Coding Challenge

Objective

Build a small FHIR server for the `Patient` resource that showcases your full-stack skills: a HTTP service, PostgreSQL with a custom extension, FHIR REST API with search and pagination.

Requirements

Functional scope

- Implement a minimal FHIR R4 compliant API for `Patient` with:
  - `POST /fhir/Patient` : create a new Patient, assign an ID, persist via the Postgres extension, and return metadata headers.
  - `GET /fhir/Patient/{id}` : fetch by ID.
  - `GET /fhir/Patient?name=...&birthdate=...&gender=...&_count=...&_offset=...` : basic search with pagination.

Database & extension

- Persist and retrieve resources through the Postgres extension, store payloads as `jsonb`, and add any helper tables/indexes you need for search.

```
-- Example interface shape (you may adapt names/signatures)
SELECT fhir_put('Patient', $jsonb)          -- → uuid (new id)
SELECT fhir_get('Patient', $uuid)           -- → jsonb
SELECT fhir_search('Patient', $param, $op, $value)  -- → setof uuid
```

- You may explore PGRX as a way to build Postgres extensions ergonomically.

Nice-to-have (optional)

- History: `GET /fhir/Patient/{id}/_history`.
- Richer `OperationOutcome` details (error codes, locations).

Acceptance criteria

- Resources are persisted and queried via the extension, not direct ad-hoc SQL against app tables.
- Search supports name substring, birthdate ranges (`ge` / `le`), and gender; pagination is stable.
- Tests are automated and reproducible.

Deliverables

- Git repository (public or zipped) containing:
  - `server/` : FHIR REST API web service in the language/framework of your choice.
  - `db/` : Postgres extension.
  - README with:
    - Project overview
    - Exact run/test commands
    - API overview (paths/params, expected headers, sample requests/responses)