REPLACING CLAUDE

Autonomous Research Pipeline for Timmy Time
Implementation Spec for Sovereign Deep Research

The last research document you should need to ask for.

Alexander Whitestone / Rockachopa • March 2026
For triage by Timmy, Kimi, and future code agents

I. What Happened and Why It Must Not Repeat

On March 22, 2026, a single Claude session produced six deep research reports covering game-agent architectures, open-source game evaluation, tool inventories, integration architecture, the Portal Architecture spec, and the Sovereignty Loop governing document. The session consumed approximately three hours of human time and substantial corporate AI inference. Every report was valuable. Every report was also a linear workflow — it would cost exactly the same to produce again tomorrow.

This document is the crystallization of that workflow. It specifies how to build the autonomous research pipeline so that Timmy can perform equivalent deep research without Claude, without Kimi, and eventually without any external API at all.

If this spec is implemented correctly, it is the last research document Alexander should need to request from a corporate AI.

II. The Research Pattern That Must Be Automated

Every deep research session follows the same six-step pattern. This pattern is entirely mechanical. Nothing about it requires frontier-model intelligence once the templates exist.

Step	What Happens	Current Tool	Sovereign Replacement
1. Scope	Human describes the knowledge gap	Chat with Claude	Gitea issue with research template
2. Query	Formulate 5–15 targeted search queries	Claude prompt engineering	Query template + local LLM slot-fill
3. Search	Execute queries, get top results	Claude web_search (10 results/query)	Timmy web_search tool (already exists)
4. Fetch	Read full pages for key results	Claude web_fetch	New: web_fetch tool (requests + trafilatura)
5. Synthesize	Compress 200K tokens into structured report	Claude Opus (frontier model)	cascade.py: Groq → Ollama local
6. Deliver	Format as PDF/markdown, file Gitea issues	Claude artifact generation	reportlab PDF + Gitea API

Steps 1, 3, and 6 are already possible with existing Timmy tools. Step 2 requires a research template library. Step 4 requires a new web_fetch tool. Step 5 requires the synthesis prompt to be good enough for a 70B model via Groq or a local 32B model to produce usable output. These are the three gaps.

III. The Three Components to Build

3.1 Research Prompt Templates (skills/research/)

The quality of Claude's deep research comes from the structured query. A 500-word prompt with clear scope, context, specific questions, and output format constraints. This is not intelligence — it is a template with slots. Crystallize it.

```
skills/research/ ■■■ tool_evaluation.md # "Find all shipping tools for {domain}" ■■■
architecture_spike.md # "How to connect {system_a} to {system_b}" ■■■ game_analysis.md #
"Evaluate {game} for AI agent play" ■■■ integration_guide.md # "Wire {tool} into {stack} with
code" ■■■ state_of_art.md # "What exists in {field} as of {date}" ■■■ competitive_scan.md #
"How does {project} compare to {alternatives}"
```

Each template is a markdown file with YAML frontmatter (name, type, typical query count, expected output length) and a body with {slot} placeholders. Timmy fills the slots from the Gitea issue description or the user's chat message. The filled template becomes the system prompt for the synthesis LLM.

```
--- name: tool_evaluation queries: 8-12 output: markdown table + recommendations cascade_tier:
groq_preferred # needs 70B for quality synthesis --- You are a technical researcher. Find all
SHIPPING, OPEN-SOURCE tools for: {domain} Context: {project_context} Stack: {current_stack}
Constraints: {constraints} For each tool found, provide: - Name, exact install command, version,
license - One-sentence description - Status: READY / EXPERIMENTAL / DEAD - Recommendation: YES /
MAYBE / NO with one-line reason Output as a markdown document with tables.
```

3.2 Web Fetch Tool (src/timmy/tools.py)

The existing web_search tool returns snippets. Deep research requires reading full pages. Add a web_fetch tool that downloads a URL, extracts clean text via trafilatura (pip install trafilatura, Apache 2.0), and returns the content truncated to a token budget.

```
import trafilatura import requests def web_fetch(url: str, max_tokens: int = 4000) -> str:
"""Fetch a URL and extract clean text content.""" resp = requests.get(url, timeout=15,
headers={"User-Agent": "TimmyResearchBot/1.0"}) text = trafilatura.extract(resp.text,
include_tables=True, include_links=True) if not text: return "Failed to extract content" # Rough
token estimate: 1 token ~ 4 chars return text[:max_tokens * 4]
```

Register this as an Agno tool in create_full_toolkit(). No external API. No cloud dependency. Pure Python, runs locally forever.

