Healthcare IT Developer Training Program

Empowering the Next Generation of FHIR-Enabled Full Stack Developers

LEARNERS Digital

“The more you sweat in training, the less you bleed in battle.”

About Me

Devesh Prakash Singh
Lead Soft Engineer at Onyx
Education- MCA(Central University) + PGDAC(CDAC-Bangalore)
active member of “HL7 India FHIR community”
Project worked on data ingestion for claims & clinical
Building the secure API on top of FHIR server – Provider Access API
Mandate – CMS-9115 & CMS-0057

This Far

- Python OOPS – inheritance (DRY)
- Database – data modelling, ER diagram
- Soft engineering – Product ?? ----→ PPL GET

- Data engineering – data processing, ETL
- UI & UX – user interface

What NEXT ??FHIR

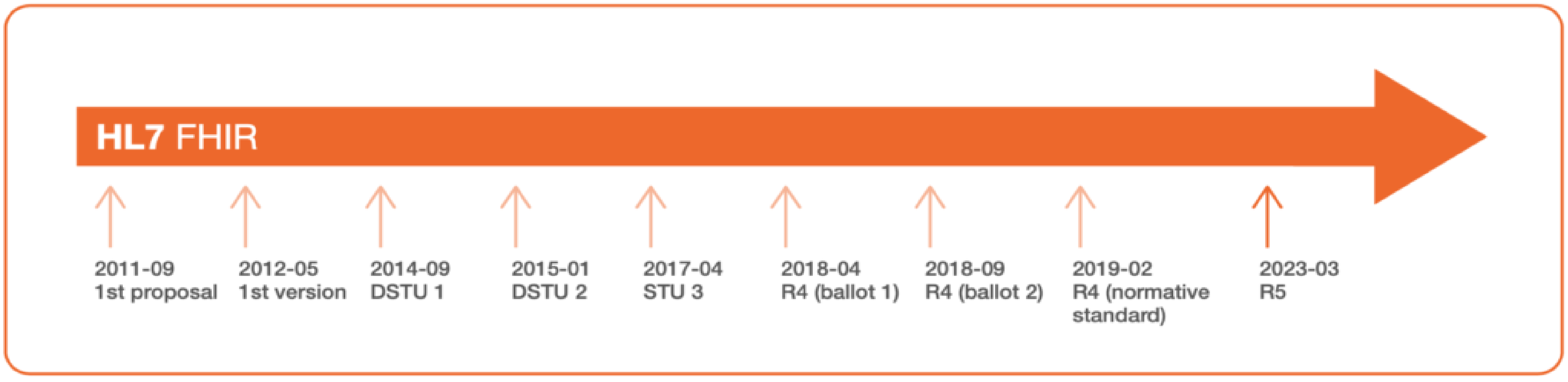
- F - FAST
- H - Healthcare
- I – Interoperability
- R - Resource

https://www.hl7.org/index.cfm

http://hl7.org/fhir/R4/index.html

Pre FHIR

Standard	Type	Used For	Status Today
HL7 v1	Messaging	Legacy	Not used
HL7 v2(1980)	Messaging	Lab, ADT, Billing	Very widely used
HL7 v3	Messaging/XML	Theoretical	Largely failed
CDA	Document (XML)	Discharge Summary, CCD	Still used
FHIR R4(2019)	API + JSON	Most modern systems	Most common
FHIR R4B	API	Patch update	Transitional
FHIR R5(2023)	API	Latest advanced version	Growing adoption



HL7 FHIR

2011-09 1st proposal

2012-05 1st version

2014-09 DSTU 1

2015-01 DSTU 2

2017-04 STU 3

2018-04 R4 (ballot 1)

2018-09 R4 (ballot 2)

2019-02 R4 (normative standard)

2023-03 R5

Where FHIR fits in

FHIR = **HL7’s modern solution**

Designed to replace:

•HL7 v2 (messaging)

•HL7 v3 (XML complexity)

•CDA (documents)

FHIR uses:

•**API-first approach**

•**JSON, REST**

•**Resources**, not documents

FHIR- topics to cover

- Intro to FHIR - Fast Healthcare Interoperability
  - Resources how to read the resource
  - bundles, key elements, cardinality, Data Types terminology, valueset & binding
  - Manual conversion as per defined structure(json)
  - Publicly available FHIR server (Hapi & Fir.ly)
  - CRUD operation, Rest Paradigm - Postman
  - Search Querying basic to advance include, revinclude
  - Validator –manual, Inferno and through java zar file one
  - Conversion data pipeline – PyTHon way, ETL way
- Mini Project:
  - convert different files into FHIR format for different resources, validate and ingest them to FHIR server
  - Search query practise

What is FHIR

FHIR – Fast Healthcare Interoperability Resource

is a **standard for exchanging healthcare data electronically**. Created by **HL7 International**, it aims to make healthcare data exchange **simple, fast, modern, and developer-friendly**.

