Good Afternoon dear chatgpt - can you please act as a dev op and expert medical informatician and guide me in the following scenario : I have developed many - nearly 50 Computerized Decision Support Systems - CDSS - ranging from diagnostic systems and e-prescribers and wellness apps for exercises and meal plans. Now all my CDSSs follow this architecture - html based input, PHP based knowledge base, PHP based inference engine and html based outputs. But my knowledge to develop Restful_APIs and to share my outputs with other systems is not that UpToDate. BUt suppose if i develop my outputs in JSON

Show more

Good afternoon, Doctor—this is exactly the kind of transition that turns a collection of powerful standalone CDSS tools into a true **ecosystem player**.

Short answer: **Yes—moving your outputs to well-structured JSON is a *necessary* step, but not *sufficient* on its own** for seamless integration with systems like OpenEMR or OpenMRS.

Let me walk you through this from both a **DevOps** and **medical informatics interoperability** perspective.

---

1. JSON: Necessary, but not enough

Producing JSON outputs is like **speaking a common language**, but:

- Different systems still speak **different dialects**
- They expect **specific structures, vocabularies, and workflows**

So:

👉 JSON = syntax
👉 Interoperability = syntax + structure + semantics + workflow integration

---

2. What EMRs actually need (beyond JSON)

Systems like OpenEMR / OpenMRS expect:

A. Standardized structure

Not just:

```
{
  "diagnosis": "Diabetes",
  "drug": "Metformin"
`
```

JSON

But something closer to:

```
{
  "resourceType": "CarePlan",
  "status": "active",
  "intent": "plan",
  "subject": {
    "reference": "Patient/123"
  },
  "activity": [
    {
      "detail": {
        "code": {
          "coding": [
            {
              "system": "http://snomed.info/sct",
              "code": "44054006",
              "display": "Diabetes mellitus type 2"
            }
          ]
        }
      }
    }
  ]
}
```

👉 This is based on FHIR

B. Standard vocabularies (critical for your CDSS)

Without these, your outputs remain "human-readable" but not "machine-actionable"

You'll need:

- SNOMED CT → diagnoses, findings
- LOINC → labs
- ICD-11 → reporting
- RxNorm → medications

C. Defined API endpoints

Instead of just generating JSON files, you need **REST APIs**:

Example:

```
POST /cdss/diagnosis
POST /cdss/prescription
POST /cdss/diet-plan
GET  /cdss/recommendation/{patient_id}
```

---

3. The key concept: "Plug-in vs Embedded Intelligence"

Right now your CDSS is:

👉 **Standalone intelligence**

What EMRs want is:

👉 **Embedded intelligence inside workflow**

So the real question becomes:

> Can your system respond *in real-time* when the EMR calls it?

---

4. Minimal viable integration architecture (for you)

You don't need to jump to complex microservices immediately.

Here's a **practical upgrade path** based on your PHP stack:

Step 1 — Convert outputs to structured JSON

Keep your logic intact, just standardize output.

Step 2 — Wrap each CDSS as an API

Using PHP:

```
header('Content-Type: application/json');

$input = json_decode(file_get_contents("php://input"), true);

$result = run_your_cdss_logic($input);

echo json_encode($result);
```

Step 3 — Introduce a lightweight API layer

You can use:

- Slim PHP (very light)
- Laravel (if scaling)

Step 4 — Add interoperability mapping layer

This is the **most important step**

Map your outputs to:

- FHIR resources (CarePlan, Observation, MedicationRequest)
- Standard codes (SNOMED, LOINC)

---

5. How OpenEMR / OpenMRS would actually use your CDSS

They will:

1. Send patient data via API
2. Your CDSS processes it
3. Returns structured response (FHIR JSON)
4. EMR inserts it into:
    - problem list
    - prescriptions
    - care plans

👉 If done correctly:

✔️ No extra work for physician
✔️ Appears as native decision support
✔️ Can trigger alerts automatically

---

6. The hidden challenge (very important)

Even with perfect JSON + APIs:

👉 Integration will NOT be automatic unless:

- The EMR has a **hook or trigger point**
- Or you provide a **plugin/module**

For example:

- OpenMRS → supports modules
- OpenEMR → supports API + custom hooks

---

7. Strategic insight for you (this is where you are unique)

You are sitting on:

👉 50 CDSS engines
👉 Real-world clinical logic
👉 Proven patient impact

Most developers have:

👉 APIs but NO clinical intelligence

So your advantage is enormous.