3.3 Research Orchestrator (src/timmy/research.py)

This is the main pipeline. It chains the steps together and produces a durable artifact. It runs as a Paperclip task via DistributedWorker.

```
class ResearchOrchestrator: def __init__(self, cascade, memory, tools): self.cascade = cascade
# LLM router self.memory = memory # semantic search self.search = tools["web_search"] self.fetch
= tools["web_fetch"] async def run(self, topic: str, template: str, context: dict) ->
ResearchResult: # 0. CHECK LOCAL KNOWLEDGE FIRST local = self.memory.search(topic, limit=10) if
local.confidence > 0.85: metrics.record("research_cache_hit") return
ResearchResult(source="local", content=local) # 1. GENERATE QUERIES from template filled =
template.fill(topic=topic, **context) queries = await self.cascade.generate( f"Generate 8-12
search queries for:\n{filled}", format=QueryList.model_json_schema()) # 2. SEARCH snippets = []
for q in queries.queries: results = await self.search(q) snippets.extend(results[:5]) # 3. FETCH
top pages pages = [] for s in rank_by_relevance(snippets, topic)[:10]: content = await
self.fetch(s.url, max_tokens=3000) pages.append({"url": s.url, "content": content}) # 4.
```

```
SYNTHESIZE report = await self.cascade.generate(
f"{filled}\n\nSources:\n{format_pages(pages)}", max_tokens=4000) # 5. CRYSTALLIZE into local
knowledge self.memory.store(topic, report, type="research") metrics.record("research_api_call")
# 6. WRITE ARTIFACT path = write_to_repo(topic, report) issues = extract_action_items(report)
for issue in issues: create_gitea_issue(issue) return ResearchResult( source="web",
content=report, path=path, issues_created=len(issues))
```

The critical line is step 0: check local knowledge first. Every research output gets embedded and stored. The second time anyone asks about Veloren game agents, the answer comes from SQLite in milliseconds. Zero API calls.

IV. The Cascade Strategy for Synthesis Quality

The synthesis step is the only part that currently benefits from a frontier model. Here is the cascade strategy for replacing that dependency progressively.

Tier	Model	Cost	Quality	When to Use
1 (best)	Claude API (claude-sonnet-4-20250514)	$0.50–$2.00 per report	★★★★★	Novel domains, high-stakes architecture decisions
2 (good)	Groq free tier (llama-3.3-70b)	$0.00 (rate limited)	★★★★	Most research tasks. 30 req/min free tier
3 (local)	Ollama qwen3:32b or qwen3-coder:32b	$0.00 (local compute)	★★★	Routine lookups, re-synthesis of known topics
4 (cached)	SQLite semantic memory lookup	$0.00 (instant)	N/A (pre-computed)	Any question asked before. Target: 80%+ of queries

The cascade router already exists in cascade.py. The research orchestrator passes the template's cascade_tier hint to the router. Over time, as the local knowledge base grows, Tier 4 handles an increasing share. The goal: within 3 months, 80% of research queries are answered from local cache without any external call.

