**GetGSA: AI + RAG**

**Goal**

Ship a tiny, working slice of **GetGSA** that can:

1. **Ingest and classify** common GSA onboarding docs (as raw text),
2. **Extract** key fields and detect **missing** ones,
3. Build a **policy-aware checklist** using a small, provided **GSA Rules Pack** (RAG),
4. **Redact PII** (minimum: emails and phone numbers) in stored copies,
5. Generate a **negotiation prep brief** and a **client email** summarizing gaps,
6. Provide a **single-page UI** and a minimal **API**,
7. Include a **repeatable test harness** with sample inputs and expected outputs.

You must use **AI** somewhere essential (classification and/or the RAG checklist + brief generation). If you can’t call an external API, you may run a local model or mock LLM calls behind a well-defined interface but must show the prompts + reasoning strategy, and fall back to classical methods when the model abstains.

**What we give you (mini knowledge base)**

Use these five short “policy” snippets as your **GSA Rules Pack** (treat them like tiny source docs you’ll index and retrieve from):

**R1 – Identity & Registry:**

* Required: UEI (12 chars), DUNS (9 digits), and active SAM.gov registration.
* Primary contact must have valid email and phone.

**R2 – NAICS & SIN Mapping (subset for test):**

* 541511 → 54151S
* 541512 → 54151S
* 541611 → 541611
* 518210 → 518210C

**R3 – Past Performance:**

* At least **1** past performance ≥ **$25,000** within last **36 months**.
* Must include customer name, value, period, and contact email.

**R4 – Pricing & Catalog (starter rules):**

* Provide labor categories and rates in a structured sheet.
* If missing rate basis or units, flag “pricing\_incomplete”.

**R5 – Submission Hygiene:**

* All personally identifiable info must be stored in redacted form; only derived fields and hashes are stored by default.

You must turn these into a small **vector store / index** and do retrieval for your checklist & brief. Cite which rule(s) you used by ID (R1…R5).

**Sample inputs (the “customer dropped files”)**

Use these raw text samples to simulate parsed PDFs/DOCX. Some are intentionally incomplete:

**Company Profile (A):**

Acme Robotics LLC

UEI: ABC123DEF456

DUNS: 123456789

NAICS: 541511, 541512

POC: Jane Smith, jane@acme.co, (415) 555-0100

Address: 444 West Lake Street, Suite 1700, Chicago, IL 60606

SAM.gov: registered

**Past Performance (PP-1):**

Customer: City of Palo Verde

Contract: Website modernization

Value: $18,000

Period: 07/2023 - 03/2024

Contact: John Roe, cio@pverde.gov

**Past Performance (PP-2):**

Customer: State of Fremont

Contract: Data migration & support

Value: $82,500

Period: 10/2022 - 02/2024

Contact: sarah.lee@fremont.gov

**Pricing Sheet (text; simplified):**

Labor Category, Rate, Unit

Senior Developer, 185, Hour

Project Manager, 165, Hour

You may add your own edge-case docs (e.g., missing DUNS, invalid email, PP value commas, etc.) to demonstrate robustness.

**What to build**

**1) Backend (any stack)**

**Endpoints** (minimum):

* POST /ingest  
  Body: { documents: [{name, type\_hint, text}] }
  + type\_hint is optional (“profile”, “past\_performance”, “pricing”)
  + Store redacted copies (mask emails + phones)  
    Returns: { doc\_summaries: [...], request\_id }
* POST /analyze  
  Body: none (operates on last ingested set or by request\_id)
  + **Classify** each document (LLM encouraged; if uncertain → “unknown” with **abstain** signal).
  + **Extract** fields (UEI, DUNS, POC email/phone, NAICS[], SAM status; PP customer/value/period/email; pricing rows).
  + **RAG**: Build a **policy-aware checklist** citing **R#** evidence, including:
    - required.ok boolean
    - problems[] like missing\_uei, sam\_not\_active, past\_performance\_min\_value\_not\_met, pricing\_incomplete, etc., with **evidence** and **rule IDs**
  + **Generate** two text artifacts:
    - **Negotiation Prep Brief** (2–3 paragraphs): what’s strong/weak and where to push on pricing/completeness (cite rules R#).
    - **Client Email Draft** (polite, concise): list missing items and next steps.  
      Returns: { parsed, checklist, brief, client\_email, citations: [{rule\_id, chunk}], request\_id }
* GET /healthz → {"ok": true}

**AI requirements**

* Use an LLM (cloud or local) for at least one of: **doc classification**, **RAG checklist reasoning**, **brief/email generation**.
* Implement **abstention**: when confidence is low, the model should return “IDK/needs human” (e.g., no hallucinated fields).
* Show your **prompt(s)** and short reasoning approach in the repo.

**RAG requirements**

* Build a tiny vector index from **R1–R5**.
* Retrieval must drive the checklist & brief (attach rule IDs in responses).

**PII redaction**

* Redact emails and phone numbers before storage; keep parsed fields separately.

**2) UI (single page)**

* Two panes: **Input** (paste texts or select sample docs) and **Results**.
* Buttons: **Ingest**, **Analyze**.
* Render: checklist (with rule citations), parsed fields, brief, client email.
* Simple, clean; no need for design polish.

**3) Tests & eval harness**

* **Unit tests (≥4)** covering:
  1. Missing UEI → missing\_uei flagged (R1)
  2. Past performance threshold (PP-1 at $18,000) → past\_performance\_min\_value\_not\_met (R3)
  3. Proper NAICS→SIN mapping with dedupe (R2)
  4. PII redaction masks emails/phones on stored docs (R5)
* **RAG sanity test (1)**: when R1 is removed from the index, checklist should fail to cite it (or abstain appropriately).
* Provide a **make test** or one-command runner.

**Deliverables**

* Running API & UI (README with commands).
* A short **ARCHITECTURE.md**:
  + 1–2 diagrams (ASCII ok): data flow, components, where AI is used.
  + Why your design can scale from 10 → 1,000 customers (queues, stateless API, storage choices).
  + How you’d add “Pricing Pack v2” later without breaking things.
* **PROMPTS.md**: prompts, guardrails, and abstention policy.
* **SECURITY.md**: redaction approach, max input size, basic abuse limits.
* Tests passing locally.