**GetGSA Coding Test:**

GetGSA Coding Exercise (45–60 min) — Instructions

This test simulates the kind of ambiguity you’ll face as our first full-stack AI engineer. You’ll need to figure things out quickly and ship a vertical slice of product without a PRD.

**Timebox**

Please spend **45–60 minutes**. We’re looking at how you structure code, make decisions under constraints, and deliver value — not a polished product.

**Task**

Build a small service that:

1. Accepts two text inputs:
   * **Company Profile** (pseudo-PDF text block)
   * **Past Performance** (pseudo-DOCX text block)
2. Parses them into structured JSON (company details, NAICS, past performance).
3. Validates fields and flags missing/invalid items.
4. Maps NAICS → recommended SINs (using the table below).
5. Builds a **checklist** showing if required conditions are satisfied.
6. Exposes a POST /ingest API endpoint returning { parsed, issues, recommended\_sins, checklist, request\_id }.
7. **Bonus (not required):** One-page UI (two text areas + button) calling your API and rendering JSON response.

**Sample Input Texts**

**Company Profile**

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**

Customer: City of Palo Verde

Contract: Website modernization

Value: $180,000

Period: 07/2023 - 03/2024

Contact: John Roe, cio@pverde.gov

**NAICS → SIN Mapping**

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

**Requirements**

* **Language/Framework:** Any you prefer.
* **Core:** API, parsing, validation, checklist.
* **Tests:** At least 3 unit tests proving your parser catches missing UEI, bad email, and NAICS→SIN mapping.
* **Audit:** Log request id + validations run + outcome (stdout/file).
* **Deliverables:**
  + Running API
  + Test suite (single command to run)
  + README with run/test instructions
  + Simple UI

**Acceptance Criteria**

* Parses correctly (company\_name, UEI, DUNS, NAICS[], POC, address, SAM, past performance).
* Flags missing/invalid fields (missing\_uei, invalid\_poc\_email, etc.).
* Correct NAICS→SIN mapping with no duplicates.
* Checklist logic accurate (required.ok true/false).
* Tests pass locally.
* Clean, modular code with basic logging.

**Submission**

* Share a **Git repo link** with your solution.
* Include your **README** with clear instructions.
* Ensure tests run successfully (pytest or your runner).
* Frequent commits are appreciated — we’ll review history.

**Evaluation Rubric**

We score across:

* **Correctness (35 pts)** – parsing, validation, mapping, checklist.
* **Code quality (25 pts)** – structure, naming, logging.
* **Tests (15 pts)** – ≥3, cover negatives, deterministic.
* **Robustness (10 pts)** – sanitization, safe parsing.
* **Product thinking (10 pts)** – JSON structure, checklist usefulness.
* **Bonus (5 pts)** – UI, Dockerfile, extra tests.

**Clarifications**

* No external LLM/API calls are required. Simulate parsing with regex/string ops.
* Focus on delivering a working vertical slice, not completeness.
* If you run out of time, commit what you have — partial solutions are better than none.

Good luck! We want to see how you think, not perfection.

Thanks!