Key goals of FHIR

- Easy to implement (JSON, REST APIs, OAuth — like modern web apps)
- Flexible for different healthcare ecosystems
- Supports global interoperability
- Reduces cost of integrations
- Standardizes healthcare data formats

Why FHIR ??

Before FHIR:

- HL7 v2 → Very widely used, but inconsistent (pipe-delimited messages, custom formats).
- HL7 v3 → Too complex to adopt widely.
- CDA → Good for documents, not API-style exchanges.

FHIR solves these by:

- Being **modular** → small building blocks called **resources**
- Being **web-native** → REST, JSON, XML, OAuth2
- Being **extensible** but still standardized
- Being **profiled** to meet real-world requirements (e.g., US Core)

Discussion: why FHIR- current problem

Before FHIR (Legacy Challenges)

- **Proprietary Data Formats** – Every hospital, payer, and EHR vendor(cerner & epic) had its own data format (HL7 v2, CDA, custom CSV/XML).
- **Lack of Real-Time Data Sharing** – Claims data and clinical data were exchanged in **batch files (X12, flat files)** taking days or weeks.
- **Difficult Integration** – Point-to-point integrations between providers, payers, and third parties were expensive & time-consuming.
- **Limited Patient Access** – Patients couldn’t easily access their own medical data across different healthcare systems.
- **Regulatory Push** – The US government (via **CMS & ONC** rules, like CMS-9115-F and CMS-0057) mandated open APIs for interoperability.

Discussion: why FHIR-solution

With FHIR

- **Standardized Format** – JSON, XML, and REST-based APIs (like modern web applications).
- **Real-Time Interoperability** – Data exchange in seconds, not days.
- **Plug-and-Play APIs** – Any app can connect to any FHIR-compliant server.
- **Patient Empowerment** – Patients can get their clinical & claims data via apps (SMART on FHIR, Patient Access APIs).
- **Regulatory Compliance** – Meets US CMS Interoperability & Patient Access Final Rule.

