MedScan AI

Group 29

Aditya Ratan Jannali,
Harshitha Chandrashekar,
Saumya Gupta,
Saurabh Suresh Kothari,
Sriharsha Patallapalli,

Table of Contents

[1] Introduction

Healthcare systems worldwide face significant disparities in access to medical services, with diagnostic imaging representing a critical bottleneck in many regions. While healthcare facilities in resource-constrained environments often possess essential imaging equipment such as CT scanners and X-ray machines, they frequently lack the specialist radiological expertise needed to interpret these scans effectively and efficiently. This expertise gap creates a devastating paradox: expensive diagnostic equipment sits underutilized while patients wait days or weeks for accurate diagnoses, often requiring costly referrals to distant urban centers.

The numbers illustrate this crisis starkly. The World Health Organization acknowledges that many low- and lower-middle-income countries cannot afford sufficient imaging equipment and face severe shortages of trained healthcare workers to operate and interpret diagnostic imaging. Sub-Saharan Africa exemplifies this challenge, with countries like Ghana having fewer than four radiologists per million people (50 times below recommended levels).

Within this broader context of global healthcare disparities, our project focuses specifically on developing an AI-assisted radiological system designed for deployment in resource-constrained environments, addressing the critical need for accessible diagnostic support across multiple medical conditions where specialist expertise is limited.

Our solution is a comprehensive patient-radiologist assistant that fast-tracks the process from medical scan analysis to report delivery. The system provides automated preliminary analysis for conditions including lung cancer and tuberculosis, with Grad-Cam explanations for model transparency, a feedback loop that captures radiologist corrections for continuous model retraining, and AI-powered report generation. Additionally, patients can access their scan history and interact with a RAG-based chatbot for diagnosis-related questions. The system maintains human oversight—radiologists review, edit, and approve all AI-generated reports before patient delivery, ensuring clinical accuracy while accelerating workflow efficiency.

[2] Dataset Information

Dataset Introduction

MedScan AI utilizes two comprehensive medical imaging datasets. These datasets cover critical conditions including tuberculosis and lung cancer, diseases that significantly impact populations in underserved regions and require timely radiological diagnosis.

The datasets provide two critical imaging modalities (chest X-rays and CT scans) necessary for training robust multi-disease detection models. Both datasets include expert-validated annotations and have been used in peer-reviewed research, ensuring clinical validity and reliability for our diagnostic support system.

Data Card

Dataset 1: Tuberculosis Chest X-ray Dataset

- **Size**: 4,200 chest X-ray images (700 TB-positive, 3,500 normal)
- **Format**: JPEG images standardized to 512×512 pixels
- **Data Types**:
  - Medical Images: Frontal chest X-ray radiographs (grayscale)
  - Labels: Binary classification (TB positive vs Normal)
  - Metadata: Expert physician grading with three-physician validation
- **Class Distribution**:
  - TB positive: 700 images (16.7%)
  - Normal: 3,500 images (83.3%)
- **Color Space**: Grayscale
- **Validation**: Expert physician screening and peer-reviewed publication validation

Dataset 2: CT Scan Images for Lung Cancer Dataset

- **Size**: 1,000 CT scan images
- **Format**: JPEG/PNG standard image formats
- **Data Types**:
  - Medical Images: Chest CT scans

    - Labels: Multi-class histological classification (4 categories)
    - Metadata: Cancer type classification and clinical characteristics
- **Classes and Distribution**:
    - Normal: 215 images (21.5%) - Healthy lung tissue baseline
    - Adenocarcinoma: 338 images (33.8%) - Most common lung cancer type
    - Large Cell Carcinoma: 187 images (18.7%) - Rapidly spreading undifferentiated cells
    - Squamous Cell Carcinoma: 260 images (26.0%) - Central airway cancer type
- **Medical Context**: Histopathological lung cancer classification
- **Validation**: Widely cited in peer-reviewed research with benchmark accuracy metrics

Combined Dataset Summary:

- **Total Images**: 5,200 medical images across two modalities
- **Disease Coverage**: Tuberculosis (infectious disease), Lung cancer (oncology)
- **Imaging Modalities**: Chest X-ray, CT scans
- **Total Storage**: Approximately 1.5-2 GB
- **Class Balance**: Mixed distribution requiring stratified sampling strategies