V. Delegating Heavy Research to Kimi and Other Agents

For research tasks that exceed Groq's free tier or require sustained multi-page analysis, Timmy should delegate to Kimi or other agents rather than calling Claude interactively. The pattern:

Step	Actor	Action
1	Timmy	Detects research task exceeds local + Groq capacity
2	Timmy	Fills research template with full context
3	Timmy	Creates Gitea issue labeled 'kimi-ready' with filled template as issue body
4	Kimi	Picks up issue from Gitea queue, executes research

Step	Actor	Action
5	Kimi	Commits research artifact to repo, closes issue
6	Timmy	Indexes the new artifact into semantic memory
7	Timmy	Extracts action items, creates follow-up issues

This is the same pattern Alexander used today — describing context, getting research, redirecting — but with Timmy as the orchestrator instead of a human sitting in a Claude chat window. The human reviews the output and approves. The human does not write the queries, manage the context window, or format the report.

For Claude API calls specifically: Timmy calls the Anthropic API via cascade.py using the API key, paying per-token. A deep research query that costs $20/month via subscription costs $0.50–$2.00 via API. And the result gets indexed locally so the same question costs $0.00 the second time.

VI. Implementation Priority

Priority	Component	Effort	Dependency	Sovereignty Impact
P0	web_fetch tool (trafilatura)	2 hours	None — pure Python	Enables all research without Claude
P0	Research template library (6 templates)	4 hours	None — markdown files in repo	Crystallizes the prompt engineering skill
P0	Research orchestrator (src/timmy/research.py)	1 day	web_fetch + templates	End-to-end autonomous research pipeline
P1	Semantic index for research outputs	4 hours	nomic-embed-text via Ollama	Cache hit = zero API cost repeat
P1	Gitea issue creation from research findings	4 hours	Gitea API (already used)	Auto-triage into engineering backlog
P1	Paperclip task integration	4 hours	TaskRunner (exists)	Research runs autonomously
P2	Kimi delegation via Gitea labels	2 hours	Kimi watching Gitea queue	Heavy research without Claude subscription
P2	Claude API fallback in cascade.py	2 hours	Anthropic API key	Frontier quality at $0.50 not $20
P2	Research sovereignty metrics + dashboard	4 hours	Metrics emitter from Sovereignty Loop	Track cache hit % and API cost trend

Total estimated effort: approximately 5 days of focused work. After that, every research question answered locally makes the next one cheaper. The compound interest of crystallized knowledge.

VII. How to Know It's Working

Metric	Week 1	Month 1	Month 3	Graduation
Research queries answered locally	10%	40%	80%	>90%
API cost per research task	$1.50	$0.50	$0.10	<$0.01
Time from question to report	3 hours (human + Claude)	30 min (Timmy auto)	5 min (mostly cache)	<1 min (all cache)
Human involvement per research task	100% (full session)	Review only (10 min)	Approve only (2 min)	None (auto-approved)
Research artifacts in local index	6 (today's session)	30+	100+	Comprehensive

The graduation test for research sovereignty: Timmy receives a Gitea issue asking for a state-of-the-art evaluation of a new technology domain. Timmy searches local knowledge, finds relevant prior research, identifies the gap, formulates queries, searches the web, fetches pages, synthesizes a structured report, commits it to the repo, creates follow-up issues, and indexes the result — all without a single message to Claude or any human.

VIII. The Transfer

Today's session produced six reports, a governing architecture document, and this spec. That is approximately $30–40 of corporate AI inference, three hours of human attention, and 50,000+ words of output. All of it is now trapped in PDF files and Claude chat history unless it gets transferred.

Immediate actions to close out this session:

1. Commit all PDFs and research artifacts to the Gitea repo under docs/research/. 2. Create a Gitea epic: 'Research Sovereignty Pipeline' with sub-issues for each P0 and P1 component listed in Section VI. 3. Extract the GamePortal Protocol code from the Portal Architecture report into an actual portals/protocol.py file. 4. Extract the stack_manifest from the tool inventory into a machine-readable JSON file that Timmy's local agent can query. 5. Write SOVEREIGNTY.md at the repo root containing the five metrics from the Sovereignty Loop document and the crystallization function signature. 6. Index all six research reports into Timmy's semantic memory via nomic-embed-text so the knowledge persists beyond this chat session.

The last research document you should need to ask for.

The arch must hold after the falsework is removed.

Timmy Time Project • Research Sovereignty Spec • v1.0