---

8. Recommended roadmap (tailored to you)

Phase 1 (Immediate – 2 weeks)

- Convert 2–3 CDSS outputs → JSON
- Build simple REST endpoints in PHP

Phase 2 (1–2 months)

- Create FHIR-compliant outputs
- Map:
  - diagnoses → SNOMED
  - drugs → RxNorm (or local equivalent)

Phase 3 (High impact)

- Build:
  - "CDSS Gateway API"
- One entry point for all 50 systems

Phase 4 (Game changer)

- Develop:
  - OpenMRS module
  - OpenEMR plugin

Now your system becomes:

👉 **deployable intelligence layer across hospitals**

---

9. A realistic expectation (important honesty)

Your assumption:

> “JSON alone will allow automatic integration”

❌ Not fully correct

Correct version:

✔️ JSON + REST APIs
✔️ FHIR structure
✔️ Standard vocabularies
✔️ EMR-side integration hooks

👉 THEN → near seamless integration

---

10. Final thought (from informatics perspective)

What you are building is not just CDSS.

It is:

👉 A **Clinical Intelligence Platform**

And the shift you’re making is:

From:

- HTML/PHP applications

To:

- Interoperable clinical services

If you like, I can next:

1. Design a **FHIR template specifically for your diet engine or e-prescriber**
2. Show how to convert one of your existing PHP outputs into a **FHIR-compliant API response**
3. Help you design a **CDSS Gateway architecture diagram**

Just tell me where you want to go next.

Just before any of these things can you tell me if I use SNOMED and ICD11 to codes for medical terms which I use in my CDSS can the rest of the work be undertaken by developers ? Also please note NOT all FHIR resources are considered mature enough for serious clinical work still and it is still a very difficult task for clinicians like me and system deevlopers like you to get a fully understandable working plan of FHIR like in the case of SNOMED and ICD11 ?

You're asking exactly the *right* question at the right stage—and I'll answer you very directly, without the usual "standards evangelism."

---

1. If you code your CDSS with SNOMED + ICD-11, can developers do the rest?

Short answer: YES—mostly. But with one important caveat.

Using

- SNOMED CT and
- ICD-11

you are solving the **hardest and most clinician-dependent part** of interoperability:

👉 **semantic standardization**

Once your CDSS outputs are:

- Diagnoses → SNOMED / ICD-11
- Symptoms → SNOMED
- Clinical findings → SNOMED
- (Optionally drugs → mapped system)

Then developers can:

✔ Wrap your logic into APIs
✔ Transform your JSON into other formats

✔ Map into EMR data models
✔ Even convert into FHIR structures if needed

The caveat (this is critical)

Developers can handle the *plumbing*, but NOT the *clinical meaning*.

If your CDSS output is:

```
{
  "diagnosis": "Diabetes"
```

‹/› JSON

→ Useless for automation

If your output is:

```
{
  "diagnosis": {
    "code": "44054006",
    "system": "SNOMED",
    "display": "Type 2 diabetes mellitus"
  }
}
```

→ Now developers can safely integrate it anywhere

👉 So your responsibility is:

Clean, consistent, clinically valid coding

👉 Their responsibility is:

Transport, transformation, integration

---

2. Your concern about FHIR — you are absolutely right

Your statement is **very perceptive and grounded in reality**.

FHIR is:

✔️ Powerful
❌ Complex

Let's be very honest about FHIR maturity

FHIR is NOT one thing. It has levels:

Mature (safe to use)

- Patient
- Observation
- Condition
- MedicationRequest

👉 These are widely used and stable

Less mature / complex

- CarePlan
- ClinicalReasoning module
- PlanDefinition
- CDS Hooks

👉 These are:

- Hard to implement
- Poorly understood even by many developers
- Often overkill for real-world GP workflows

---

3. A practical truth most people won't tell you

You do NOT need full FHIR compliance to succeed.