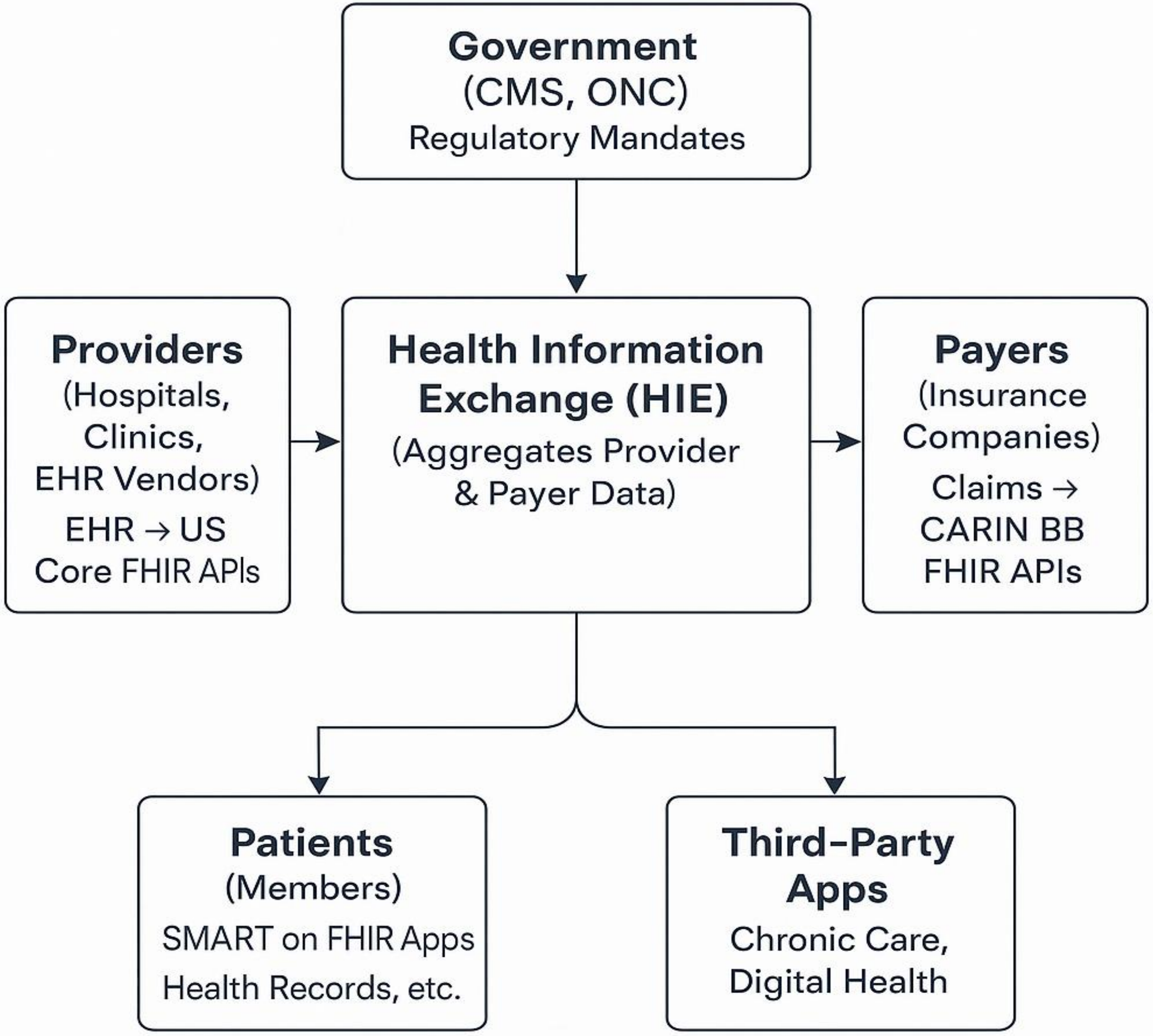
Discussion: why FHIR-solution

Think of FHIR as the **“universal language”** allowing different healthcare entities to communicate seamlessly.

Main Player & Their Role

Entity	Data Type Shared	FHIR Use Case
Providers (Hospitals, Clinics, Physicians)	Clinical data (EHR: encounters, observations, medications)	Share patient records via FHIR APIs (e.g., US Core Implementation Guide)
Payers (Insurance Companies like BCBS, Aetna, Humana)	Claims, Coverage, Prior Authorizations	Provide Patient Access & Provider Access APIs using CARIN-BB & Da Vinci guides
Patients (Members)	Access to their own data	Mobile apps fetch FHIR data from Payers & Providers
Health Information Exchanges (HIEs)	Aggregated regional data	Use FHIR to share with providers & payers
Third-Party Apps / Digital Health Vendors	Wellness apps, chronic care mgmt	Consume FHIR APIs via SMART on FHIR
Government Agencies (CMS, ONC)	Regulatory enforcement	Require FHIR-based API implementations

Discussion: High Level Diagram



Government
(CMS, ONC)
Regulatory Mandates

Providers
(Hospitals, Clinics, EHR Vendors)
EHR → US Core FHIR APIs

Health Information Exchange (HIE)
(Aggregates Provider & Payer Data)

Payers
(Insurance Companies)
Claims → CARIN BB FHIR APIs

Patients
(Members)
SMART on FHIR Apps
Health Records, etc.

Third-Party Apps
Chronic Care, Digital Health

How it is structured

How FHIR is structured

- **Resources** → smallest unit (ex: Patient, Observation)
- **Profiles** → constraints on resources
- **ValueSets/CodeSystems** → controlled coding
- **Interactions** → REST operations
- **Implementation Guides (IGs)** → packaged rules for a project

FHIR version and ecosystem

FHIR Versions

Most common:

- **R4** → Stable, widely used
- **R4B** → Minor patch on R4
- **R5** → Newer, not fully mandated yet

Production systems mostly use **R4/R4B**.

The FHIR Ecosystem

Tools & servers:

- **Servers** → HAPI FHIR, Firely Server, Microsoft FHIR Server, HealthLake
- **Validators** → HL7 Validator, Inferno
- **Profiling tools** → Forge, SUSHI
- **Apps** → SMART-on-FHIR apps

Resources

What is a FHIR Resource?