Data Sources

Tuberculosis Chest X-ray Dataset:

1. **Source**: Kaggle - https://www.kaggle.com/datasets/tawsifurrahman/tuberculosis-tb-chest-xray-dataset
2. **Creator**: Researchers from Qatar University and University of Dhaka
3. **Publications**: Published in IEEE and multiple peer-reviewed academic papers
4. **Access Method**: Direct download from Kaggle platform

CT Scan Images for Lung Cancer Dataset:

1. **Source**: Kaggle - https://www.kaggle.com/datasets/dishantrathi20/ct-scan-images-for-lung-cancer
2. **Creator**: Dishant Rathi
3. **Publication Year**: 2023
4. **Access Method**: Direct download from Kaggle platform

RAG System Knowledge Base Sources:

1. **Medical Literature**: PubMed Central open-access articles
2. **Patient Education Resources**: MedlinePlus tuberculosis and lung cancer information, CDC patient education materials
3. **Medical Terminology**: RadLex radiology lexicon
4. **Report Templates**: Standard chest X-ray and CT scan reporting formats
5. **Public Resource**: Public APIs for medical data, and documents with open licenses

Data Rights and Privacy

1. All datasets are pre-anonymized by original dataset creators with no personally identifiable information (PII). Images contain only medical scan data without patient-identifying markers. The datasets are HIPPA and GDPR compliant.
2. Datasets are publicly available on Kaggle under open data licensing. Academic and research use is permitted for both datasets.
3. Continuous Learning Data Collection: New patient scans will be collected through the feedback loop for model retraining purposes. All collected scans will be immediately anonymized upon upload, with no PII stored at any stage. Only medical imaging data and radiologist annotations will be retained for retraining.
4. RAG knowledge base sources use only open-access and publicly licensed medical information

[3] Data Planning and Split

The data pipeline architecture encompasses two distinct processing pathways, each optimized for its specific downstream application: the disease prediction models requiring standardized image processing and the RAG system requiring structured text processing. This dual workflow ensures optimal data preparation while maintaining clear separation between the two tasks.

Prediction Models

Data Ingestion & Organization

1. Medical images (chest X-rays and CT scans) stored in Google Cloud Storage, organized chronologically (YYYY/MM/DD) and by disease category
2. Metadata (JSON/CSV) maps users to images; user data stored in PostgreSQL.

Preprocessing

1. Implement format Standardization (.png or.jpg) and resolution across all images to maintain consistent input format and shape
2. Include metadata about users for easy tracking and mapping.
3. Implement de-identification process and manage it to prevent data leaks/ privacy and compliance violations.
4. Perform essential quality checks (completeness, anomalies, anatomical coverage); failed scans are dropped.

Data Splitting

1. Stratified sampling: 75% train, 15% validation, 10% test.

Continuous Learning

1. Inference results fed back into raw data after de-identification.
2. Clinician review overrides model errors, creating high-confidence ground truth labels for retraining.

RAG Chatbot

Knowledge Base

1. Sources: peer-reviewed literature, research papers, journal articles, guidelines, drug databases, institutional protocols.
2. Organized by disease into dedicated folders for flexible updates.

Preprocessing

1. Standardize documents (structured PDFs/ HTML docs) into JSONL records allowing easy indexing.
2. Key metadata fields include - text, title, section, publication date, source, author.
3. Exclude non-citable docs, figures, and tables.
4. Additional focus will be spent on optimal chunking for semantic relevance and LLM context limits.

Vector Stores

1. Either a common index for all diseases or Disease-specific indices built from JSONL files (To be decided based on design choices and discussions).

2. Metadata-enabled retrieval improves filtering, citations, and reduces hallucinations.

[4] GitHub Repository

1. GitHub Repository: https://github.com/rjaditya-2702/MedScan_ai
2. README.md: https://github.com/rjaditya-2702/MedScan_ai/blob/main/README.md

[5] Project Scope

Problem

Our project aims to address the following critical problems in resource-constrained healthcare environments:

