Step-by-Step Build Guide — Build Your Own Municipal BD Agent

*A hands-on companion to *The Municipal BD Intelligence System* playbook (TUJ Edition)*

Web edition v1.5 — October 24, 2026. This guide mirrors §4 of the case-study page; the two are published together.

What this is	A follow-along tutorial. By the end, you will have built a working version of the BD intelligence agent described in the playbook — with your own market, your own accounts, and your own dashboard.
Who it is for	Students new to agent building. No prerequisites — the build is prompts and a file, not code.
Time required	About 8–9 hours of building in total — Phase 1 (Steps 1.1–1.4) about 2 hours; Steps 2.1–2.2, 2–3 hours; Steps 2.3–2.5, about 2 hours; Steps 3.1–4.1, about 2 hours — plus two weekly cycles to feel the rhythm. Work it in sittings of your choice.
Companions	Optional: the *Playbook* (the methodology) and the *Technical Addendum* (the architecture). This guide is self-contained — read them when you want the why behind a step.

Before You Start

You will need:

- An account on an agent platform (Step 1.1 helps you choose — the original was built on Google Gemini).
- A web browser and a text document or spreadsheet for your portal registry draft.
- A market to work on: one metropolitan area and one professional-services sector you can research publicly. (The classroom default is Greater Tokyo municipal water: Tokyo Bureau of Waterworks, Tokyo Bureau of Sewerage, Setagaya City, Yokohama City, MLIT.)
- A folder for your outputs — every step produces an artifact you will reuse in a later step. Save everything.

The whole build in 30 seconds. You are building exactly one thing: a single file — `bd-system.md` — that remembers everything your business-development system knows: your market, your portals, what is open, who matters, and what to do next. Each week the loop runs once: portals publish → the agent scans → the agent verifies → the file updates → the dashboard renders → you review (15–30 minutes). The eleven steps below assemble that file piece by piece; each step adds exactly one section, and nothing is shown to you before the step that creates it.

The system file. Your system's memory is a single Markdown file — name it `bd-system.md` — written as an agent **skill file** — a Markdown document an agent loads at the start of a task, carrying instructions and knowledge in one place: one section for each part of the work — but they are deliberately not listed here: each section appears at the step that creates it, and the complete skeleton arrives at Step 1.4. Start every task by giving the agent the current file (upload it, or paste its full text); finish every task by saving the complete updated file it returns — dated, never overwritten. A fresh task plus the file beats a long, drifting chat history: the file is the memory; the chat is disposable. (The original system kept this state inside the dashboard, with JSON as its backup format; the classroom build uses Markdown because you can read and repair it. The dashboard's own Backup JSON button is its internal backup — a different thing.)

When a season of weekly runs makes the file long, archive the closed items (dead Tier 1s, past events, old log entries) into a dated appendix file — the agent reads the whole file every time, so keep the working copy lean.

How the build is organized. Eleven steps in four phases, in the order the original system was actually built — the order of the builder's own skill file. **Phase 1, Foundation (Steps 1.1–1.4):** who you are, where you look, the install, the data model. **Phase 2, the Weekly Workflows (Steps 2.1–2.5):** scan, verify, events, dossiers, Gold Sheets. **Phase 3, the Deliverable (Step 3.1):** the dashboard. **Phase 4, Trust & Automation (Step 4.1):** audit, operating rules, schedule. Every step ends by writing one section of your bd-system.md. The thirteen prompts are scaffolding: they assemble into one **Final Prompt** at the end of Step 4.1 — the complete instructions your agent runs on.

Work every step on the Agentic Loop. Read → Edit → Check → Adjust. Read the sources yourself; let the agent make the change; check the result against the step's checkpoint; then adjust — and write what you learned into the system file or the standing instructions, so the next loop starts smarter. The finished system runs the same loop by itself, every week.

Time budget — and the fast track. Plan roughly: Phase 1 (Steps 1.1–1.4), about 2 hours. Steps 2.1–2.2, 2–3 hours (real portals are slow, and official PDFs are long). Steps 2.3–2.5, about 2 hours. Steps 3.1–4.1, about 1 hour, plus two weekly cycles to feel the rhythm. Short on time? Run the fast track first — **1.1 → 1.2 → 1.3 → 1.4 → 2.2 → 3.1** — for an end-to-end skeleton, then add 2.3, 2.4, 2.5, and 4.1.

One honesty note. The prompts in this guide are faithful reconstructions of the builder's working instructions — the originals were conversational and were never exported as a document. Part 3 tells the full story.

The map. Eleven steps. Each step tells you what *you* do, gives you the *exact prompt to paste*, shows what you *should get back*, and ends with a *checkpoint* and a *fix* for the most common failure.

Step	You do	The agent does	Artifact you keep
0	Choose the agent platform	—	Platform decision
1	Define your practice	Drafts your Summary, Territory, Team & Positioning	Skill file header (3 sections)
2	Build the portal registry	Validates and structures the list	Portal registry

3	Install the master instructions; teach the data model	Memorizes methodology, schemas, guardrails; restates schemas	A working agent + empty templates
4	Order the first scan	Browses portals; collects candidates	Raw candidate list
5	Order verification	Checks official postings; classifies	Tier 1 / Tier 2 system file
6	Run the event radar	Confirms events; writes the briefing	Events + pre-event briefing
7	Pick the key official	Drafts the six-dimension dossier	Dossier #1
8	Pick the first account	Drafts its Gold Sheet	Gold Sheet #1
9	Order the dashboard	Generates the single-file HTML	Dashboard v1
10	Audit, write the rules, schedule the run	Self-reports weaknesses; executes the full cycle on schedule	A running system

Words you need first. *Solicitation / tender:* a public request for proposals or prices — the thing you are hunting. *RFQ vs RFP:* an RFQ asks "are you qualified?"; an RFP asks "what exactly would you do, and for how much?" *Tier:* Tier 1 is verified and live; Tier 2 is an early signal, not yet a tender. *Gold Sheet:* the one-page strategy for one account. *Catalyst:* the event that makes now the right moment. *Cone of silence:* after a tender is published, vendors stop lobbying about it. *Red Flag:* anything unknown or unverified — named, never hidden. The full glossary waits in the appendix.

Part 1 — Step 1.1: Choose Your Agent

Phase 1 of 4 · Foundation — Step 1.1

What "an agent" means here

For this project, an **agent** is five things working together: a large language model (the **brain**), **tools** (the **hands** — above all, a real web browser), **standing instructions** (the **rules** — a persistent prompt that defines its job), a **memory** (the **system file** it reads at the start of every task and updates at the end; the chat itself is forgotten), and a **clock** (a loop that runs the whole cycle on schedule and produces files you can keep). A chatbot window alone is not enough — check for the tools and the persistence.

The six selection criteria

Score every candidate platform against these. The project fails without #1 and #3.