A **resource** is a small, structured unit of healthcare data.

Examples:

- **Patient** → demographics
- **Encounter** → hospital visits
- **Observation** → vitals, labs
- **MedicationRequest** → prescriptions
- **Condition** → diagnoses
- **AllergyIntolerance** → allergies

FHIR has **150+ resources**, grouped by domain (clinical, financial, workflow).

Resources

Resource Structure

Each resource contains(Inheritence)

Resource --- DomainResource --- Resource

- **id**
- **meta**
- **text (narrative)**
- **data elements (attributes)**
- **references** (links to other resources)

FHIR version and ecosystem

Resource Guide

This page describes the resources and their functional intent in more detail to assist implementers to understand their purpose and scope, and their supporting classifications.

http://hl7.org/fhir/resourceguide.html

Data type

FHIR Data Types — Complete Explanation

- FHIR defines **data types** to represent the “shape” and “format” of data elements inside resources.
- Every attribute in a resource (like Patient.name, Observation.value, Encounter.period) uses one of these data types.

FHIR defines **3 levels** of data types:

- **Primitive Types**
- **Complex Types**
- **Specialized Data Types** (a subset of complex types)

Data type- primitive

Primitive Data Types (simple, text-based)

- These are the smallest units of data.They always start with a **lowercase** letter.

Type	Meaning
boolean	true / false
integer	whole numbers
decimal	decimal numbers
string	text
code	coded string restricted by a ValueSet
id	internal FHIR identifiers
uri	Uniform Resource Identifier
url	internet URL
canonical	canonical URL for FHIR artifacts
base64Binary	Base64 encoded data
instant	exact timestamp (precision to milliseconds)
date	only the date (YYYY-MM-DD)
dateTime	date + time
time	time only (HH:MM:SS)
oid	object identifier
markdown	text with markdown formatting
unsignedInt	integer ≥0
positiveInt	integer >0

Data type

Complex Data Types

- These are **structured objects** that contain multiple fields. They start with an **uppercase** letter.

Example:

1. HumanName has family, given, prefix, suffix.

```
{
               "use": "official",
               "family": "doe",
               "given": [
                    "john"
               ]
          }
```

2. Address

```
{
  "line": ["A-222 MG Road"],
  "city": "Pune",
  "postalCode": "411001"
}
```

Terminology binding

Terminology binding means **linking a FHIR element** (like Observation.code, Condition.code, Patient.gender, etc.) **to a set of allowed coded values**.

In simple words:

- **It tells what codes are allowed for a specific field in a FHIR resource.**
- FHIR uses **ValueSets** to define the allowed codes, and terminology binding tells *which* ValueSet applies *where*.

Terminology binding

Healthcare uses *coded data* for consistency and interoperability:

- SNOMED CT → clinical concepts
- LOINC → labs
- ICD-10 → diagnoses
- RxNorm → medications
- HL7 internal codes
- Custom/local codes (only if needed)

Terminology binding ensures:

- Data is **standardized**
- Systems **understand each other**
- API consumers consistently **interpret values**
- Validation becomes possible

Terminology binding

Examples:

1. ***Observation.code***

must come from a LOINC ValueSet

2. ***Condition.code***

must come from SNOMED CT or ICD-10 ValueSet

3. ***Patient.gender***

must come from the AdministrativeGender ValueSet

4. ***AllergyIntolerance.code***

must come from a SNOMED allergy ValueSet

Terminology binding

Terminology binding involves:

- **Element** (FHIR data field)
  - Example: Observation.status
- **ValueSet** (collection of allowed codes)
  - Example: http://hl7.org/fhir/ValueSet/observation-status
- **Binding Strength** (very important!)
  - Required – must support --- (validation error)
  - Extensible
  - Preferred
  - Example

These 3 things form a **complete binding definition**.

Terminology binding

Binding Strengths (VERY IMPORTANT)

This defines *how strictly* you must follow the ValueSet.

1. REQUIRED

You **must** use one of the codes from the ValueSet.

Example:

Observation.status
Allowed: registered | preliminary | final | amended
Cannot send any code outside this set.

2. EXTENSIBLE

Use codes from the ValueSet **unless** your concept cannot be represented, then you can extend with your own code.

