IRIS EHR Agent

## Abstract

Hospitals need to convert patient notes into standard billing codes (ICD-9 for diagnoses, CPT for procedures). Currently, human coders read medical notes and manually find matching codes - this is slow, expensive, and blocks hospital revenue.

The IRIS Medical Coding Assistant will use Generative AI to automate this task: it should search through previously-coded notes to find similar cases and suggest medical codes instantly. When it's not confident, it should use a smart AI model to reason through the problem. This creates fast, reliable code assignment.

# Background -

TechCare Solutions is a healthcare technology provider working with multiple hospital clients. Currently, these hospitals:

* **Manually map** patient notes and look up medical codes in books/databases
* **Spend hours per record**, creating huge delays
* **Experience inconsistent coding**, that lead to billing problems and compliance issues
* **Can't find enough trained medical coders,** making operations expensive

To stay competitive, TechCare's clients need an automated system that can quickly read patient notes, find similar past examples, and suggest accurate codes - freeing their staff to focus on patient care.

## Objective

Build a simple AI coding assistant that:

* Takes a patient note (like "Patient has chest pain, did X-ray")
* Finds similar past cases automatically
* Suggests the most likely billing codes
* Shows confidence level and reasoning

# Problem Statement

MedClinic needs a system that:

1. **Reduces coding time** from hours to minutes
2. **Improves accuracy** by learning from past cases
3. **Handles busy periods** without hiring more staff
4. **Provides consistent results** across all staff

**Your task:** Design and implement, on IRIS, a LangGraph-based EHR RAG agent that:

* Ingests and indexes clinical note datasets
* Uses Vertex AI Embeddings (or Weaviate DB) + Matching Engine for vector retrieval of similar past records
* **Indexes** historical clinical notes and their ICD-9/CPT codes in Vertex AI Matching Engine
* **Retrieves** the top-K most similar records for any new note
* **Aggregates** and assigns codes based on retrieval confidence
* **Falls back** to Gemini 2.5 Flash reasoning when similarity is below threshold
* **Deploys** to Vertex AI Endpoints or Cloud Run behind a secure API
* **Monitors** key metrics (accuracy, latency, fallback rate, token usage) in real time and accuracy via LangFuse/mlflow

# Data

|  |  |  |
| --- | --- | --- |
| Dataset | Records | Purpose |
| source\_records.csv | 1 ,976 notes with JSON-encoded ICD-9 & CPT mappings | Ground-truth corpus for retrieval and evaluation |
| Vector index (pre-loaded) | 15 K note embeddings | Enables instantaneous top-K similarity search |
| Loader scripts | Python notebooks | Push vectors and metadata to Vertex AI Matching Engine |

**1. Source Records Dataset (source\_records.csv)**

This dataset contains 1,976 clinical records with 15 fields, including clinical documentation (radiology reports and medical examination notes) with comprehensive medical code mappings. The dataset includes patient identifiers, timestamps, clinical categories, detailed clinical text, and JSON-formatted arrays of ICD-9 diagnoses, procedures, and CPT codes with complete descriptions.

