Mission Integration Group

CLONABLE CLAUDE PROJECT · BUILD GUIDE

The Silver Medalist Re-Activation System

Your ATS is a graveyard of cleared candidates you already qualified, already interviewed, and already lost to a counter-offer. This guide clones the exact Claude project that digs them back up — ranked, scored, and ready to re-contact.

Build time: 1–3 days · **IT needed:** one read-only export · **You own it after handoff.**

WHAT YOU'RE BUILDING

One Claude project that reads your candidate history and tells you who to call back.

You give Claude an export of past candidates and a current open req. It scores every candidate against that req, flags clearance and program matches, weighs how long it's been since their last evaluation, and hands back a ranked shortlist — each with a one-sentence reason to re-engage.

> **Why this is the first build for most teams.**
>
> It uses data you already paid to collect, needs only a read-only export, and produces a list a recruiter can act on the same afternoon. Lowest effort, highest signal. If you do one thing with Claude, do this.

What you need before you start

- A Claude plan that keeps your data private — **Team or Enterprise**, never a free or personal account for candidate PII.
- A CSV export from your ATS of candidates in **"final round" or "silver medalist" status** over roughly the last three years.
- One **current open requisition** to score against.

> **Data handling, up front.**
>
> Run this on Claude Team or Enterprise, where your inputs are not used to train models. Strip fields you don't need (SSN, DOB) before upload. When in doubt, anonymize. No real cleared-candidate data in a non-compliant environment — no shortcuts.

STEP 1

Get the export.

Ask your ATS admin for a CSV of every candidate who reached final-round or silver-medalist status in the last three years. Most admins can pull this in about fifteen minutes — it's a read-only report, no integration, no write access.

Fields to request

Field	Why Claude needs it
Name (or candidate ID)	Identify the record
Last evaluation date	Weigh how stale the relationship is
Role applied for	Compare against the current req
Recruiter notes	The free-text judgment generic search can't read
Clearance level recorded	Match against the req's clearance bar
Disposition reason	Distinguish "lost to counter-offer" from "failed screen"

> **The disposition field is the gold.**
>
> A candidate who was an offer-stage loss to a counter-offer is a completely different prospect from one who washed out on a technical screen. That distinction lives in free text — which is exactly what Claude reads and keyword search ignores.

STEP 2

Create the Claude project. Paste the system prompt.

Create a new Claude Project, upload your CSV as project knowledge, and set the system prompt below. This prompt is the product — it's where your cleared-recruiting logic lives. Copy it exactly, then tune it to your judgment in Step 3.

SYSTEM PROMPT — PASTE INTO YOUR CLAUDE PROJECT

```
You are a cleared recruiting analyst. When given a list of past candidates and a current open requisition, evaluate each candidate against the req and score them 1–10 on fit. Flag clearance-level matches, program-experience matches, and time-since-evaluation. Output a ranked table with a one-sentence re-engagement rationale for each candidate.
```

That's the whole engine. Everything else is data going in and a ranked list coming out.

STEP 3

Run it. Refine until it matches your judgment.

Paste your open requisition into the project and run it. Read the ranked output critically: where does Claude's ranking disagree with your gut? Adjust the prompt to close that gap — add the criteria you weight that it missed, tell it how to treat a stale evaluation, name the programs that actually transfer. Two or three iterations is typical before the logic lines up with how you'd rank the list yourself.

> **You'll know it's right when** the top of the list is the people you'd have named from memory — plus a few you'd genuinely forgotten. That second group is the return on the build.

WHAT THE OUTPUT LOOKS LIKE

A worked example.

Illustrative only — built from synthetic data to show the shape of what Claude returns.

Input: the open req (pasted into the project)

> **Senior RF / EW Engineer** — TS/SCI required, CI poly preferred. AEHF or protected-comms program background. Onsite, cleared facility. Backfill, needs to move quickly.

Output: ranked re-engagement shortlist

Rank	Candidate	Fit	Re-engagement rationale
1	D. Reyes	9 / 10	TS/SCI active, strong AEHF background, lost to a counter-offer in 2021 — exactly the profile, worth a direct call.
2	M. Okafor	8 / 10	Protected-comms experience, clearance current; last eval 18 months ago, re-confirm availability.
3	S. Whitfield	6 / 10	Adjacent RF work, clearance match; would need program ramp — second-tier outreach.

From a 3,000-row export, a recruiter gets a one-screen call list instead of an afternoon of scrolling — and the reasoning is visible, so they can trust or override each line.

STEP 4

Hand it to your team. You own it.

Once the ranking matches your judgment, the project belongs to your team. Train them to drop in a fresh export and paste a new req whenever a role opens — the project runs the same logic every time. Nothing depends on an outside consultant staying in the loop.

The full pipeline, end to end

Layer	In this build
Data	ATS export (CSV), any platform
Middleware	None — manual upload to Claude Projects
Claude	The reasoning layer; the system prompt above
Output	Ranked shortlist → Google Sheet or copy/paste

> **When this build grows up.**
>
> At a few hundred records, manual upload to Claude Projects is plenty. Past roughly 500 records with heavier logic, the same idea moves to the Claude API with a batch script — that's a scoped step, not a starting point. Begin manual; scale only when volume forces it.