Used when:
standard codes exist
but not enough for all use cases

Example:
Procedure.code → SNOMED CT extensible

3. PREFERRED

You **should** use codes from the ValueSet, **but not mandatory**.

Used when:
good standard exists
but local variation is common

4. EXAMPLE

ValueSet is just a **guideline**.
You can use any code system.

Used for educational/example purposes.

LEARNERS Digital

Patient conversion to FHIR JSON

Name : Bruce Wayne

Address : 1007 Mountain Drive, Gotham, NJ, 07001, USA

Phone number : +1-999-000-0001

Gender : male

Dob : 19th Feb 1972

What is a Canonical in FHIR?

- A **canonical** is a special FHIR **data type** that represents a **canonical URL** pointing to a **FHIR definition**.
- **Canonical = reference to a FHIR *definition***
- **NOT a reference to a resource instance**
- **NOT used for clinical data links**
- Canonical is used only for **metadata**, **profiles**, **terminology**, and **IGs**.
- **Canonical means: A globally unique URL that identifies a FHIR definition.**
- This URL **does not need to be a real web page** —it is just a globally unique identifier.

"Canonical is a unique URL that identifies a FHIR definition (profile, ValueSet, CodeSystem, etc.). It's used for linking definitions, not data"

What Can a Canonical Point To?

Canonical URLs identify **FHIR artifacts**, such as:

- **StructureDefinition (profiles)**

http://example.org/fhir/StructureDefinition/MyPatientProfile

- **ValueSet**

http://hl7.org/fhir/ValueSet/observation-status

- **CodeSystem**

http://loinc.org

http://snomed.info/sct

- **OperationDefinition**
- **SearchParameter**
- **CapabilityStatement**
- **ImplementationGuide**

Reference and canonical

What is a Reference in FHIR?

- A **Reference** is a data type in FHIR used to **link one resource to another**.
- It creates **relationships** between resources.

Think of it like a **foreign key** in a database.

- **Reference = link to another real-world resource instance**
- **Points to actual FHIR resources stored on a server**
- Not used for definitions (that’s what canonical is for)

```
Syntax {
  "reference": "Patient/123"
}
```

Example Observation referring Patient

```
"subject": { "reference": "Patient/123" }
```

Encounter

```
"subject": { "reference": "Patient/123" }
```

MedicationRequest

```
"performer": { "reference": "Practitioner/45" }
```

Condition - encounter

```
"encounter": { "reference": "Encounter/789" }
```

Type of Reference

FHIR allows different ways to reference resources:

1. Simple (relative) reference

"reference": "Patient/123"

Most common.
Server interprets relative to its base URL.

2. Absolute URL reference

"reference": "https://api.hospital.com/fhir/Patient/123"

Used when referencing across systems.

3. Logical reference (identifier-based)

No actual FHIR URL → only identifier known.

```
{
 "identifier": {
  "system": "http://hospital.com/mrn",
  "value": "998877"
 }
}
```

Used when resource doesn't exist yet or is not accessible.

4. Contained resource reference

"reference": "#med1"

When resource is embedded inside another resource.

5. Version-specific reference

"reference": "Observation/123/_history/2"

Points to a specific version of the resource.

Reference-Structure

type

Explicitly states what type of resource is expected.

```
{
  "reference": "Practitioner/45",
  "type": "Practitioner"
}
```

identifier

Used for logical references.

display

Human-friendly description.

```
{
  "reference": "Patient/123",
  "display": "John Doe"
}
```

Reference-why

Why Reference is critical in FHIR

References create a **graph of interconnected clinical data**.

For example:

```
Patient <-----Encounter <--Condition ←-
  Observation ←——— DiagnosticReport
```

Without references, resources would be isolated and meaningless.

Reference-Rules

Important Rules of Reference

References must be resolvable

Server should locate the resource.

Must follow resource type allowed in the logical model

Example:

`Observation.subject` → can reference

- Patient
- Group
- Location
- Device

Use relative references(Litteral Reference) when resources are on the same server

Better portability.

“**Reference** in FHIR is a data type used to link one real, instance-level FHIR resource to another FHIR resource”

Get all Observations for a Patinet

FHIR REST API: Get All Observations for a Patient

GET [baseURL]/Observation?subject=Patient/{patientId}

System ??

WHAT IS "system" IN FHIR?