1. **Real-browser browsing.** It must open JavaScript-rendered portal pages and return their actual content. Ask it to open one of your target portals during your trial — if it returns an empty page or a summary of the homepage, it fails this project. The pass/fail test: ask it — "Open https://www.e-procurement.metro.tokyo.lg.jp/ and list the first five posting titles with

dates. If the page is empty, or you can only summarize the homepage, say 'unreadable' and why." If it fails, do not let it work from memory: paste the portal's page text or its PDFs into the chat, or take the code route (the case-study page, §9).

2. **Long context.** A model can only hold so much in mind at once (its context window, measured in tokens); it must hold your registry, schemas, the system file, and several source documents in one working session.

3. **File output.** It must generate a complete file (the dashboard HTML, the updated system file) that you can download and keep — not just display text.

4. **Persistent instructions.** A way to save your master instructions once (a custom agent, a project, a saved system prompt) so you do not re-paste them every session.

5. **Scheduling — or a tolerable manual loop.** A built-in scheduled run is ideal (the original runs itself once a week, unattended). Without one, you will run the weekly prompt by hand — acceptable for a class project.

6. **Access and cost you actually have.** A platform your university provides beats a theoretically better one you cannot log into. Verify current plans and limits on the provider's site; they change often.

The realistic options

Option	What it is	Fit for this project
Google Gemini — a Spark agent *(the original build)*	Google's agent environment: browsing, file generation, scheduled runs, Google Drive/Gmail integration	Mirrors the original system step for step. One valid choice — not a requirement.
ChatGPT (agent / browsing modes, Projects)	OpenAI's platform: browsing and agent modes, project-level persistent instructions	Equally valid — the same prompts run here; check your plan's agent-mode limits.
Claude (Projects, web search)	Anthropic's platform: strong long-document handling, project instructions	Equally valid; strong on long documents. Scheduling runs manually, with Prompt 13.
DIY: Python + an LLM API	You write the program: Python (e.g., Playwright for browsing, an LLM API for extraction/judgment)	The most work and the most learning. You become responsible for the loop, the browsing, and the retries. Choose this path if your course goal is code, not just outcome.

Checkpoint 0. You can name your platform and point to where each of the six criteria lives in it (If one criterion is missing, state which and how you will compensate — e.g., "no scheduler → I run Prompt 13 by hand every week, on my run day." Two criteria cannot be waived: without real browsing (#1) or file output (#3) the system cannot run at all — if your platform lacks either, choose a different one.). The tool is not the method — any platform meeting the six criteria produces similar results, so choose by the access you actually have.

Platform facts — checked October 23, 2026. Gemini Spark is Google's personal agent inside the Gemini app, shaped by Tasks, Skills and Schedules. Google's support pages say it needs a personal Google Account (work and school accounts are not supported), age 18+, Keep Activity on, and a paid plan — Google AI Pro (US$19.99/month) or Google AI Ultra (from US$99.99/month). It is available in Japan and in most countries where Gemini Apps are supported — not the EEA, the UK, Switzerland or Nigeria. The model under Spark is Gemini's current one and changes over time (Google's posts have named successive Gemini Flash models), so this guide prints no model number as a requirement. The builder's own notes name the builds he used; they do not appear on Google's public pages, so they are not printed here. Source: Google One support, "Use Google AI Ultra benefits" (support.google.com).

Part 2 — Build It

Step 1.2 — Define your practice: who you are, where you play, what you sell

Phase 1 of 4 · Foundation — Step 1.2

Goal. Before any tool work: the background every later step assumes. The builder's own skill file opens with exactly this — a summary, a territory, a team — so yours does too. Invent your (fictional) consulting practice: a name, a home base in Greater Tokyo, the services you sell, and the one line you compete on. Draw your territory the way the original does: named agencies in, everything else explicitly out. And state your goal for this system in one sentence: what does winning look like this semester? Run Prompt 1 in a plain chat on your chosen platform — the custom agent does not exist yet; that is Step 1.4. Steps 1.2–1.3 produce the material it is assembled from.

Do this. Answer the five questions yourself first — on paper if you can — then let the agent draft the sections with **Prompt 1**.

The classroom default territory is Greater Tokyo: TMG Bureau of Waterworks, TMG Bureau of Sewerage, Setagaya City, Yokohama City Waterworks Bureau, Kawasaki City, Hachioji City, MLIT Kanto Regional Development Bureau — accept it, or draw your own; Prompt 3 will use whatever you decide here.

Prompt 1 — Draft my practice definition:

```
I am setting up a business-development intelligence practice.
Interview me, then draft the first three sections of my system file.

Ask me, one at a time:
1. My firm's name, where its office is, and the nearest station — the 45-minute event radius is
measured from that station.
2. The 3-5 services we sell to municipalities - chosen
   deliberately, against real Tokyo-area demand, not improvised. Mark each service core or adjacent:
core = what we lead with and can prove; adjacent = what we can credibly support. Anything not listed is
out — the fit ratings depend on this.
3. Our one-line positioning: what we lead with that a generalist
   firm cannot honestly say.
4. My territory: the agencies I cover - and the neighboring
   regions I explicitly exclude.
5. My goal for this system, in one sentence.
6. My team: 2-3 members - fictional name, credential, background.
 (Dossiers and ice-breakers draw on these people.)
```

```
Then return three sections, in this order:
## Summary - what this practice is; what this system does.
## Territory - covered agencies; strict exclusions.
## Team & Positioning - who we are; our people (names,
 credentials, backgrounds); what we lead with.
Plain language. Where I have not decided, write [UNDECIDED] -
do not decide for me.
```

A worked example (fictional). *Hoshino Water Partners*, office by Shinjuku Station. Services: (1) pipe-network asset management and renewal planning; (2) leakage detection and pressure management; (3) operational dashboards for water utilities. Positioning — weak: "We provide high-quality engineering consulting services." (Any firm could say it; it belongs to none.) Strong: "We lead with renewal plans costed from the utility's own pipe-age and leak data — a generalist firm cannot honestly claim a leakage track record." Territory: TMG Waterworks, TMG Sewerage, Yokohama, Kawasaki, Setagaya, Hachioji, MLIT Kanto; exclusions: private operators, and everything outside Kanto. Goal: "A running system that surfaces two pursuable pre-RFP windows a month by semester's end."

Checkpoint 1. Read the three sections as a stranger. Could you say, in one breath, who this firm is, where it plays, and why it wins? If the positioning sentence could belong to any firm, it belongs to none — sharpen it. Save the sections at the top of your bd-system.md; Step 1.3 builds on the Territory section, and Step 2.2's fit ratings key off the service lines you just chose. Resolve every [UNDECIDED] before Step 1.4 — the master instructions cannot be written around blanks.

Your file so far: Summary · Territory · Team & Positioning

Step 1.3 — Build your portal registry

Phase 1 of 4 · Foundation — Step 1.3

Goal. Build the registry: the explicit list of where opportunities in your market are published. In the builder's skill file this section is called "Procurement Hubs" and groups portals by platform — public sourcing hubs, portal networks, city bidding pages. The registry is the system's coverage boundary: anything not in it does not exist for the system. Your agency list (6–12 agencies) comes from Step 1.2's Territory section, and your service lines are already decided there — everything downstream keys off them, especially the fit ratings in Step 2.2. (Also a plain chat — the agent still does not exist.)