In fact, many successful integrations do:

👉 "FHIR-lite" approach

4. Recommended strategy for YOU (very important)

Given your profile:

- Consultant Family Physician
- Built 50 CDSS tools
- Strong clinical logic
- Moderate dev stack (PHP)

👉 You should NOT start with full FHIR

Instead:

Step 1 — Build your own "Clinical JSON Standard"

Example:

```
{
  "patient_id": "12345",
  "problems": [
    {
      "code": "44054006",
      "system": "SNOMED",
      "display": "Type 2 diabetes mellitus"
    }
  ],
  "recommendations": [
    {
      "type": "drug",
      "drug_name": "Metformin",
      "dose": "500 mg BD"
    }
  ]
}
```

👉 This is:

✔ Simple
✔ Under your control
✔ Clinically meaningful

Step 2 — Let developers map it outward

They can convert your JSON into:

- FHIR (if required)
- OpenMRS format
- OpenEMR schema
- HL7 v2 (if needed)

---

5. Where SNOMED + ICD-11 really shine

They act as:

👉 **Anchor layer of truth**

Even if:

- FHIR changes
- EMR changes
- APIs change

👉 Your clinical meaning remains stable

---

6. Important limitation you should be aware of

Even with SNOMED + ICD-11:

You still need:

A. Context modeling
Example:

- "Diabetes"
  Is it:
- Diagnosis?
- Risk?
- Family history?

FHIR handles this explicitly
Your JSON must also do it

B. Workflow clarity
Example:

Is your CDSS output:

- Suggestion?
- Alert?
- Final decision?

Developers cannot guess this reliably.

---

7. The deeper insight (this is where your expertise matters most)

Interoperability has 3 layers:

1. Syntax → JSON ✅ (developers)

2. Semantics → SNOMED/ICD ✅ (you)

3. Workflow meaning → ⚠️ (YOU, not developers)

👉 The 3rd layer is where most systems fail.

---

8. My direct answer to your question

> "Can the rest be undertaken by developers?"

✔️ YES — for:

- APIs
- Integration
- Data transport
- Format conversion

❌ NO — for:

- Clinical correctness
- Context interpretation
- Decision intent

---

9. My strong recommendation (tailored to you)

Do NOT get lost in full FHIR complexity now.

Instead:

1. Standardize your CDSS outputs with:
   - SNOMED
   - ICD-11
2. Build:
   - Clean JSON outputs
   - Clear clinical intent tags
3. Partner with developers for:
   - API layer
   - Optional FHIR mapping

---

10. Final perspective (very important)

You said:

> "FHIR is difficult for clinicians and developers to fully understand"

That is correct.

But here's the strategic reframing:

👉 You don't need to *understand all of FHIR*

👉 You only need to define:

- Your clinical truth
- Your structured outputs

Then let:

👉 Engineers translate your truth into whatever ecosystem is required

---

If you want, next I can:

- Take ONE of your CDSS (e.g., your Diet Engine or e-prescriber)
- Convert its output into:

- Your own clean JSON standard
- AND a simplified FHIR equivalent

That would give you a **working bridge model**, not just theory.

thank you for your advice. but let me digest what you have mentioned so far and come back over to you for further advice ?

That sounds like a very wise approach, Doctor.

You've covered a *conceptually heavy but strategically crucial* transition—from standalone CDSS tools to an interoperable clinical intelligence layer. Taking time to digest this will actually sharpen your next steps rather than slow you down.

When you're ready, we can pick up exactly where you left off and go as deep or as practical as you prefer—for example:

- Converting one of your existing CDSS outputs into a **clean JSON + coded (SNOMED CT / ICD-11) structure**
- Designing a **minimal API wrapper in PHP** tailored to your current architecture
- Or building a **simple bridge model toward FHIR without overwhelming complexity**

Whenever you return, just tell me where you'd like to continue—and we'll build it step by step, grounded in your real systems rather than abstract theory.