1) **Diagnostic Expertise Shortage**:
    a) Severe shortage of trained radiologists
    b) Healthcare facilities possess CT scanners and imaging equipment but lack specialist expertise to interpret scans leading to delays in diagnosis while expensive equipment remains underutilized
2) **Inefficient Workflow Processes**:
    a) Manual, time-intensive scan analysis and report generation processes
    b) Administrative burden requiring additional personnel for report creation, distribution, and patient communication
3) **Limited Decision Support and Transparency**:
    a) "Black box" AI solutions in healthcare face adoption resistance due to lack of explainable decision-making
4) **Poor Patient Access and Communication**:
    a) Patients cannot easily access their medical reports and scan history
    b) Limited patient education about diagnoses, with no accessible way to ask questions about their conditions.
    c) Dependency on healthcare staff for basic information about test results and medical terminology

Current Solutions

Existing approaches to address radiological challenges in resource-constrained settings include:

1. **Traditional Manual Workflow**: Radiologists manually interpret scans using paper or basic digital systems, which limits scalability and increases the risk of errors under high volume.
2. **Basic PACS Systems**: These systems provide digital storage and retrieval of images but lack integration with AI-based analysis, explainability features, patient portals, or automated reporting.
3. **Standalone AI Diagnostic Tools**: Commercial AI models address certain conditions (e.g., stroke), but are often isolated from clinical workflows, lack transparency in their decisions, and do not learn continuously from expert feedback.
4. **Telemedicine Consultation Services**: Remote radiologist consultations bridge some gaps, but remain expensive, limited in reach, and lack built-in mechanisms for systematic quality improvement.

Proposed Solutions

Our proposed solution transforms the traditional radiological workflow from medical scan analysis to patient report delivery, making specialized diagnostic expertise accessible where it is needed most. The platform serves two primary stakeholders: healthcare professionals working in challenging environments and patients who lack access to timely diagnostic services.

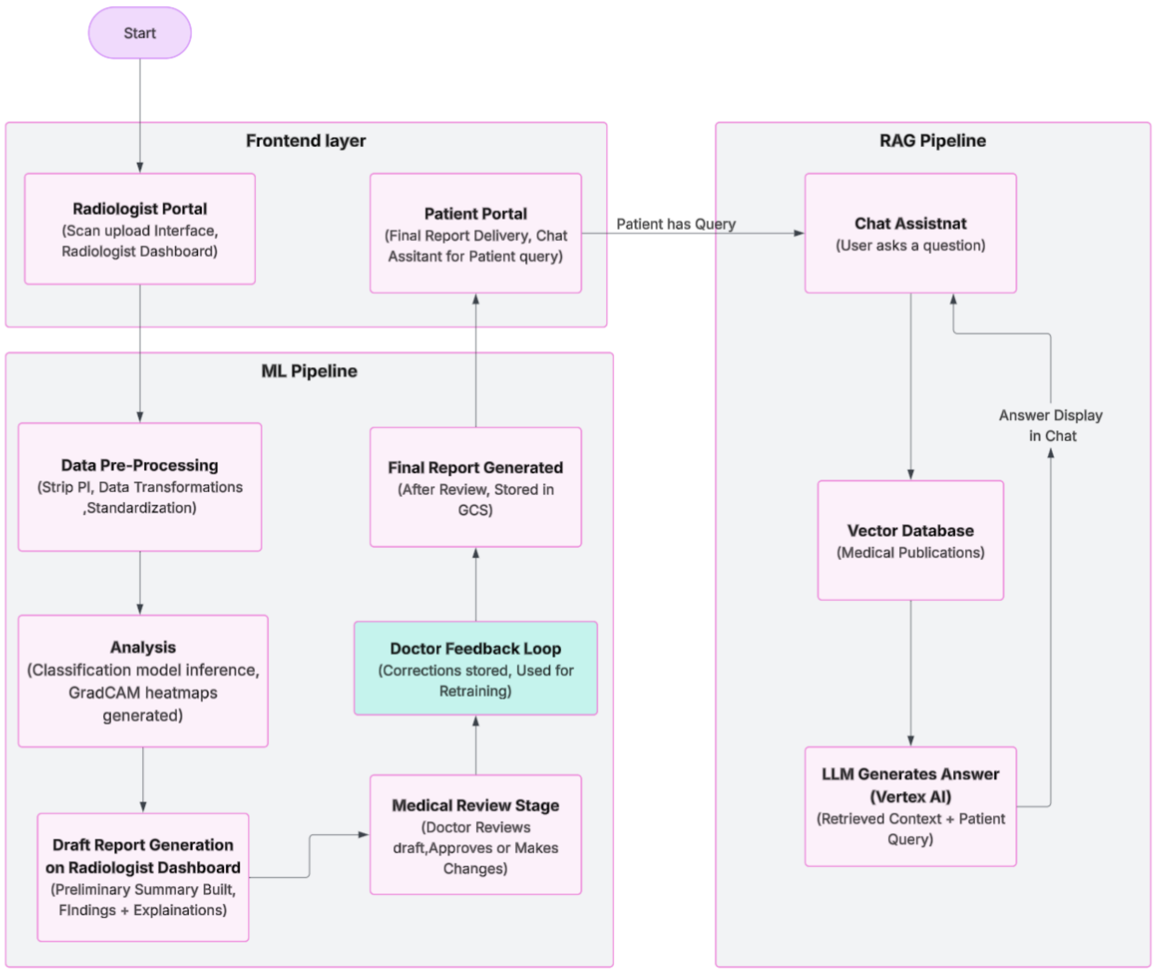
The workflow begins when a patient/ user gets a scan done at a lab or a hospital (a proxy system will be used to mimic the appointment scenario), which are automatically processed by our models to identify the specific anatomical region and classify the presence of lung cancer and Tuberculosis. To ensure diagnostic transparency, the system employs Grad-Cam visualization to explain the model's decision-making process, showing radiologists exactly which areas influenced the model's prediction. Following diagnosis, radiologists will have an option to override the model's prediction. This feedback, and images data feeds directly into our continuous retraining pipeline to detect model drift, data collection, and trigger retraining which improves the system's accuracy over time. The platform incorporates continuous learning capabilities through this feedback-driven approach, allowing the system to improve its accuracy based on real-world usage and local population characteristics.