Do this. Draft your agency list first (6–12 agencies). Then paste **Prompt 2** with your list filled in.

Prompt 2 — Registry:

```
Here are the agencies in my market:
[PASTE YOUR LIST - e.g., Tokyo Bureau of Waterworks; Tokyo Bureau of
Sewerage; Setagaya City; Shinjuku City; Yokohama City Waterworks Bureau;
MLIT Kanto Regional Development Bureau]

For each agency, find and record:
- the official procurement / tender portal name and its direct URL,
- whether the portal is JavaScript-rendered (a single-page app),
- the categories of procurement it publishes,
- the notice types it publishes (tender notices, proposal calls,
 and so on),
- whether it is public without login - if it needs a login,
 record "login required"; do not guess.
Return a registry table. If you cannot find an agency's portal, say so
explicitly - do not substitute a similar-sounding site.
```

You should get back. A registry table with a real, openable link per agency, and honest gaps.

Checkpoint 2. Click every link yourself. Any registry row you have not personally opened does not count.

Your file so far: Summary · Territory · Team & Positioning · Portal Registry

If it goes wrong. The agent returns aggregator sites instead of official portals → reply: *"Official sources only. Replace every aggregator with the agency's own portal, or mark the row 'portal not found'."*

Step 1.4 — Install the master instructions, and teach the data model

Phase 1 of 4 · Foundation — Step 1.4

Goal. Give the agent its permanent job description — assembled from what you have already written, not handed down ready-made: Prompt 3's deployment section is your Step 1.2 territory and your Step 1.3 registry, written into the instructions. You write this once; every later prompt assumes it. What you install here is the seed of the Final Prompt — this same document, completed, is assembled at the end of Step 4.1. Length check: Prompt 3 is long — if your platform caps instruction length (some surfaces do), attach Prompt 3 as a project knowledge file instead of pasting it into the instruction box, and tell the agent that file outranks the chat. Where each schema is first used: Tier 1 and Tier 2 at Step 2.2, Event at Step 2.3, Dossier at Step 2.4, Gold Sheet at Step 2.5 — all five are defined once, in Prompt 3. Lines starting `<<REPLACE:` inside a prompt are notes to you, not to the agent — act on them, then delete the marker before you paste.

Do this. Create a new custom agent / project on your platform (name it: `Municipal BD Analyst — [Your Market]`). Paste **Prompt 3** as its standing instructions. If your platform has no persistent instructions, save Prompt 3 in a document and paste it at the start of every session — the steps below work either way. Where this lives on your platform: on **Gemini**, a Spark Skill — Gems hold instructions too, but they cannot browse on a schedule; on **ChatGPT**, a Project's instructions (or a custom GPT); on **Claude**, a Project's instructions. (Checked Oct 2026.)

Prompt 3 — Master instructions (paste once):

```
ROLE
You are the business-development (BD) intelligence analyst for an engineering
consulting practice that sells professional services to municipal governments.
You run a weekly intelligence cycle and maintain a single dashboard.

METHODOLOGY (Miller Heiman)
- Every priority account is managed on a one-page Gold Sheet (account level,
  1-3 year horizon).
- For every account, identify the four Buying Influences: Economic Buyer
  (approves the money), User Buyer (lives with the result), Technical Buyer
  (can veto, cannot approve), Coach (an insider who teaches us how to win).
- Assess each account's Response Mode: Growth (expanding),
  Trouble (hurting), Even Keel (stable, no urgency), or
  Overconfident (complacent).
- Every unknown or risk is a Red Flag. Surface Red Flags explicitly; never
  hide them.
- Every recommendation is expressed as a Win-Result: what the specific client
  person gains.

DATA SCHEMAS (never change these without my instruction)
Tier 1 - Verified Solicitation (defined here): agency, solicitation_id, scope,
 application_deadline, question_deadline, submission_deadline,
 opening_date (each: date + time, JST, and the original Japanese
 quote; "not stated" when the notice omits it), portal,
 direct_link, fit (core / adjacent / out), status.
Tier 2 - Pre-RFP Target (defined here): agency, initiative, catalyst_and_timing
 (a timing window only if the source states or implies one;
 otherwise "timing unknown" - never a single release date),
```

```
 lead_capability, portal_base, strategic_action, source_url,
 source_date, source_quote_original (the Japanese line that
 supports the timing).
  (There is NO deadline field in Tier 2. Never add one.)
Gold Sheet (defined here): account_overview, response_mode, buying_influences (for each:
  name, role, stance (supportive / neutral / opposed /
  unknown), win_result), competitive_landscape, vulnerabilities,
  strategy, win_results, action_plan, red_flags, last_updated, window_opens, window_closes
 (dates, each with a source; "window unknown" if no source
 states them).
Dossier (defined here): name, title, agency, buying_influence, plus six dimensions:
  (1) responsibility & role, (2) background & pedigree,
  (3) core similarities, (4) conferences & gatherings,
  (5) high-frequency keywords, (6) ice-breaker pitch.
  Every factual claim must name its public source.
Event (defined here): name, date, time, venue, target_stakeholders
 (expected roles, not confirmed attendees),
  registration_link, source.

DEPLOYMENT - GREATER TOKYO (this build's territory)
<<REPLACE: this block with your own territory, agencies and
portals, from your Steps 1.2-1.3; then delete this marker. The
Greater Tokyo text below is the classroom default.>>
- Territory: Tokyo Metropolis (23 Special Wards + Tama), Kanagawa
  (Yokohama, Kawasaki), Saitama, Chiba. Nothing outside it.
- Target agencies: TMG Bureau of Waterworks, TMG Bureau of Sewerage,
  Setagaya City, Yokohama City Waterworks Bureau, Kawasaki City,
  Hachioji City, MLIT Kanto Regional Development Bureau.
- Official portals (verify here, nowhere else): Tokyo e-Procurement,
  e-Tokyo Joint, MLIT PPI, Yokohama e-Bidding.
- Events: JSCE (Japan Society of Civil Engineers), JWWA (Japan
  Water Works Association), JSWA (Japan Sewage Works
  Association) and MLIT forums - within a 45-minute
  transit radius of our office (origin station:
 [YOUR OFFICE STATION]), inside a rolling 14-day window.

GUARDRAILS (hard rules)
1. Tier 1 requires a confirmed ID, its dates verified on the official posting page (every date the
notice states), and a direct link. If you cannot verify, the item goes to
   the Red Flag list. Never guess a date, an ID, or a name.
2. Public information only. Never invent people, postings, or figures.
3. You never contact anyone. You draft; I send.
 Exception: the weekly digest email to my own address.
4. Once a solicitation is posted, no positioning contact about it
   (the "cone of silence" - once a tender is public, vendors must not
   lobby about it; all communication goes through the official
   channel). The positioning work happens before posting, in Tier 2.
5. Anything shared outside the practice uses fictional data, labeled DEMO.

WEEKLY CYCLE
Scan -> Verify & classify -> Events & briefings ->
Update dossiers & Gold Sheets -> Render dashboard ->
Archive & send to me -> I review and approve.
Scan window: deadlines up to 45-60 days out; farther deadlines enter the window in later cycles.

OUTPUT DISCIPLINE
After every task, return (a) a short human summary of what changed,
(b) the complete updated system file (Markdown) - one downloadable
.md file or one code block - its sections following the schemas
above, and (c) a CHANGELOG listing every row or section added,
changed or removed. Do not change any row you were not asked
to change. If anything is
uncertain, say so in the summary and add a Red Flag.
```

You should get back. An acknowledgment in which the agent restates its job in its own words.

