PORTFOLIO CASE STUDY

# Multi-Agent Product Viability System

*AI-Orchestrated Business Analysis — Building, Stress-Testing, and Rebuilding an Eight-Agent System*

| 6 Hours | $4.11 | 8 Agents | $15K–$25K |
|---|---|---|---|
| Total Active Time | Total API Cost | Specialized Agents | Consulting Equivalent |

## Section 1: What Was Built

A fully functional multi-agent AI system that autonomously evaluates product viability — taking a product concept as input and producing a comprehensive, executive-ready analysis as output. The system uses eight specialized AI agents orchestrated through CrewAI, each powered by Claude Sonnet via the Anthropic API, running on a local Windows machine with Python 3.12.

Each agent runs in sequence, receiving prior agents' outputs as context before producing its own. A dedicated QA/Validation agent reconciles every upstream output — enforcing consistency and a hard feasibility gate — before the final agent, functioning as a VP of Product Strategy, synthesizes the analysis into a single unified viability report. Verified market data is injected from a separate, source-linked module (market_data.py) so the agents reason on top of real, cited figures rather than inventing them. Output is saved as nine markdown files: eight per-agent outputs plus the unified final report.

The system was stress-tested against a real prototype — the Follow Me Wagon autonomous cargo wagon — and its output was critiqued against executive-presentation standards across multiple evaluation-and-rebuild cycles. Each cycle surfaced and corrected a class of defect, culminating in one that was not mechanical but analytical: the system had confidently recommended a market the prototype's hardware could not physically serve. The final validated analysis was produced as a polished, presentation-ready report document.

| Agent | Role | Key Output |
|---|---|---|
| Agent 1 | Market Intelligence Analyst | Competitive landscape; source-linked market sizing by segment; confidence ratings |
| Agent 2 | Customer Analysis Specialist | Buyer personas; willingness-to-pay; feasibility-gated beachhead recommendation |
| Agent 3 | Hardware Documentation Specialist | Bill of materials; staged v1 → v2-dev → production hardware path; payload reality |
| Agent 4 | Financial Modeling Analyst | Staged cost model; unit economics; break-even; investment memo |
| Agent 5 | Go-to-Market Strategist | Positioning, channel copy, launch timeline — with no unverified performance claims |
| Agent 6 | Risk Assessment Analyst | IP, legal, competitive, technical, financial, and cybersecurity risk matrix |
| Agent 7 | QA / Validation Analyst | Cross-agent reconciliation; enforces the feasibility gate and beachhead consistency |
| Agent 8 | VP of Product Strategy (Orchestrator) | Unified viability report synthesizing all prior outputs into one recommendation |

## Core Capability

A non-developer with minimal prior Python experience built an eight-agent AI system from scratch — then evaluated its output with enough rigor to catch not just mechanical defects but a domain-level error (a recommended market the hardware could not physically serve), and rebuilt the system across multiple cycles until the analysis was honest, grounded, and decision-ready.

## Section 2: Why It Was Built

The project was driven by a direct question: can a complete non-developer build a meaningful, production-relevant multi-agent AI system — without any prior experience in Python, CrewAI, APIs, or agentic frameworks — and then hold its output to the standard a CTO or VP of Product would actually accept?

The harder, more revealing question came second: when the system produces polished, confident, complete-looking output that is wrong in a way only domain judgment can catch, will you catch it — and can you rebuild the system so it stops being wrong? That second question turned out to be the real test, and the heart of this case study.

> **Core Question**
>
> "Can a non-developer build a functional multi-agent AI system from scratch — and then catch the confident, well-written answer that is wrong, supply the constraint the model never had, and rebuild until the output meets the standard a CTO would accept?"

## Section 3: Starting Competence

| | |
|---|---|
| Prior Python Experience | Effectively zero — no prior Python code written before this work |
| Prior CrewAI / API Use | None — first time using the CrewAI framework or making direct Anthropic API calls |
| Prior Agentic AI Experience | Minimal — growing familiarity with AI tools from prior portfolio projects |
| Prior Programming Background | Minimal — Minor software development training from 15+ years ago |
| Environment Setup | First time configuring a Python environment, .env files, and dependency installation on Windows |
| AI Tool Fluency | Growing — first time building a system that itself runs AI |
| Skills Transferred In | Structured planning, disciplined scoping, precision questioning, and the instinct to verify before trusting |

## Section 4: How AI Was Used

### System Design & Build

The build was conducted in real-time collaboration with Claude. Starting from zero — no Python environment, no familiarity with CrewAI — the work covered environment setup, package installation, agent design, task-description writing, the context-passing architecture, and execution debugging. The resulting crew.py defined eight agents with distinct roles and tasks, a sequential execution chain passing each output

forward as context, a verified market-data module injected as ground truth, and a validation gate ahead of final synthesis.