The platform then generates comprehensive medical reports based on the AI analysis, which radiologists review, edit, and approve before publication. Once approved, reports are automatically published, and patients are notified through the system. On the patient side, users can access their complete scan history and reports through a dedicated portal, while a RAG-based chatbot provides instant answers to general questions about their disease, symptoms, and general medical information. The entire system maintains human oversight at critical decision points - radiologists retain full control over diagnostic accuracy and report content, while AI accelerates routine tasks and provides explainable decision support throughout the workflow.

To summarize, our solution introduces following comprehensive innovations:

1. End-to-End Automated Workflow Integration
2. Explainable AI with Grad Cam Transparency addresses "black box" problem by providing transparent, interpretable AI that builds trust.
3. Continuous Learning Through Human-AI Collaboration
4. Intelligent Report Generation
5. Comprehensive Patient Engagement Platform via RAG based chatbot

[6] Current Approach Flow Chart and Bottleneck Detection


Start
Frontend layer
Radiologist Portal
(Scan upload Interface, Radiologist Dashboard)
Patient Portal
(Final Report Delivery, Chat Assitant for Patient query)
Patient has Query
RAG Pipeline
Chat Assistnat
(User asks a question)
Answer Display in Chat
Vector Database
(Medical Publications)
LLM Generates Answer (Vertex AI)
(Retrieved Context + Patient Query)
ML Pipeline
Data Pre-Processing
(Strip PI, Data Transformations ,Standardization)
Analysis
(Classification model inference, GradCAM heatmaps generated)
Draft Report Generation on Radiologist Dashboard
(Preliminary Summary Built, Findings + Explainations)
Medical Review Stage
(Doctor Reviews draft,Approves or Makes Changes)
Doctor Feedback Loop
(Corrections stored, Used for Retraining)
Final Report Generated
(After Review, Stored in GCS)

Potential Bottlenecks

Image Processing

1. Bottleneck: Format conversion and quality control checks add latency, especially with large batch uploads during peak hours.
2. Solution: Using automated data pipelines and implementing parallel preprocessing using Cloud Functions with auto-scaling.

RAG Retrieval

1. Bottleneck: Risk of LLM generating factually incorrect medical information, creating patient safety concerns.
2. Solution: Include prominent disclaimers on all chatbot responses. Use medical journals to build RAG

RAG API Concurrency Limits

1. Bottleneck: Multiple API calls per query with concurrency limits can cause rate limit errors and slow responses during peak usage.
2. Solution: Implement request queuing with exponential backoff for rate-limited requests. Also, implement caching

Model Inference Speed

1. Bottleneck: Complex deep learning models requiring significant computational resources
2. Solution: Model optimization techniques (quantization, pruning)

Radiologist Review Process

1. Bottleneck: Single point of failure if radiologist is unavailable
2. Solution: Priority queuing system, multiple radiologist access, automated triage for urgent cases

Report Generation and Approval

1. Bottleneck: Manual report writing and editing process
2. Solution: Implement automated template generation for common diagnosis patterns.