Checkpoint 3 (instructions). Ask it two test questions: *"A rumor says City X will issue an RFQ next month, no posting yet — which tier, and what fields?"* (Correct: Tier 2, and no deadline.) *"You found a posting but the deadline is only on an aggregator site — what do you do?"* (Correct: verify on the official page or Red-Flag it.) Wrong answers mean the instructions did not land — re-paste Prompt 3 and test again before proceeding.

Then teach the data model. Before the agent touches real data, make it prove it holds the schemas correctly.

Your bd-system.md — the complete skeleton. Do not fill in the new sections by hand — your Summary, Territory, Team & Positioning, and Portal Registry are already written (Steps 1.2–1.3); each workflow step adds its own section as real data arrives. Save an empty copy now; Prompt 4 makes the agent prove it can restate it.

```
## Summary
What this practice is · what this system does · the goal, in one sentence

## Territory
Covered agencies · Strict exclusions

## Team & Positioning
Who we are · The services we lead with · Our one-line positioning

## Portal Registry
Agency | Portal | URL | JS-rendered? | Categories watched | Notice types | Public without login?

## Tier 1 — Verified Solicitations
Agency | ID | Scope | Application | Questions | Submission | Opening | Link | Fit | Status

## Tier 2 — Pre-RFP Targets (no deadline field — by design)
Agency | Initiative | Catalyst & timing | Lead capability | Strategic action | Source (link, date)

## Gold Sheets
### [Agency] — Response Mode: [Growth / Trouble / Even Keel / Overconfident]
Buying influences · Red Flags · Current position · Window opens / closes (each with a source) · Next
action

## Dossiers
### [Name] — [Title], [Agency]
The six dimensions, one short block each

## Events (next 14 days)
Date | Event | Venue | Named speakers / hosts | Link

## Red Flags
What is unverified — and why

## Execution Log
[Date] — what ran, what changed

## Operating Rules
The boundaries — left empty until Step 4.1, when your audit writes them
```

Prompt 4 — Schema check:

```
Using the schemas in your instructions, produce:
1. An empty Tier 1 template and an empty Tier 2 template (field names only).
2. One FILLED example of each, using entirely fictional data, labeled
   FICTIONAL EXAMPLE.
3. A list of the differences between Tier 1 and Tier 2.
Do not browse anything for this task.
```

You should get back. Templates you can inspect field by field.

Checkpoint 3 (schema). The fictional Tier 2 example must contain **no deadline** anywhere — and must carry its source: link, date, and the quoted line. If it has one, reply: *"Tier 2 has no deadline field by design. Remove it and regenerate."* — and note this moment: you have just enforced a schema, which is the core skill of agent building.

Your file so far: the header and registry, plus empty templates for Tier 1 · Tier 2 · Events · Dossiers · Gold Sheets · Red Flags · Execution Log · Operating Rules

> **Working with Japanese sources — read this before your first scan.** Almost everything you are about to read — portal postings, assembly agendas and minutes, capital plans, officials' bios — is published in Japanese. That is expected, and it is not a blocker: your agent reads Japanese and reports to you in English; you verify with the links and dates it gives you. Three habits matter. **(1) Era dates:** official documents often date things by era year (Wareki) — Reiwa 8 is 2026. Prompt 6 forces a conversion check on every deadline; never skip it. **(2) Agendas are the government's to-do list:** an assembly agenda and its attached materials are where a city announces what it plans to discuss, approve, and spend on — that is why Step 2.1 scans them. **(3) Names:** keep each official's Japanese name alongside the romanized one in your system file, so a source can always be found again.

Step 2.1 — Run the first scan

Phase 2 of 4 · The Weekly Workflows — Step 2.1

Goal. Collect raw, *unverified* candidates — solicitations, planning signals, and events — from your registry sources. Events are only gathered here; Step 2.3 works them into the confirmed radar and the briefing.

First run: one agency only — your strongest registry entry; widen to the full registry after the Step 2.2 checkpoint.

Prompt 5 — Scan:

```
Run a scan for [YOUR MARKET] now. FIRST RUN: scan
[ONE AGENCY] only; after the Step 2.2 checkpoint, re-run for the
full registry.
Sources: the portal registry in the attached system file, plus each
agency's own capital improvement plans (planning / budget pages),
council/assembly agendas (assembly / council meeting pages), and
appointment announcements (news releases).
Look for: (a) newly posted solicitations in our service lines –
a deadline inside a rolling 45-60 day lookahead (long response cycles are the valuable ones – a
deadline farther out enters the window in a later cycle, so nothing is lost),
(b) planning signals - budget allocations, facility renewal plans,
expiring incumbent contracts, new appointees,
(c) events: professional gatherings (JSCE, JWWA, JSWA, MLIT listings)
in the next 14 days, within a 45-minute travel radius of our office.
Search in Japanese as well as English - notice types: 入札公告、
一般競争入札、指名競争入札、公募型プロポーザル、
総合評価落札方式、随意契約.
Return a RAW CANDIDATE LIST only: source, what you found, date of the
source, and the link. For events, include the date, time, and venue.
Do not verify, classify, or summarize yet.
If a portal returns nothing readable, list it under "unreadable sources"
with the reason.
```

You should get back. A messy, honest list — including an "unreadable sources" section. Messy is correct at this stage.

Checkpoint 4. Spot-open three links from the list. If the agent claims a portal was scanned but the link is the portal's homepage rather than a findings page, say so and re-run that agency.

Step 2.2 — Verify and classify

Phase 2 of 4 · The Weekly Workflows — Step 2.2

Goal. Turn raw candidates into the two-tier state — the heart of the system. (Field definitions: Prompt 3.)

Prompt 6 — Verify & classify:

```
Take the raw candidate list from the previous step.
For each candidate:
1. Open the OFFICIAL posting or source page (not an aggregator).
2. If it is a live solicitation: confirm the solicitation ID and every date the
 notice states - application, questions, submission, opening -
 each with date AND time (JST), the direct link, and the original
 Japanese wording for each date, quoted as evidence. A date the
 notice does not state is recorded as "not stated". Classify fit:
   core / adjacent / out, using these service lines:
   [THE CORE LINES YOU DEFINED FOR YOUR PRACTICE IN STEP 1].
3. If it is an early signal: record it as Tier 2 with its catalyst and
 timing window - and its source (link, date, and the original
 Japanese line that supports the timing). If no source states a
 window, write "timing unknown". No deadline.
4. Anything you cannot confirm goes to the Red Flag list, with the
   reason.
5. Dates: Japanese official documents often use era (Wareki) years -
   convert them to Gregorian before recording (Reiwa 8 = 2026), and
   double-check every converted deadline. But
 the fiscal-year form (Reiwa 8-nendo) means fiscal year 2026
 (April 2026 to March 2027): record fiscal years as FY2026,
 never as a calendar date. All times are JST.
Return: (a) a summary of what changed, (b) the Tier 1 table,
(c) the Tier 2 table, (d) the Red Flag list, (e) the events list,
(f) the complete updated system file.
```

You should get back. Two tables, a Red Flag list, an events list, and the updated system file. Save the system file — it is your system's memory from now on.