**Data Dictionary - Source Records:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Sample Value** |
| row\_id | Integer | Unique identifier for each clinical record | 1, 2, 3... |
| SUBJECT\_ID | Integer | Anonymous patient identifier | 3, 5, 7... |
| HADM\_ID | Integer | Hospital admission identifier | 145834, 142345... |
| CHARTDATE | String | Date when clinical documentation was created | "2101-10-26" |
| CHARTTIME | String | Date and time of documentation | "2101-10-26 6:01" |
| STORETIME | String/Null | Storage timestamp (mostly null) | null |
| CATEGORY | String | Clinical department or service category | "Radiology", "Nursing", "Physician" |
| DESCRIPTION | String | Brief description of the clinical procedure/exam | "CHEST (PORTABLE AP)", "CARDIAC CATH" |
| CGID | Float/Null | Caregiver identifier (mostly null) | null, 1234.0 |
| ISERROR | String/Null | Error flag for documentation (mostly null) | null, "Y" |
| TEXT | String | Complete clinical documentation including radiology reports, examination notes, medical findings, and impressions | "[**2101-10-26**] 6:01 AM CHEST (PORTABLE AP)... IMPRESSION: These findings are consistent with moderate left heart failure..." |
| diagnoses | String (JSON) | Array of ICD-9 diagnosis codes with descriptions | "[{'ICD9\_CODE': '2639', 'SHORT\_TITLE': 'Protein-cal malnutr NOS', 'LONG\_TITLE': 'Unspecified protein-calorie malnutrition'}...]" |
| procedures | String (JSON) | Array of ICD-9 procedure codes with descriptions | "[{'ICD9\_CODE': 9604, 'SHORT\_TITLE': 'Insert endotracheal tube', 'LONG\_TITLE': 'Insertion of endotracheal tube'}...]" |
| cpt\_codes | String (JSON) | Array of CPT codes with section headers and descriptions | "[{'CPT\_CD': '94003', 'SECTIONHEADER': 'Medicine', 'SUBSECTIONHEADER': 'Pulmonary', 'DESCRIPTION': 'VENT MGMT;SUBSQ DAYS(INVASIVE)'}...]" |

# Solution Design & Detailed Phases

**Phase 1: Setup**

**Data Preparation**

* Clean up 100 sample patient records
  + Inspect source\_records.csv for missing values, malformed JSON fields, duplicate records
  + Standardize date formats (CHARTDATE), normalize whitespace, remove PHI if present
  + Tokenize or segment very long notes into 1–2 KB chunks for efficient embedding
* Upload to Google Cloud Storage

**Tools Used:**

* Google Cloud Storage (file storage)
* Vertex AI Workbench (data cleaning)

**Phase 2: Smart Search**

**Vector Search Setup**

* Convert patient notes to numbers (embeddings) using Vertex AI
  + Call Vertex AI Embeddings API to generate 1,024-dim vectors for each chunk (size is optional)
  + Attach metadata: row\_id, SUBJECT\_ID, HADM\_ID, original text snippet
* Create searchable index of past cases
* Test with sample queries

**Tools Used:**

* Vertex AI Embeddings API (Represent documents as vectors)
* Vertex AI Matching Engine (fast similarity search)

**Example:**

* New note: "Broken arm, needs cast"
* System finds: 3 similar past cases with broken bones
* Suggests codes used in those cases

**Phase 3: Smart Assistant**

**Retrieval Agent**

* Takes new patient note
* Finds top 3 most similar past cases
* Calculates confidence score

**Reasoning Agent**

* If confidence is low (less than 70%), use AI reasoning
* Ask Gemini AI: "Based on these similar cases, what codes should we use?"
* Provide explanation for the suggestion

**Tools Used:**

* LangGraph (workflow management)
* Vertex AI Gemini (AI reasoning)

**Phase 4: Sample Web Interface**

**User Interface**

* Simple form: paste patient note, click "Get Codes"
* Shows: suggested codes, confidence level, similar cases found
* Allow manual corrections and feedback

**Tools Used:**

* App Engine (web hosting)
* Cloud Functions (API backend)

**Phase 5: Monitoring**

**Basic Tracking**

* Count how many codes suggested per day
* Track accuracy (when users make corrections)
* Alert if system is down or slow

**Tools Used:**

* Cloud Monitoring (system metrics)
* LangFuse/mlflow (AI performance tracking)

# **Expected Deliverables**

1. **Code Assets**
   * LangGraph workflow definitions
   * Python scripts: data ingestion, embedding loader, Cloud Functions API
   * Prompt templates for both RetrieverAgent and ReasonerAgent
   * App Engine application code
2. **Google Cloud Configuration**
   * Cloud Deployment Manager templates for all GCP resources
   * Cloud Functions deployment configurations
   * Vertex AI Matching Engine setup scripts
3. **Deployment Artifacts**
   * Cloud Functions URLs
   * App Engine application URLs
   * Cloud IAM policy configurations
4. **Monitoring & Alerts**
   * Cloud Monitoring dashboard configurations
   * Alert policy definitions
5. **Documentation**
   * Architecture diagrams
   * API documentation for the /ehr/encode endpoint
   * Setup and troubleshooting guides