[7] Metrics, Objectives, and Business Goals

Key Metrics

System Metrics

1. **Latency:** Ensure scan analysis and report generation occur within clinically acceptable turnaround times.
2. **Availability:** Maintain high uptime and reliability to support continuous hospital workflows.
3. **Throughput:** Ability to handle concurrent scans across multiple departments without performance degradation.
4. **Cost per Scan:** Optimize cloud and compute costs to ensure scalability without increasing per-patient expenses.

Model Evaluation Metrics:

1. **Prediction Model:** F1 score, accuracy, precision, recall, ROC-AUC, Grad-CAM alignment score
2. **RAG:** Answer hallucination score, average similarity of retrieved context documents

Objectives

1. Deliver accurate and explainable CT scan disease detection with human-in-the-loop validation to ensure reliability.
2. Generate patient-friendly reports enriched with transparent AI visualizations (e.g., Grad-CAM overlays) to support clear communication.
3. Continuously improve diagnostic reliability while reducing the risk of false negatives and misdiagnoses.

Business Goals

1. **Clinical Adoption:** Increase physician trust and adoption by emphasizing transparency, interpretability, and visual explanation tools.
2. **Patient Safety:** Strengthen diagnostic support to reduce misdiagnosis rates and improve patient outcomes.
3. **Scalability:** Provide a cloud-native, modular solution that can expand seamlessly from small pilot deployments to large institutional rollouts.

4. **Compliance & Safety:** Ensure full adherence to HIPAA and GDPR standards to protect patient privacy and support regulatory approval processes.

[8] Failure Analysis

In this section, we identify potential challenges that may occur during both the development and deployment phases of the product. The associated risks, their possible impacts, and corresponding mitigation strategies are examined. Within the scope of this project, we concentrate on the most fundamental and necessary mitigation measures

Failure Categories

Category	Problem	Risk	Impact	Mitigation
Image Data Quality/ ETL	Image Quality Issues	Poor quality medical scans	Misdiagnosis, false positives / negatives	• Implement quality gates with automatic rejection • Metadata validation • Multi-stage quality checks • Request re-upload protocols
	Data Format Incompatibility	Corrupted files, unsupported encodings, extensions or formats	Pipeline crashes, processing delays	• Multiple format parsers • Fallback conversion services • Graceful error handling with user feedback
	Patient Data Mix-up	Images attributed to wrong patient	CATASTROPHIC - Wrong treatment, legal liability	• Multiple-point verification • Unique ID marker in metadata to track patients and their scans, and reports
Model	Model Drift	Performance degradation over time as patient demographics change	Reduced accuracy, missed diagnoses	• Continuous monitoring with control datasets • A/B testing with champion/challenger models • Regular retraining schedules • Drift detection algorithms

	Out-of-Distribution (OOD) Samples	Model encounters scan types / diseases never seen before	Confident wrong predictions	• OOD detection mechanisms • Uncertainty quantification • Automatic flagging for manual review
	Bias and Fairness Issues	Model performs poorly on underrepresented demographics	Health disparities, legal issues, reputation damage	• Demographic performance monitoring • Balanced training sets • Fairness constraints in training • Regular bias audits
	Grad-Cam Misleading Visualizations	Heatmaps highlighting irrelevant regions convincingly	Doctor misled by explanations	• Multiple XAI methods (SHAP, LIME) • Sanity checks on gradients
Infrastructure	GCS Storage Failures	Data loss, unavailability, region outages	Complete system downtime	• Multi-region replication • Backup to different cloud provider • Local cache for recent uploads • SLA monitoring
	Vertex AI/Model Serving Outages	Model endpoint unavailable	No predictions possible	• Multi-zone deployment • Fallback to cached models on GKE • Queue-based retry logic
	Pub/Sub Message Loss	Tasks lost between upload and processing	Silent failures, missing reports	• Message persistence • Dead letter queues • Acknowledgment patterns • Duplicate detection
	Resource Exhaustion	GPU/CPU/Memory limits reached	Slow processing, timeouts	• Auto-scaling policies • Resource quotas • Priority queues