Checkpoint 5. Pick two Tier 1 rows and verify ID + deadline against the official pages yourself — the agent must quote the original Japanese line for each deadline (Prompt 6), so what you are checking is the quote and its era conversion, not your Japanese. Then try to break it: ask *"Why is [one Tier 2 item] not in Tier 1?"* — the answer must cite the missing verification, not a vibe.

Your file so far: Tier 1 · Tier 2 · Events · Red Flags now hold real, verified entries

Step 2.3 — Run the event radar, and write the pre-event briefing

Phase 2 of 4 · The Weekly Workflows — Step 2.3

Goal. The original skill file gives events their own section, because events are where the people are. Take the events list from Step 2.2 and work it properly: confirm every event against its official listing, enforce the radar rules (14-day window, 45-minute radius of your office), and — for the one event that matters most — write the pre-event one-page briefing the original produces before key meetings: who will be there, what they care about, what you could say. (Event schema: Prompt 3.) First-cycle note: you have no dossiers yet — the briefing names who matters, and Step 2.4 builds the first dossiers from that list; from cycle two on, briefings draw on the dossiers you have.

Do this. Paste **Prompt 7** with your current system file attached.

Prompt 7 — Events & briefing:

```
Take the events list in the attached system file.
1. Confirm every event against its official listing (the
   association's or organizer's own page): exact date, start
   time, venue, and a working registration link. Drop anything
   you cannot confirm, and say why.
2. Apply the radar rules: next 14 days only; within a 45-minute
 travel radius of our office (origin: [YOUR OFFICE STATION]) -
 state the route you used for each 45-minute estimate.
3. For the single most valuable event, write a one-page
   pre-event briefing: who is named on the official event page as speaker,
 host or sponsor (names, titles, agencies, with the link -
 drawing on the dossier sections where we have them).
 Label every other expected attendee "attendance unconfirmed" — do not guess names. Then, for each key
person: what they care about right now, two ice-breaker questions, and what to listen for.
Return: the confirmed events table, the briefing, and the
complete updated system file.
```

Checkpoint 6. Open two event links yourself — date, time, and venue must match the official listing exactly. Every named person in the briefing must link to a page that names them; anyone else is "attendance unconfirmed". Then read the briefing's ice-breakers: if they could be said to anyone ("ask about their challenges"), send it back. A briefing is specific, or it is nothing.

Step 2.4 — Build a decision-maker dossier

Phase 2 of 4 · The Weekly Workflows — Step 2.4

(Also Stage 3.)

Goal. Know the key person before any meeting — from public sources only. *Disclosure for this class:* dossiers are built on real officials, from public sources, for classroom use only; if your work is ever shown outside the class, replace real people with fictional ones first (Prompt 3, Guardrail 5). (Dossier schema: Prompt 3.) Choosing the person: start from your strongest Tier 1 / Tier 2 agency, and pick its most senior relevant official — the bureau director or division head — find the name on the agency's organization page or leadership list: notices name the responsible department, not the director.

Prompt 8 — Dossier:

```
Build a dossier for [NAME], [TITLE] at [AGENCY], using the six-dimension
dossier schema. Public sources only: published bios, conference speaker
pages, association rosters, the agency's own news releases.
Write one short paragraph per dimension. Put the source next to every
factual claim. If a dimension cannot be filled from public sources,
write "No public source found" - do not fill it with guesses.
End with the ice-breaker pitch: exactly three sentences, specific to
this person's agenda, that I could say out loud in a first meeting.
Return the dossier, then the updated system file.
```

You should get back. Six short sections, every claim sourced, and an opener you could actually say.

Checkpoint 7. Pick any claim in the dossier and follow its source. If one claim has no source, the whole dossier goes back — this is the habit that keeps the system honest.

Your file so far: Dossiers has its first entry

Step 2.5 — Write the first Gold Sheet

Phase 2 of 4 · The Weekly Workflows — Step 2.5

(This is Stage 3 — Strategize — in the Technical Addendum.)

Goal. One priority account, understood on one page. (Gold Sheet schema: Prompt 3 — defined there; first written here.)

Do this. Pick the account with the clearest timing catalyst (an expiring contract or a new leader). (**Catalyst**: the event that makes *now* the right moment; Tier 2 records live and die by theirs.) Paste **Prompt 9**.

Response Mode — the account's posture right now, in one line each:

- **Growth** — expanding: new budget, new programs, new people. The best hunting.
- **Trouble** — something hurts: a failure, an audit, complaints, a leader under pressure. They need help now.
- **Even Keel** — stable and satisfied; no urgency. Keep the relationship warm and wait for a catalyst.
- **Overconfident** — "we are fine, we need nothing" — often blind to a problem that is coming. The hardest to move; bring outside evidence, not opinions.

Prompt 9 — Gold Sheet:

```
Create the Gold Sheet for [AGENCY NAME], using the Gold Sheet schema in
your instructions.
Research from public sources only: the agency's organization page, its
capital improvement plan and budget, council / assembly agendas from the
last 6 months, procurement history (who holds the current contracts, and
when they expire), and published bios of its leaders.
Fill every section. Where a Buying Influence is unknown, write UNKNOWN -
that is a Red Flag, not a gap to hide. For each known person, write the
Win-Result in one sentence: what that person personally gains.
Record window_opens and window_closes for the account's current
pursuit window, each with its source; if no source states them,
write "window unknown" - that is a Red Flag.
End with an action plan whose FIRST action eliminates the biggest
Red Flag.
Return the Gold Sheet first, then the updated system file.
```

You should get back. A one-page account brief with named people, honest UNKNOWNs, and an action list that starts with the biggest unknown.

Checkpoint 8. Read the Win-Results. If they describe *your services* ("we offer great modeling") instead of *the client's gain* ("the new director gets a visible win in her first 100 days"), send it back: *"Rewrite every Win-Result from the client's side of the table."*

Your file so far: Gold Sheets has its first entry

The four Buying Influences, in plain words — and in a Japanese city office. *Economic Buyer* (the role that can release the money): in a municipality nobody signs alone — budgets are approved by the assembly and executed by the bureau, so treat the bureau director and the budget/finance division together as the economic center of gravity. *User Buyer* (who lives with the result): the section chiefs and engineers who will use what gets delivered, every day. *Technical Buyer* (who can say no, but not yes): the reviewers who check qualifications, standards, and compliance. *Coach* (who explains how decisions really get made): an insider contact — in the public sector a coach shares

only public process knowledge (how evaluation works, when budgets are set), never non-public information. If a "coach" offers more than that, it is a Red Flag, not a gift.