In FHIR, **system** is a URI that tells **which namespace the identifier or code belongs to**.

It defines the *authority* that issued the identifier

It ensures the identifier is globally unique

It prevents collisions (because many hospitals may use similar numbers)

Where do you see "system"?

Identifier.system

Coding.system

Quantity.system (units)

NamingSystem

Terminology systems (LOINC, SNOMED, RxNorm)

1) system in Identifier

Example

```
"identifier": [
  {
    "system": "http://hospitalA.com/mrn",
    "value": "12345"
  }
]
```

Meaning

`system` = who issued this MRN

`value` = actual ID number

The same value from another system is **NOT the same patient**:

```
System A → value 12345
System B → value 12345
```

System tells you which MRN namespace this belongs to.

2) system in Coding

Every coded value must come from a known **terminology system**.

Example (LOINC)

```
"coding": [
  {
    "system": "http://loinc.org",
    "code": "8310-5",
    "display": "Body temperature"
  }
]
```

Here:

`system` = "This code is from LOINC terminology"

`code` = "8310-5"

Without system, a code is meaningless.

Bundle

What is a Bundle in FHIR?

A **Bundle** is a FHIR resource that represents a **collection of resources packaged together**.

Bundles are used when **multiple resources need to travel together**.

https://hl7.org/fhir/R4/bundle.html

See the bundle structure

Bundle-Types

1. Collection

A simple group of resources. Used when you just want to store multiple related resources together. It has **no server actions**, no create/update/delete, no GETs.
It's **just a package of data** — like a **folder** containing multiple files.

2. Searchset (most common in API responses)

Returned when you perform a FHIR search.

```
GET /Observation?patient=123
```

Server response:

```
"type": "searchset"
```

Contains:

matching resources

pagination info (next link, previous link)

3. Batch

Multiple independent requests in one Bundle.

Execute all requests
Order does NOT matter
If one fails → others continue

Example:

GET Patient/123

GET Observation/456

POST Condition

Bundle-Types

4. Transaction

Multiple operations that must succeed **atomically**.

```
"type": "transaction"
```

Either **all succeed** or **none succeed**

Used for synchronized database updates

Example:

Create Patient

Create Encounter referencing Patient

Create Observations referencing Encounter

All must happen together.

5. Message

FHIR messaging like HL7v2 replacement.

Used in real-time event systems.

6. Document

Represents a **clinical document** like a CCD, discharge summary, etc.

Must have a **Composition** resource as the first entry.

Bundle

```json
{
  "resourceType": "Bundle",
  "type": "transaction",
  "entry": [
    {
      "fullUrl": "urn:uuid:p1",
      "resource": { "resourceType": "Patient" },
      "request": { "method": "POST", "url": "Patient" }
    },
    {
      "fullUrl": "urn:uuid:e1",
      "resource": {
        "resourceType": "Encounter",
        "subject": { "reference": "urn:uuid:p1" }
      },
      "request": { "method": "POST", "url": "Encounter" }
    }
  ]
}
```

CapabilityStatement

A **CapabilityStatement** is a document published by a FHIR server that describes:

“What this server can do.”

It tells clients (apps, other systems) exactly which:

- resources the server supports (Patient, Observation, Encounter…)
- operations it supports (search, read, write, transaction…)
- search parameters you can use
- security / authentication methods
- versions of FHIR it understands

Think of it like a **contract** or **API documentation** for a FHIR server.

Because every FHIR server is different. Clients must know the server’s capabilities **before** interacting with it.

CapabilityResource

Key sections:

1. Software Info

Name, version, publisher.

2. FHIR Version

R4, R4B, R5, etc.

3. REST Endpoints

For each endpoint:

- mode: server / client
- interactions supported (read/update/delete/search)
- supported resources

4. Supported Resources

CapabilityResource: Key sections

1. Software Info

Name, version, publisher.

2. FHIR Version

R4, R4B, R5, etc.

3. REST Endpoints

For each endpoint:

mode: server / client

interactions supported (read/update/delete/search)

supported resources

4. Supported Resources

```
{
 "type": "Patient",
 "interaction": [
  { "code": "read" },
  { "code": "search-type" },
  { "code": "create" }
 ]
}
```

CapabilityResource: Key sections

5. Search Parameters

Which queries are allowed:

identifier

family

birthdate

gender

6. Security

How to authenticate:

OAuth2

SMART on FHIR

API keys

7. Operations

Special FHIR operations the server supports:

`$validate`

`$everything`

`$expand`

`$lookup`

Search Query in FHIR

FHIR Search is a RESTful way to retrieve resources from a FHIR server using **URL query parameters**.

