NurseView

Medical Intelligent Document Processing (IDP) Pipeline

A Python-based middleware solution designed to ingest unstructured clinical data sources (scans, handwritten notes) and transform them into interoperable FHIR R4 resources and HL7 v2.x messages for seamless EHR integration.

Author: Soham S. Deshmukh

Version: 1.0.0

Date: October 28, 2025

Contents

1 Objective and Scope

1.1 Objective

The primary objective of the NurseView pipeline is to automate the extraction of clinical data from unstructured sources. By leveraging Google Gemini Vision, the system aims to eliminate manual data entry errors and accelerate the availability of patient information in Electronic Health Records (EHR).

1.2 Scope

The pipeline handles the end-to-end processing flow:
- **Ingestion:** Accepting PDFs, JPEGs, and PNGs (Clinical Notes, Labs, Prescriptions).
- **Transformation:** Converting raw pixels to structured JSON entities.
- **Standardization:** Mapping entities to FHIR R4 and HL7 v2.x.
- **Deployment:** Cloud-native execution on Google Cloud Run.

2 High-Level Architecture

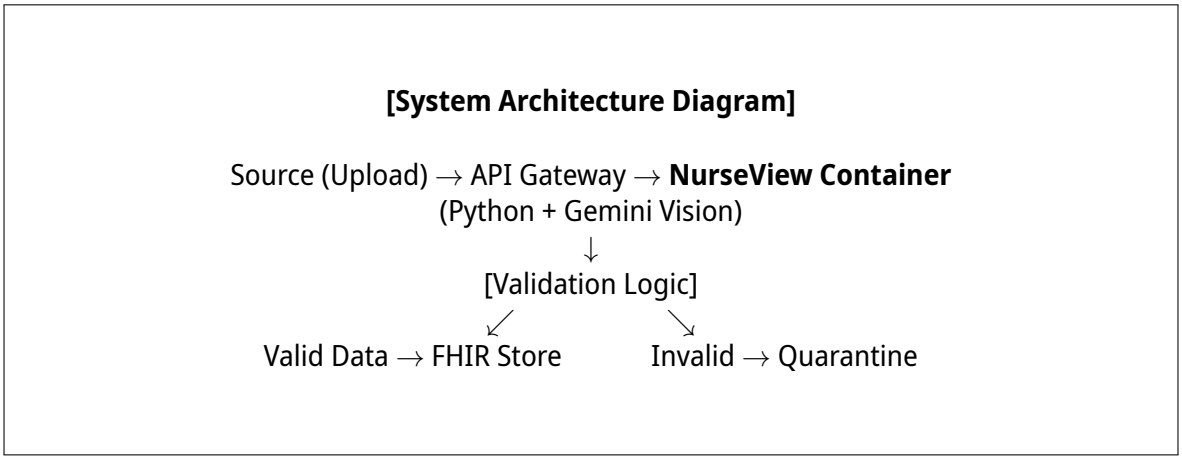

Figure 1: NurseView Component Architecture

2.1 Main Components

- **Orchestrator:** Python application serving as the central controller.
- **AI Service:** Google Gemini Vision API for OCR and Named Entity Recognition.
- **Container Engine:** Docker for packaging dependencies.
- **Infrastructure:** Google Cloud Run (Compute) and Cloud Storage (Quarantine).

3 Pipeline Stage Flow

3.1 Data Pipeline Stages

1. **Ingest:** File arrives via REST API `POST /process`. System performs mime-type validation.
2. **Transform:** Payload is encoded and sent to Gemini Vision. Response is parsed into a standardized internal dictionary.

3. **Load/Serve:** Validated data is converted to FHIR resources and returned as JSON response or pushed to a webhook.

3.2 CI/CD Pipeline Stages

1. **Source:** Code pushed to Git repository.
2. **Build:** Docker image built from `Dockerfile`.
3. **Test:** Unit tests run via `pytest` to verify FHIR mapping logic.
4. **Deploy:** Image pushed to Artifact Registry; Cloud Run service updated.

3.3 Triggers and Sequencing

- **Start:** HTTP Request or Pub/Sub message.
- **Stop:** Completion of response or max retry exhaustion.
- **Manual Approvals:** Required for deployments to the `production` environment.

4 Environments and Configuration

4.1 Environments

- **Dev:** Localhost or Development Project. Debug mode enabled.
- **Staging:** Mirror of production with obfuscated data.
- **Production:** Strict IAM access, optimized concurrency, live EHR connection.

4.2 Configuration & Secrets

- **Config:** Managed via `.env` files and `constants.py`.
- **Secrets:** Stored in Google Secret Manager; injected as environment variables at runtime.

5 Tools, Dependencies, and Integrations

- **Language:** Python 3.9+
- **Libraries:** `fhir.resources`, `hl7apy`, `google-generativeai`.
- **Integrations:** Google Vertex AI, FHIR Store API.

6 Governance and Change Management

6.1 RACI Matrix

- **Responsible:** Lead Developer (Implementation).
- **Accountable:** Product Owner (Roadmap).
- **Consulted:** Medical Informatics Specialist (Standards).
- **Informed:** Users/Clinicians.

6.2 Change Protocol

Modifications follow the "Git Flow" strategy. Feature branches merge to `develop`; `main` is protected. Pull Requests require 1 peer review and passing CI tests.

7 Developer Quick-Start

```
# Clone and setup
git clone https://github.com/opensource/nurseview.git
cd nurseview
pip install -r requirements.txt

# Set API Key (Compact style)
export GEMINI_API_KEY="your_key_here"

# Run local
python app.py
```

Listing 1: Setup

```
# Compact operator style as requested
def calculate_retry(attempt):
    base_delay= 2
    timeout= base_delay+ (attempt* 2)
    return timeout
```

Listing 2: Logic Example

8 Glossary

Quarantine Bucket A storage location for files that fail validation, isolating them for manual human review.

FHIR Fast Healthcare Interoperability Resources.

IDP Intelligent Document Processing.