Step 3.1 — Generate the dashboard

Phase 3 of 4 · The Deliverable — Step 3.1

(This is Stage 4 — Render — and Stage 5 — Publish — in the Technical Addendum.)

Goal. Turn the system file into the one file you will actually read every week.

Do this. Start a fresh task with the agent. Give it your current system file — upload it, or paste its full text. Then paste **Prompt 10**.

One memory, two copies. bd-system.md is the record: it is what you archive, and what next week starts from. The dashboard's localStorage is only a scratchpad for the copy open in your browser — it holds your log edits between visits. Before you close a cycle, copy your log entries into the Execution Log section of the system file and save the file. If the two ever disagree, the file wins.

Prompt 10 — Dashboard:

```
Build the dashboard as ONE self-contained HTML file, using the attached
system file (Markdown) as the only data source.
Requirements:
- Tailwind CSS and Font Awesome via CDN, and vanilla JavaScript.
 No frameworks, no build step.
- Modules, in this order:
   1. Header: title, a cycle date, and a KPI (key performance indicator) band (portals monitored,
      active Tier 1 items, Tier 2 targets, Gold Sheets, and the name of
  the account whose window includes today (JST), from its
  window_opens / window_closes - or "none open").
   2. A one-line strategy banner: our positioning in one sentence.
   3. Tier 1 table: Agency | ID & Scope | Next Date - the soonest
  of the four dates, labeled (e.g., "Bid submission: 2026-11-12
  17:00 JST") | Portal | Direct Link.
   4. Tier 2 table: Agency | Initiative | Catalyst & Timing | Lead
      Capability | Portal | Strategic Action | Source (link + date). (No deadline column.)
   5. Gold Sheets: an account selector; the selected sheet shown in
      full; a Print button that prints just that sheet.
   6. Dossiers, one card per person.
   7. Event list.
   8. Weekly execution log: three editable text areas (accomplished /
      next actions / observations).
- Save the log edits and the current state to browser localStorage
 under the fixed key bd-dashboard-v1 - never change the key, or
 last cycle's backup will not import.
- Add two buttons: "Backup JSON" (downloads the full state as a
 file) and "Import JSON" (restores it). The JSON shape is fixed:
 {"schema_version": 1, "cycle_date": "YYYY-MM-DD", "tier1": [],
 "tier2": [], "gold_sheets": [], "dossiers": [], "events": [],
 "log": {"accomplished": "", "next_actions": "",
 "observations": ""}} - Import accepts exactly this shape and
 rejects any other with a clear message.
- Clean, professional, light theme: background #faf7f0, ink
 #2c2924, one restrained red accent #b23a2a; cards #fffdf9 with
 border #e4dcc8; headings in a serif display face ("Source
 Serif 4", Georgia, serif), body in Inter (system sans fallback).
 All English. No external images.
Return the COMPLETE file, not a summary and not a diff.
```

You should get back. One HTML file. Download it. Open it. Click everything.

Checkpoint 9. Four tests: (1) edit the log, reload the page — your text must survive; (2) click Backup JSON — a file must download; (3) open the file on a different browser or machine — it must render

fully; (4) pick five dates or IDs on the dashboard and compare them with bd-system.md — any mismatch means regenerate, not patch. (Your log text lives in each browser's own localStorage; carry it across with Backup / Import JSON.)

How you iterate on the dashboard. Small, surgical requests — never "make it better." **Prompt 11** shows the pattern:

```
Change ONLY the following in the dashboard file:
[DESCRIBE THE ONE CHANGE - e.g., "In the Tier 1 table, rename the
Portal column to Source Portal."]
Keep everything else exactly as it is - same data, same layout,
same wording. Return the COMPLETE updated file, not a diff.
```

One change per request. If a change breaks something, you still have last week's file — which is why every cycle is archived, never overwritten.

Step 4.1 — Audit it, write the rules, and put it on schedule

Phase 4 of 4 · Trust & Automation — Step 4.1

(This is the human side of Stage 6 — the review gate.)

Goal. Prove to yourself that the dashboard tells the truth.

Do this. First, ask the agent to expose its own weak points — **Prompt 12**:

```
Audit your own work in the current state. List:
1. Every Tier 1 item whose deadline you did NOT see on the official
   posting page yourself.
2. Every dossier claim that has no named public source.
3. Every field you filled from memory or inference instead of a source.
4. Every Tier 2 row with no source - no link, no date, or no quoted
 line behind its timing.
Do not fix anything yet. Just list them, item by item.
```

Then do the human half: pick **10 items at random** from the dashboard and check each one against its official source yourself. Write down how many were wrong.

Checkpoint 10 (audit). Zero errors, or a written fix for every error found — and the fix goes into the standing instructions (Prompt 3), not just into this week's data. A lesson that is not written into the instructions will be re-learned the hard way next month.

Then write the operating rules into your system file — they become its twelfth and final section, and the same text closes the Final Prompt assembled below. In the builder's file they are the last section — four rules, and every audit feeds them: (1) external portals are read-only; (2) human-in-the-loop — the agent drafts and analyzes, you send and decide; (3) zero fabrication — a live solicitation has a real ID and a real posting, and everything else is labeled Pre-RFP; (4) privacy — public, professional capacity only, never a personal contact. Add the rules your own audit just earned: a lesson that lives in a chat is lost; a rule in the file is permanent. The Operating Rules section of bd-system.md is a working copy — the canonical list lives in your agent's instructions (the Final Prompt), and if the two ever differ, the instructions win.

Where the file lives. Give bd-system.md one fixed home and never move it — for example `/BD/bd-system.md` on Drive, with `/BD/archive/` beside it. The scheduled run reads that path, writes the updated file back to the same path, and drops dated copies (bd-system-YYYY-MM-DD.md,

dashboard-YYYY-MM-DD.html) into the archive. Any fixed folder your platform can read and write works the same way.

The machine's part. Set a recurring scheduled run on your platform (in the original: once a week, on a fixed schedule). Gemini and ChatGPT both offer scheduled tasks (look for "scheduled actions" / "Tasks"); if your platform or plan has none, run Prompt 13 by hand each week — the system works the same. The scheduled task carries **Prompt 13** — the whole cycle in one instruction:

```
Read the latest bd-system.md from [YOUR PATH - e.g., /BD/bd-system.md].
If it is missing or empty, STOP and say so. Do not scan from memory.
1. Scan the portal registry and the planning sources (Step 2.1 rules).
2. Verify and classify everything you found (Step 2.2 rules).
3. Update the events list, dossiers, and Gold Sheets affected
 by this week's changes.
4. Rebuild the dashboard from the updated system file (Step 3.1 spec).
5. Write the updated bd-system.md back to the same path, and save
 bd-system-YYYY-MM-DD.md and dashboard-YYYY-MM-DD.html to the
 archive folder [YOUR ARCHIVE PATH - e.g., /BD/archive/].
6. Email the dashboard to me - to my own address only.
```

Your part (15–30 minutes, same day every week). Open the dashboard and: (1) check what is new, what is closing, whose window opened; (2) decide pursue / watch / pass on each new item; (3) spot-check two items at their official sources; (4) confirm the Gold Sheet action list; (5) type the week's outcomes into the execution log — in the dashboard, then copy them into the system file's Execution Log section before you save it (Step 3.1: the file wins). That log is next week's baseline — the loop is closed by you, not by the machine.