Example

GET {BASE_URL}/Patient?name=somename

Returns all patiens where the name contains “somename”

Reference

https://hl7.org/fhir/R4/search.html

Search Query :structure

Syntax

GET [baseURL]/[ResourceType]?[SearchParameter]=[value]

Example

GET https://fhirserver.com/Patient?identifier=12345

Search Query :different elements

Element	Meaning	Example
Base URL	FHIR server root	`https://hapi.fhir.org/baseR4`
Resource Type	FHIR resource collection to search	`Patient`
Search Parameter	The field you want to filter by	`name`, `identifier`, `birthdate`
Modifier (optional)	Controls how search matches	`name:contains=tony`
Prefix (optional)	Used for numbers, dates	`birthdate=gt2010-01-01`
Chaining (optional)	Search linked resources	`Encounter?patient.name=stark`
RevInclude/Include (optional)	Load referenced resources	`_include=Patient:organization`
Sort, Pagination, Count	Control response	`_sort=-birthdate&_count=20`

Search Query :types of search param

Type	Meaning	Example
string	Text match	name=stark
token	Exact code match	`identifier=mrn
reference	Resource reference	general-practitioner=Practitioner/12
number	Numeric values	value-quantity=gt5
date	Dates/timestamps	birthdate=ge1990-01-01
uri	Matching URIs	url=http://loinc.org

Search Query :modifiers

Modifiers change how the matching works

String Modifiers
name:contains=tony
name:exact=Tony Stark

Token Modifiers
identifier:of-type=mrn|12345
code:text=aspirin

Reference Modifiers
patient:identifier=mrn|12345

Search Query :Prefixes (for date & number searches: used for date, datetime,number,quantity)

Prefix	Meaning
eq	equal
ne	not equal
gt	greater than
lt	less than
ge	greater than or equal
le	less than or equal
sa	starts after
eb	ends before

birthdate=ge1990-01-01
value-quantity=gt5

Search Query :Multiple Search Conditions

AND (all conditions must match)

Patient?gender=male&birthdate=ge1990-01-01

OR (any value matches)

Patient?name=tony,bruce

Search Query :Chained Parameters

Used when searching *via* a reference.

Example: Find encounters where patient name contains "stark":

Encounter?patient.name=stark

Patient?general-practitioner.name=Strange

Search Query :_include and _revinclude

Include referenced resources

Patient?_include=Patient:organization
→ Also returns the organization resources.

Reverse include (who references this)

Patient?_revinclude=Encounter:patient
→ Returns encounters linked to that patient.

Search Query :Sorting, Pagination & Count

Sorting

Patient?_sort=birthdate #ascending

Patient?_sort=-birthdate # descending

Pagination

Patient?_count=50

Continue page(server returns next link)

GET [nextPageUrl]

Search Query :Examples

1. Patient by MRN (token search)
Patient?identifier=system|12345

2. Patient with partial name match

Patient?name:contains=ton

3. Encounter for patient MRN 12345

Encounter?patient.identifier=mrn|12345

4. Observations for patient 123 after 2024

Observation?patient=123&date=gt2024-01-01

5. All patients with both name and DOB

Patient?name=stark&birthdate=1970-05-29

Search Query :Rules

Official FHIR Rules Summary
BaseURL + ResourceType → must always come first

Search parameters → must be supported by server (CapabilityStatement tells you)

Strings → partial match by default

Tokens → system + value format: system|value

Dates → must use prefixes (ge, gt, le, etc.)

Modifiers → change meaning of search

Chaining → use when referencing another resource

Include/RevInclude → bring referenced data

Multiple parameters → & = AND, , = OR

Search Query :Business query

get all the heart rate data where heart rate value is less than equal to 44, search this data for Patient whose resource Id is 17579

/Observation?code=value&value-quantity=value&subject=value

Code=http:/lonic.org|8867-4&value-quantity=le44&subject=Patient/17579

Profile & IG

What is FHIR Profiling?

Think of **FHIR** as a *flexible universal standard*.

