │   │
│   │  • Zapier (transitioning)          • Git-based prompt versioning │   │
│   │                                    • Cursor + MCP for research   │   │
│   │                                                                   │
│   └───────────────────────────────────────────────────────────────────┘
│
└──────────────────────────────────────────────────────────────────────────
```

Composable Local Experiences:

Tool	Role in Workflow
Claude Code	Terminal-based agent for code, git, system tasks
Cursor	IDE with MCP integrations (Perplexity, browser)
Gemini CLI	Research, analysis, alternative perspectives
beads (bd)	Agent-native issue tracking
Modal	Local → cloud deployment with same code

Why this matters: - Agents help build agents (meta-productivity) - Shared repo context keeps everyone aligned - Joey learns tools incrementally while staying productive - Same patterns used for development AND production agents

---

Why Modal Fits

Modal aligns with this composable philosophy:

Feature	Benefit
Python-native	Joey can write agents directly
Decorator-based	@app.function() - no YAML/config
Scale to zero	Only pay when agents run
Sub-second cold starts	Fast response times
Code = infrastructure	Versioned in Git
$30/mo free tier	Low barrier to experiment

```python
# Example: Composable agent on Modal
import modal

app = modal.App("marketplace-agents")

@app.function()
def classify_response(response_text: str, config_version: str) -> dict:
    """Classify creator response using versioned prompt."""
    prompt = load_prompt(config_version)  # From repo or Airtable
    result = claude.classify(prompt, response_text)
    return {
        "intent": result.intent,
        "confidence": result.confidence,
        "reasoning": result.reasoning
    }
```

Architecture Approach

Technology Stack (Confirmed)

```
┌──────────────────────────────────────────────────────────────────────────────────────
│                                   STACK OVERVIEW
├──────────────────────────────────────────────────────────────────────────────────────
│
│   LAYER              TECHNOLOGY              WHY
│   ─────              ──────────              ───
│
│   Database           Airtable                Internal teams can access without
│                                              SQL/Snowflake queries
│
│   Infrastructure     Cloudflare              Workers, D1, R2, Queues
│                      (automations/           Global edge, existing expertise
│                       workflows)
│
│   Agent Runtime      Modal + CF Workers      Python agents (Modal), edge logic (CF)    │
│
│   Frontend           SvelteKit / Next.js     Dashboard (Svelte), Forms (Next)          │
│                      on CF / Vercel
│
│   Code               Git repos               Versioned, reviewable, CI/CD              │
│                      (CREATE SOMETHING)
│
│   Agent Configs      Airtable tables         Versioned prompts, thresholds             │
│                      (like Algorithms)       (internal team can view/edit)             │
│
└──────────────────────────────────────────────────────────────────────────────────────
```

Why Airtable Stays as Database

Alternative	Problem
Snowflake	Internal teams need SQL skills to query
Postgres	No visual interface for reviewers
Firebase	Different paradigm, migration cost

Airtable benefits: – Reviewers use Airtable Interface (no code) – Automations trigger workflows – API access for external systems (pyAirtable) – Internal teams self-serve without engineering

System Map (What Exists + What's Planned)

```
┌──────────────────────────────────────────────────────────────────────────────────
│
│  FRONTENDS (SvelteKit / Next.js)
│  ───────────────────────────────────────
│                                                                                  │
│   ┌─────────────────┐   ┌─────────────────┐   ┌─────────────────┐                │
│   │ Asset Dashboard │   │    App Form     │   │ Template Valid. │                │
│   │ (SvelteKit/CF)  │   │(Next.js/Vercel)│   │ (Designer Ext)  │                │
│   │                 │   │                 │   │                 │                │
│   │ □  Exists        │   │ □  Exists        │   │ □  Exists        │                │
│   └────────┬────────┘   └────────┬────────┘   └────────┬────────┘                │
│            │                     │                     │
│            └─────────────────────┼─────────────────────┘
│                                  │
│                                  ▼
│  DATABASE (Airtable)
│  ──────────────────────
│
│   ┌──────────────────────────────────────────────────────────────────────────────┐
│   │  MARKETPLACE                                                                 │
│   │  Assets │ Versions │ Zendesk Msg │ Creators │ Agent Configs │ Agent Logs     │
│   │   □      │    □      │     □         │    □      │     □            │     □            │
│   ├──────────────────────────────────────────────────────────────────────────────┤
│   │  PARTNER/EXPERTS (Joey)                                                      │
│   │  Experts │ Partners │ Algorithms (v7) │ Expert Matching                      │
│   │    □      │    □      │       □           │      □                               │
│   └──────────────────────────────────────────────────────────────────────────────┘
│                                  │
│                                  │ Automations trigger
│                                  ▼
│  INFRASTRUCTURE (Cloudflare)
│  ─────────────────────────────────────
│                                                                                  │
│   ┌─────────────────┐   ┌─────────────────┐   ┌─────────────────┐                │
│   │     Workers     │   │       D1        │   │       R2        │                │
│   │ (API, webhooks) │   │   (meetings,    │   │    (files,      │                │
│   │                 │   │    local data)  │   │     assets)     │                │
│   │ □  Dashboard     │   │ □  Meetings      │   │ □  Meetings      │                │
│   │ □  Validation    │   │                 │   │                 │                │
│   │ □  Agent APIs    │   │                 │   │                 │                │
│   └─────────────────┘   └─────────────────┘   └─────────────────┘                │
│                                  │
│                                  ▼
│  AGENT RUNTIME (Modal / Cloudflare Workers)
│  ─────────────────────────────────────────────────
│
```