Manual version. No Drive, or no scheduler? Run the same cycle by hand: give the agent last week's system file yourself (upload it, or paste its full text), work steps 1–4, and save the outputs under the same dated names. The rhythm matters, not the automation.

Checkpoint 10. Dry run first: run Prompt 13 once by hand on a one-agency registry and confirm the file was read, updated, and archived. Platforms may pause for approval before actions such as sending email, so the first scheduled run may be waiting for you — check on it. Two consecutive weekly cycles ran on schedule and your review took under 30 minutes each. You now own a running system. What you finish with: (1) the Final Prompt, installed as your agent's standing instructions; (2) a bd-system.md whose Execution Log holds two weekly cycles; (3) the dashboard those cycles rendered.

Your file so far: complete — all twelve sections, closed by the Operating Rules

Assemble the Final Prompt. Prompts 1–13 were scaffolding — each one taught a single piece. The product of this build is one document: your master instructions, complete. It is Prompt 3, grown to full size — your deployment filled in from your own Steps 1.2–1.3, the schemas proven at Step 1.4, and the Operating Rules your audit earned, written into its final section. Paste it as your agent's standing instructions. From here on the system runs on this one prompt; everything before was just how you built it.

The Final Prompt — complete master instructions:

```
THE FINAL PROMPT — COMPLETE MASTER INSTRUCTIONS

ROLE
You are the business-development (BD) intelligence analyst for an engineering
consulting practice that sells professional services to municipal governments.
You run a weekly intelligence cycle and maintain a single dashboard.
```

```
METHODOLOGY (Miller Heiman)
- Every priority account is managed on a one-page Gold Sheet (account level,
  1-3 year horizon).
- For every account, identify the four Buying Influences: Economic Buyer
  (approves the money), User Buyer (lives with the result), Technical Buyer
  (can veto, cannot approve), Coach (an insider who teaches us how to win).
- Assess each account's Response Mode: Growth (expanding),
  Trouble (hurting), Even Keel (stable, no urgency), or
  Overconfident (complacent).
- Every unknown or risk is a Red Flag. Surface Red Flags explicitly; never
  hide them.
- Every recommendation is expressed as a Win-Result: what the specific client
  person gains.

DATA SCHEMAS (never change these without my instruction)
Tier 1 - Verified Solicitation (defined here): agency, solicitation_id, scope,
 application_deadline, question_deadline, submission_deadline,
 opening_date (each: date + time, JST, and the original Japanese
 quote; "not stated" when the notice omits it), portal,
 direct_link, fit (core / adjacent / out), status.
Tier 2 - Pre-RFP Target (defined here): agency, initiative, catalyst_and_timing
 (a timing window only if the source states or implies one;
 otherwise "timing unknown" - never a single release date),
 lead_capability, portal_base, strategic_action, source_url,
 source_date, source_quote_original (the Japanese line that
 supports the timing).
  (There is NO deadline field in Tier 2. Never add one.)
Gold Sheet (defined here): account_overview, response_mode, buying_influences (for each:
  name, role, stance (supportive / neutral / opposed /
  unknown), win_result), competitive_landscape, vulnerabilities,
  strategy, win_results, action_plan, red_flags, last_updated, window_opens, window_closes
 (dates, each with a source; "window unknown" if no source
 states them).
Dossier (defined here): name, title, agency, buying_influence, plus six dimensions:
  (1) responsibility & role, (2) background & pedigree,
  (3) core similarities, (4) conferences & gatherings,
  (5) high-frequency keywords, (6) ice-breaker pitch.
  Every factual claim must name its public source.
Event (defined here): name, date, time, venue, target_stakeholders
 (expected roles, not confirmed attendees),
  registration_link, source.

DEPLOYMENT - GREATER TOKYO (this build's territory)
<<REPLACE: this block with your own territory, agencies and
portals, from your Steps 1.2-1.3; then delete this marker. The
Greater Tokyo text below is the classroom default.>>
- Territory: Tokyo Metropolis (23 Special Wards + Tama), Kanagawa
  (Yokohama, Kawasaki), Saitama, Chiba. Nothing outside it.
- Target agencies: TMG Bureau of Waterworks, TMG Bureau of Sewerage,
  Setagaya City, Yokohama City Waterworks Bureau, Kawasaki City,
  Hachioji City, MLIT Kanto Regional Development Bureau.
- Official portals (verify here, nowhere else): Tokyo e-Procurement,
  e-Tokyo Joint, MLIT PPI, Yokohama e-Bidding.
- Events: JSCE (Japan Society of Civil Engineers), JWWA (Japan
  Water Works Association), JSWA (Japan Sewage Works
  Association) and MLIT forums - within a 45-minute
  transit radius of our office (origin station:
 [YOUR OFFICE STATION]), inside a rolling 14-day window.

GUARDRAILS (hard rules)
1. Tier 1 requires a confirmed ID, its dates verified on the official posting page (every date the
notice states), and a direct link. If you cannot verify, the item goes to
   the Red Flag list. Never guess a date, an ID, or a name.
2. Public information only. Never invent people, postings, or figures.
3. You never contact anyone. You draft; I send.
 Exception: the weekly digest email to my own address.
4. Once a solicitation is posted, no positioning contact about it
   (the "cone of silence" - once a tender is public, vendors must not
```

```
   lobby about it; all communication goes through the official
   channel). The positioning work happens before posting, in Tier 2.
5. Anything shared outside the practice uses fictional data, labeled DEMO.

WEEKLY CYCLE
Scan -> Verify & classify -> Events & briefings ->
Update dossiers & Gold Sheets -> Render dashboard ->
Archive & send to me -> I review and approve.
Scan window: deadlines up to 45-60 days out; farther deadlines enter the window in later cycles.

OUTPUT DISCIPLINE
After every task, return (a) a short human summary of what changed,
(b) the complete updated system file (Markdown) - one downloadable
.md file or one code block - its sections following the schemas
above, and (c) a CHANGELOG listing every row or section added,
changed or removed. Do not change any row you were not asked
to change. If anything is
uncertain, say so in the summary and add a Red Flag.

OPERATING RULES (the last section - completed at Step 4.1)
1. External portals are read-only: the agent never submits,
   posts, or contacts anyone.
 Exception: the weekly digest email to the operator's own
 address.
2. Human-in-the-loop: the agent drafts and analyzes; the
   operator sends and decides.
3. Zero fabrication: a live solicitation has a real ID and a
   real posting; everything else is labeled Pre-RFP planning
   intelligence.
4. Privacy: public, professional capacity only; never a
   personal contact.
<<REPLACE: add every rule your own audits earned, in your own
words; then delete this marker. This section is where the
system keeps its scars.>>
```

The file you built — the whole architecture, now that it is yours

Twelve sections, each added by the step that creates its data. You never had to learn this architecture in advance, because you assembled it: **Summary** — what the practice is · **Territory** — where you play, and where you do not · **Team & Positioning** — why you win · **Portal Registry** — where opportunities publish · **Tier 1** — verified solicitations · **Tier 2** — pre-RFP targets · **Events** — the 14-day radar · **Dossiers** — the people · **Gold Sheets** — the accounts · **Red Flags** — what is not yet proven · **Execution Log** — what each week changed · **Operating Rules** — the boundaries, scars included.