### Stress-Testing and the First Rebuild

The first working system was critiqued against executive standards, and eight mechanical defects were named precisely: output truncation, no validation loop, unvalidated financial assumptions stated as fact, a go-to-market agent writing false performance specs, market figures with no citations or confidence flags, missing hardware tables, an incomplete synthesis, and no completeness check before synthesis. A structured rebuild specification was produced and executed — the critique itself a deliverable as rigorous as the build.

### The Deeper Defect: A Market the Hardware Couldn't Serve

Even after the mechanical fixes, a subtler failure persisted. The customer-analysis agent ranked the beachhead by willingness-to-pay with no feasibility constraint — so it recommended golf, the highest-paying segment, for a chassis rated to carry a fraction of a golf bag's weight. Every downstream agent then built financials, marketing, and risk analysis around a product that could not physically exist. And as prose quality improved across versions, the system grew *more* confident in the wrong answer — a polished, decisive "PROCEED" on a commercially impossible product. Better writing made the error more dangerous, not less.

The fix was not another prompt tweak. It required injecting the missing real-world constraint — spec-anchored payload capacity — as a hard feasibility gate applied *before* any willingness-to-pay ranking. A category error in the financial model (a single blended "production cost" that conflated a hobby chassis with a load-carrying machine) was retired and replaced with a three-stage cost model. A separate, source-linked market-data module replaced agent-fabricated figures. And a framework-specific bug was found: the token limit had been set on the agent object, where CrewAI silently ignores it, so every agent was capped and one truncated mid-sentence — losing its recommendation entirely. Moving the setting to the LLM object fixed it, and a completeness check with a mid-sentence detector was added to catch any recurrence.

Properly constrained, the system then did something genuinely useful: it redirected the beachhead. Set out to serve the originally intended personal-cargo market, the data pointed instead to small-farm harvest assist — a lower-payload, ROI-driven buyer with an uncontested price tier and no direct incumbent — and the final report told that redirection honestly, as a finding rather than a foregone conclusion.

> **Key Workflow Insight**
>
> The architecture is the leverage. Eight single-discipline agents, each reading the accumulated context of all prior agents, with a dedicated validation gate reconciling them before synthesis, produce cross-disciplinary reasoning that a single prompt cannot.
>
> But the most valuable skill demonstrated was not building the system — it was catching the answer the system was most confident about and most wrong about. A feasibility constraint the agents never had is the entire difference between a polished wrong answer and a correct one.

### Final Report Production

The validated analysis was produced as a polished, presentation-ready report — cover page, source-linked market figures with confidence ratings, staged financial tables, an honest engineering-gap assessment, and a clear conditional recommendation — suitable for executive review.

### Deferred by Deliberate Scope

Two production-grade improvements were identified and consciously deferred — documented rather than hidden. First, wiring a live web-search tool (CrewAI + SerperDev) into the agents so they retrieve and cite their own sources at run time, deferred because it requires a separate API key not provisioned at this stage;

verified data was injected as ground truth instead. Second, per-section length governance on the final synthesis — the token budget controls total output but does not yet enforce a hard limit on the executive summary specifically. Both are named as the next refinements for a real-company build.

## Section 5: Timeline & Conditions

| | |
|---|---|
| Total Active Time | ~6 hours across the full build, evaluation, and rebuild arc |
| Total API Cost | $4.11 (Anthropic usage) |
| Process | Initial build → structured critique → multiple evaluation-and-rebuild cycles → final report |
| Work Conditions | Part-time, weekend work alongside other commitments |
| External Deadline | None — self-directed, self-imposed success criteria |
| Platform | Python 3.12, CrewAI, Anthropic API, local Windows environment |

## Section 6: Outcome & Evidence

The delivered system is a functional eight-agent pipeline that takes a product concept as input and autonomously produces a comprehensive viability analysis — competitive landscape, feasibility-gated customer segmentation, a staged financial model, go-to-market strategy, risk and cybersecurity assessment, and a unified investment recommendation — grounded in source-linked market data, in a single execution run costing cents.

Total cost across all build and rebuild cycles was $4.11 in API usage. The comparable cost of commissioning equivalent work from a boutique strategy firm is $15,000–$25,000 for a project of this scope (2026 consulting-rate benchmarks). Against the $4.11 total, that is a cost compression on the order of **1,900×–3,100× (≈2,500× at the midpoint)** — the analytical depth delivered in hours rather than weeks.