But every country, hospital, payer, lab, app, or government has:

- different business rules
- different mandatory fields
- different code systems
- different workflows

So **one single FHIR structure cannot fit all real-world use cases.**

Profiling is the process of customizing FHIR to suit your specific use case

Profiling = adding rules on top of FHIR resources to fit your project.

Profile :what customization

we create a **Profile** that modifies/extends a FHIR Resource:

What we can:

- **Make optional fields mandatory**
  Patient.name must be required
- **Restrict data types**
  birthDate must be YYYY-MM-DD only
- **Limit code systems**
  Gender must be from HL7 AdminGender
- **Add extensions**
  Add “Patient mother’s maiden name”
- **Restrict cardinality**
  address can appear only once
- **Control value patterns**
  MRN must match regex
- **Add business rules**
  Observation must reference a Patient

Profile :summary

Summary

- **we can only make rules tighter, not looser.**
- **we can add new things but cannot remove required things.**
- **we cannot weaken the base FHIR structure.**

Implementation Guide

What is an Implementation Guide (IG)?

A FHIR **Implementation Guide (IG)** is a **published website + package** that contains:

- Your Profiles
- Extensions
- CodeSystems
- ValueSets
- Examples
- Rules
- Workflows
- Search parameters
- Documentation
- Testing scripts
- Capability Statements

IG = Complete instructions for implementing your API
IG is what others use to build and validate against your FHIR API

Profile = the Rule
Implementation Guide = Book that contains all your rules

Profile :IG

Why are Profiles & IGs important?

- **Without profiles, every FHIR server would behave differently → chaos** Profiles ensure **interoperability**.
- **IGs provide a single source of truth**

Others know:

- what fields are mandatory
- what codes to use
- what API endpoints exist
- real examples

Profiles allow validation

A FHIR server can validate incoming/outgoing resources against your profiles.

Required for national programs

- e.g. US Core, Da Vinci, CARIN, IHE, NHS

Required for app certification

- e.g., SMART on FHIR apps need to follow IG requirements.

Profile &IG : VALIDATION

VALIDATION –

INFERNO

$VALIDATE

HL7 FHIR VALIDATOR JAVA ZAR FILE

Profile &IG : 80/20 rule

What is the FHIR 80/20 Rule?

FHIR follows the principle:

80% of real-world healthcare data needs should be supported using base FHIR resources.
Only 20% should require extensions.

Why does FHIR use the 80/20 Rule?

- Because healthcare is HUGE and complex.
- Every country
- every EHR vendor
- every hospital
- every lab
- every payer
- All capture **different** data.
- If FHIR tried to include *everything*, the core standard would become **bloated, unstable, and impossible to implement**.

Profile &IG : 80/20 rule

So the FHIR designers decided:

- **Include only the common 80%**
- **Let the community handle the specialized 20% (via extensions & profiles)**

This keeps FHIR:

- scalable
- simple
- globally usable
- Flexible

Intropable

Profile &IG : 80/20 rule

How is the 80/20 Rule relevant to Profiling?

- **Profiling = customizing FHIR**
- **Extensions = adding the special 20%**

When creating profiles:

- **we should use standard FHIR elements first (the 80%)**
- **Only create custom extensions when absolutely necessary (the 20%)**

Bad Practice:

Adding custom fields even though standard fields already exist.

Good Practice:

Use FHIR core fields as much as possible.

Week 3: Working with FHIR Resources in Python

- How FHIR fits in healthcare data exchange
- Dict and Json –
  - Serialization and deserilization
  - load, dump, loads, dumps
- File Handling – csv, json, ndjson, array of json
- Conversion to FHIR JSON and Bundle format
  - Python iteration and mapping through code
  - Using fhi.resources library
  - Using Jinja template
- Request Module
  - Basic intro to API CRUD Operation Against Publicly available server
- Validation of Resources
  - Using `fhir.resources` library
  - Cmd using validator jar file
  - $validate operation
  - Using GUI inferno
- Implementation Guide- Profiling
  - Extension and Constraining the Resource
- Basics of git – clone, status, branch, pull, push
- Hands-on:
  - Extract Patient info from FHIR Bundle - read
  - Create and post Bundle to FHIR server – write
  - Patient, Provider, Coverage as Classes
  - Model Patient-Coverage relationships Reference
  - Assessment through github class room 4 descriptive problem for the concepts and 3 coding problem to see the ability of code writing for read and write op to FHIR server.

LEARNERS Digital