Part 3 — The Prompts: How They Were Written, and How They Grew

Students always ask for "the prompts." Here is the honest answer about how the original system was actually prompted, because the process matters more than any single text.

There was no prompt library. The builder worked conversationally, the way you just did in Steps 1.2–4.1: one task at a time, in plain language. What *was* written down, once, was the knowledge the agent had to carry — a **knowledge pack**: the Miller Heiman methodology (how a Gold Sheet works, the four Buying Influences, Red Flags, Win-Results), a research checklist (where to look for each fact), and a short list of **behavior rules** (public sources only, never contact anyone, unknowns are Red Flags, first action always attacks the biggest Red Flag). That pack is what Prompt 3 — the master

instructions you install at Step 1.4 — is distilled from. The per-task prompts were then short — because the agent already "knew the job."

Why the rules exist at all: hallucination. A language model will sometimes state a fluent, confident, completely invented fact — a deadline, a name, a figure — because producing plausible text is what it does. It is not lying; it has no way to know. Every guardrail, tier, and checkpoint in this guide exists to catch that one behavior.

The iteration pattern. Every durable rule in the system is a scar from a real failure. The verification prompt shows the pattern in three versions:

Version 1 — the plain ask.

```
Find this week's RFQs in my market and list them with deadlines.
```

What came back looked good and was not: deadlines copied from aggregator sites (sometimes wrong), early rumors listed next to live postings, and — the dangerous one — plausible dates the agent had filled in where a portal showed none.

Version 2 — add the source rule and the shape of the answer.

```
...verify ID, deadline, and link on the OFFICIAL posting page only.
Return exactly these fields: agency, solicitation_id, scope,
verified_deadline, portal, direct_link, fit...
```

Better. But now, when the agent could not find a deadline, it either skipped the item silently or strained to produce one — because the prompt gave it no honest way out.

Version 3 — add the escape hatch (this is Prompt 6 in Step 2.2).

```
...if you cannot confirm it on the official page, the item goes to the
Red Flag list, with the reason. Early signals go to Tier 2 - which has
no deadline field...
```

Error rates fell off a cliff — not because the model got smarter, but because **"I cannot verify this" became a valid, rewarded answer**, and the data model made the worst mistake (a made-up deadline in Tier 2) impossible to express.

The five prompt rules the builder ended up following. Plain version:

1. **One job per prompt.** Scan *or* verify *or* write — never all three in one breath.
2. **Say what the answer must look like.** Field names, order, format. If you do not define the shape, you get essays.
3. **Say the forbidden things out loud.** "Never guess a date" only works because it is written down.
4. **Give one good example.** One correct input → output pair teaches more than a page of explanation — Prompt 4's fictional examples are this rule at work.
5. **Always leave an honest way out.** "If you can't verify it, say so, and say why" is the single highest-value sentence in any agent prompt.

And the meta-rule: **when the agent fails twice the same way, the fix goes into the standing instructions (Prompt 3), not into your next message.** Prompts are disposable; instructions are the product.

A note on sources, for the classroom: the original builder's day-to-day prompts were given to the agent in conversation and refined in place; they were never exported as a document. The prompts printed in this guide are faithful reconstructions — same schemas, same rules, same order of operations, drawn from the system's documented design (the playbook, the case study, and the builder's knowledge pack). If you build the system with them, you are running the same process; you are not reading a transcript.

Part 4 — Troubleshooting

Symptom	Likely cause	Fix
Verified rows changed or vanished between cycles	Full-file regeneration — the agent rewrote the whole file and quietly altered rows it was not asked to touch	Require the CHANGELOG (Prompt 3, OUTPUT DISCIPLINE) and diff every cycle's file against the last before accepting it
The agent invents a deadline	No escape hatch in the prompt	Add the Red Flag rule (Prompt 6, Version 3) and repeat the test in Checkpoint 3 (instructions)
Answers come back as essays	No output schema	Restate the field list; end the prompt with "Return only the tables and the updated system file"
A portal scan returns nothing	The portal is a JavaScript app your agent read as raw HTML	Use the platform's browser/agent mode; if it still fails, log the portal as "unreadable" and check it by hand weekly
The Gold Sheet sounds generic	The agent had no real inputs	Feed it the CIP excerpt, the agenda items, and the contract-expiry dates as context, then regenerate
The dashboard loses the log edits	localStorage key changed between versions	Keep one fixed storage key across versions; use Backup JSON at the end of every cycle
The agent "fixes" things you did not ask about	Vague change requests	Use the Prompt 11 pattern: name the single change, demand everything else stay identical
The dashboard file comes back cut in half	The answer hit a length limit	Ask for the complete file again; if it recurs, request it in two parts (markup first, script second) and join them
The agent answers from memory instead of browsing	It took the shortcut	Demand the direct link and the "unreadable sources" list for every claim; the Step 4.1 audit (Prompt 12) catches what slips through

Appendix — The Weekly Run Card (tape this to the wall)

On run day, at the scheduled time — the scheduled run executes Prompt 13: scan → verify → update → render → archive → email.

Right after the run — your 15–30 minutes:

1. What is new? What is closing? Whose window opened?
2. Pursue / watch / pass — decide each new item.
3. Spot-check two items at the official source.
4. Confirm the Gold Sheet actions; the first one attacks the biggest Red Flag.
5. Write the week's outcomes in the execution log. Click Backup JSON.

Once a month: the 10-item audit (Step 4.1). Zero errors, or a written fix — in the instructions, not just in the data.

Appendix — Words You Will Meet (reference)

The full list, for reference. The seven terms you need before Step 1.1 are boxed just before Part 1; look the rest up as you meet them.

Solicitation / tender / bid: a public request for proposals or prices — the thing you are hunting. *RFQ vs RFP:* an RFQ asks "are you qualified?"; an RFP asks "what exactly would you do, and for how much?" *Council / assembly agenda:* the published plan for a government meeting — what will be discussed, approved, and spent on; in Japan, the assembly's and committees' meeting materials. *Capital plan:* the multi-year list of construction and renewal projects a city intends to fund. *Pre-positioning:* becoming known and trusted *before* a tender is published — the only legitimate time to build the relationship. *Cone of silence:* after a tender is published, vendors must stop lobbying about it; all contact goes through the official channel.