The output was never accepted at face value. It was stress-tested, critiqued, and rebuilt across multiple cycles — including the diagnosis and correction of a domain-level error that polished output had concealed. The final produced report file was a solid 85% deliverable: structured, source-grounded, honest about the prototype's limits, and decision-ready. However, it still requires a discerning eye and good judgement. The AI output needed refinement for readability, as well as checking the cited sources to ensure the data was real and not just made up. Various hardware components and other acronyms needed to be spelled out to ensure readability by the target audience.

## Section 7: Key Metrics

| | |
|---|---|
| Total Active Time | ~6 hours — full build, evaluation, and rebuild arc |
| Total API Cost | $4.11 (Anthropic usage) |
| Specialized Agents | 8 — sequential, each passing output as context, with a dedicated validation gate |
| Output Documents | 9 — eight agent files plus the unified final report |
| Verified Data | Source-linked market_data.py module injected as ground truth |

| | |
|---|---|
| Consulting Equivalent | $15,000–$25,000 (2026 benchmarks for equivalent scope) |
| Cost Compression | ~1,900×–3,100× (≈2,500× midpoint) vs the $4.11 total |
| Primary AI Platform | Claude Sonnet via Anthropic API (all eight agents) |
| Orchestration Framework | CrewAI (Python 3.12, local Windows environment) |
| System Evolution | Multiple rebuild cycles: feasibility gate, staged costs, verified data, truncation fix, beachhead redirection |

## Section 8: Lessons Learned

### Lesson 1: Architecture Is the Leverage

A single agent asked to produce market analysis, financial model, and risk assessment at once produces mediocre work in all three. Eight specialized agents, each focused on one discipline, each receiving the accumulated context of all prior agents, with a validation gate reconciling them before synthesis, produce dramatically more rigorous output. Getting the roles, task descriptions, context chain, and validation step right before writing code is where the real design work happens.

### Lesson 2: The Hard Part Is Catching the Confident Wrong Answer

A system can be fluent, complete, and decisive — and wrong in a way only domain judgment catches. This one confidently recommended a market the hardware could not physically serve, and as the writing improved, its confidence in the wrong answer grew. Polish amplifies error. The defining skill is supplying the real-world constraint the model never had — here, payload capacity as a feasibility gate — and refusing to be persuaded by a well-written conclusion.

### Lesson 3: Constrain by Physical Reality Before Optimizing for Value

The agents optimized for willingness-to-pay because nothing stopped them. Feasibility — what the hardware can actually do — must gate the analysis before value ranking, or the system will confidently chase the richest impossible market. The fix was structural, not cosmetic: a hard gate applied ahead of segment ranking, plus a staged cost model that refused to conflate a prototype with a production machine.

### Lesson 4: Know Your Framework's Failure Modes

The token limit was set on the agent object, where CrewAI silently ignores it; every agent was capped and one truncated mid-sentence, dropping its recommendation. The fix — set it on the LLM object — is trivial once found and invisible until you measure output length deliberately. Add a completeness check; never trust that “it ran” means “it finished.”

### Lesson 5: Honest Scope Decisions Are Part of the Work

The production-grade moves — live source-retrieval via SerperDev, per-section length enforcement — were identified and deliberately deferred, documented as next steps rather than hidden. Naming what you chose not to build, and why, is a credibility signal, not a gap to paper over.

### Lesson 6: The Critique Is as Valuable as the Build

Each rebuild specification — the eight-problem mechanical critique, then the deeper feasibility-and-cost diagnosis — was itself a sophisticated document: ordered fixes, agent-level changes, success criteria, and source-of-truth data. Translating “this isn't good enough” into “here is exactly how to fix it” is a skill that compounds across every project.

### What This Project Proves

**Zero-to-production agentic AI:** an eight-agent system designed, built, and rebuilt across multiple cycles with no prior experience in any relevant technical domain.

**Diagnostic depth beyond mechanics:** caught and corrected a domain-level error — a recommended market the hardware could not serve — that polished, confident output had concealed.

**Grounded, honest output:** source-linked market data with confidence ratings, spec-anchored feasibility limits, a staged cost model, and a data-driven beachhead redirection told as a finding.

**Full execution arc:** concept → build → critique → multi-cycle rebuild → decision-ready report.

*"I can build multi-agent AI systems from scratch — and, more importantly, catch the confident, well-written answer that's wrong, supply the constraint the model never had, and rebuild until the output is correct. I refuse to let polished AI slop through."*

— END OF CASE STUDY —