```
│
│   ┌──────────────────────────────────────────────────────────────────────────────┐
│   │  Response Classifier │ Security Scanner │ Routing Agent │ Future Agents...    │
│   │        □             │       □          │      □        │                     │ │
│   └──────────────────────────────────────────────────────────────────────────────┘
│
│
└──────────────────────────────────────────────────────────────────────────────────────

LEGEND:  □  Exists    □  Exists (needs migration)    □  Planned
```

Two Complementary Paths

Path 1: Airtable-Native (Joey's Domain)

```
┌──────────────────────────────────────────────────────────────┐
│                        AIRTABLE TABLES                       │
├──────────────────────────────────────────────────────────────┤
│  Agent Configs          │  Agent Logs            │  Assets   │
│  (versioned prompts,    │  (decisions, inputs,   │  (existing│
│   like Algorithms)      │   outputs, accuracy)   │   workflow)│
└─────────────────────────┴────────────────────────┴───────────┘
                    │
                    ▼
┌──────────────────────────────────────────────────────────────┐
│                    AIRTABLE AUTOMATIONS                      │
│  Trigger: Record created/updated                             │
│  Action: Script block OR webhook to Python service           │
│  Result: Update record with agent output                     │
└──────────────────────────────────────────────────────────────┘
```

Path 2: Cloudflare Worker (Micah's Domain)

For heavier processing (Bundle Scanner, validation correlation):

```
POST /api/agent/classify-response
POST /api/agent/pre-scan
POST /api/agent/route-asset

→ Uses Agent SDK patterns
→ Returns { decision, confidence, reasoning }
```

Recommended: Agent Configs Table

Modeled on Joey's Algorithms table:

Field	Type	Example
agent_name	Single line	response_classification
version	Number	7
status	Single select	draft / staging / production / deprecated
system_prompt	Long text	(the prompt)
model	Single select	claude-3-sonnet
temperature	Number	0.3
confidence_threshold_high	Number	0.85

Field	Type	Example
confidence_threshold_low	Number	0.70
created_at	Date	2026-01-19
created_by	Collaborator	Joey
notes	Long text	Added edge case handling for…
active	Checkbox	✓

Implementation Plan

Phase 1: Foundation (Weeks 1-2)

- ☐ Document Response Classification agent current logic
- ☐ Get accuracy baseline for existing agent
- ☐ Create Agent Configs table (modeled on Algorithms)
- ☐ Create Agent Logs table for decision tracking

Phase 2: Optimize Existing (Weeks 3-4)

- ☐ Add confidence scoring to Response Classification
- ☐ Implement multi-tier escalation (auto/quick/deep)
- ☐ Track human overrides for learning
- ☐ Measure accuracy improvement

Phase 3: Add High-Value Agents (Weeks 5-8)

- ☐ Security pre-scan automation (Bundle Scanner integration)
- ☐ Validation → Review correlation tracking
- ☐ Smart routing prototype

Phase 4: Scale & Iterate (Ongoing)

- ☐ A/B test agent versions
- ☐ Expand to P2/P3 opportunities
- ☐ Build agent performance dashboard

Action Items

From FigJam (Original)

Item	Status
Migrate to agent-first mechanism for managing this project	▯ Done (this spec structure)
Figure out cadence for advancing this project	▯ Pending
Catalog call recordings	▯ Accessed Jan 15 meeting
Express framework as table with cost dimensions	▯ Done

Item	Status
Review and consolidate	▯ This document

Immediate (This Week)

- ☐ Joey: Share Response Classification agent accuracy metrics
- ☐ Joey: Document current agent prompt/logic
- ☐ Micah: Set up correlation tracking (validation → review outcome)
- ☐ Both: Define cadence (weekly sync? async updates?)

Short-term (Next 2 Weeks)

- ☐ Design Agent Configs table schema in Airtable
- ☐ Design Agent Logs table schema
- ☐ Implement confidence scoring in Response Classification
- ☐ Establish shadow mode infrastructure

Questions for Joey

1. Are there other agents running (in Airtable automations) that we haven't documented?
2. What's your preference for cadence on this work?
3. **Tooling decision:** Should we pursue hybrid approach (Airtable trigger → Python repo → pyAirtable writeback)?
4. If yes to #3, where should the Python service be hosted? (Cloudflare Workers? Render? Vercel?)
5. Should we migrate Response Classification from Zapier as the first test case?

Success Metrics

Metric	Baseline	Target	Timeframe
Response Classification accuracy	TBD	95%+	4 weeks
Review turnaround (median)	~2-3 days	< 1 day	8 weeks
5+ day backlog	26 assets	< 5 assets	8 weeks
Automation rate	TBD	60%	12 weeks

File Index

This directory contains supporting detail:

File	Purpose
OVERVIEW.md	**This file** — canonical consolidated view
PLAN.md	Detailed implementation plan with phases
agentic-architecture.md	Framework deep-dive
agent-audit.md	Existing agents + opportunities detail

File	Purpose
system-context.md	System map and integration points
volume-analysis.md	Asset submission data analysis
research-findings.md	Perplexity research (Shopify, Etsy, calibration)
use-cases.md	Detailed use case evaluations
prd.json	Implementation stories (PRD format)

References

- Meeting transcript: January 15, 2026 (Joey + Micah)
- FigJam canvas: Marketplace Agentic Architecture Exploration
- Partner/Experts Algorithms table (v7 pattern)
- Perplexity Deep Research (January 2026)
- packages/agent-sdk/ — Agent development patterns

Last consolidated: January 19, 2026