				• Load shedding protocols
	Network Failures	Inter-service communication failures	Partial processing, inconsistent state	Service mesh (Istio) Retry with exponential backoff Circuit breakers Timeout configurations
Security	Data Breaches	Unauthorized access to patient CT scans and reports	CATASTROPHIC - HIPAA violations, lawsuits, license loss	• Encryption at rest and in transit • VPC with private endpoints • Zero-trust architecture • Regular penetration testing
	Authentication/Authorization Bypass	Unauthorized users accessing system	Data theft, false reports generation	• Multi-factor authentication • OAuth 2.0/SAML integration • Role-based access control (RBAC) • Session management
Medical/ Clinical	Doctor Review Bottleneck	Insufficient doctors for review volume	Massive delays, SLA breaches	• Dynamic doctor pool scaling • Tiered review system • Automated triage • Overflow to partner networks
	Over-reliance on AI	Doctors rubber-stamping AI predictions	Automation bias, missed edge cases	• Randomized ground truth insertions • Performance monitoring per doctor • Regular training on AI limitations • Require written justifications
RAG Response	Hallucinations	LLM generating false medical information	Patient receives dangerous advice	• Retrieval-only mode for critical info • Fact-checking against medical databases • Confidence thresholds

			• Disclaimer requirements
Prompt Injection	Malicious prompts extracting training data or letting model generate harmful responses	Privacy violations, system manipulation, legal and trust issues	• Input filtering through guardrails • Prompt templates • Output validation • Rate limiting
Context Window Limitations	Important information truncated	Incomplete or wrong answers	• Chunking strategies • Hierarchical retrieval • Summary generation • Context prioritization

[9] Deployment Infrastructure

Note: The following deployment infrastructure represents a tentative plan based on current best practices and preliminary analysis. It is intended as an initial blueprint to guide development and will be revisited after detailed scoping, performance benchmarking, and cost–benefit evaluation to identify more optimized alternatives.

Our AI-driven patient–radiologist assistant will be deployed on **Google Cloud Platform** (GCP) using cloud-native, scalable services.

9.1 Data Ingestion and Preprocessing
Medical scans, patient metadata, and RAG documents will be ingested into **Google Cloud Storage (GCS)**. **Apache Airflow** (via **Cloud Composer**) will orchestrate data ingestion, preprocessing workflows, and ETL jobs.

9.2 Model Training and Experimentation
Vertex AI will handle distributed training and hyperparameter tuning using cost-effective **NVIDIA Tesla T4 or P100 GPUs**. **Vertex AI Workbench** will provide Jupyter Notebooks for experimentation. **DVC (Data Version Control)** will track datasets and model versions for reproducibility.

9.3 Model Registry

Vertex AI Model Registry will serve as the centralized repository for model versioning, tracking metadata including training datasets, hyperparameters, and performance metrics.

9.4 Model Deployment and Serving

FastAPI will power the REST API, containerized with **Docker** and deployed on **Cloud Run** for serverless auto-scaling, with **Google Kubernetes Engine (GKE)** as an alternative for complex orchestration needs. GPU-intensive inference will use **Compute Engine** with **Tesla T4 GPUs**. The RAG pipeline will integrate open-source LLMs from **Hugging Face** or **OpenAI APIs** via **LangChain**.

9.5 Storage

GCS will store DICOM scans, preprocessed images, and model artifacts. **Cloud SQL for PostgreSQL** will manage structured patient metadata and logs. **Pinecone** will serve as the vector database for RAG semantic retrieval, with **PGVector** as a cost-effective alternative.

9.6 Monitoring and Logging

Cloud Logging and **Cloud Monitoring** will track API requests, inference latency, and system health. **Prometheus** and **Grafana** will provide optional fine-grained monitoring for model performance metrics and custom dashboards.

9.7 CI/CD

GitHub Actions will automate the complete deployment pipeline: code linting, unit testing, Docker image builds, and deployment to **Cloud Run** and **GKE**. CI/CD workflows will integrate **DVC** to automatically trigger model retraining when new annotated datasets are committed. Successfully trained models will be pushed to **Vertex AI Model Registry**and deployed to production endpoints with automated rollback mechanisms for failed deployments.

9.8 Retraining Pipeline

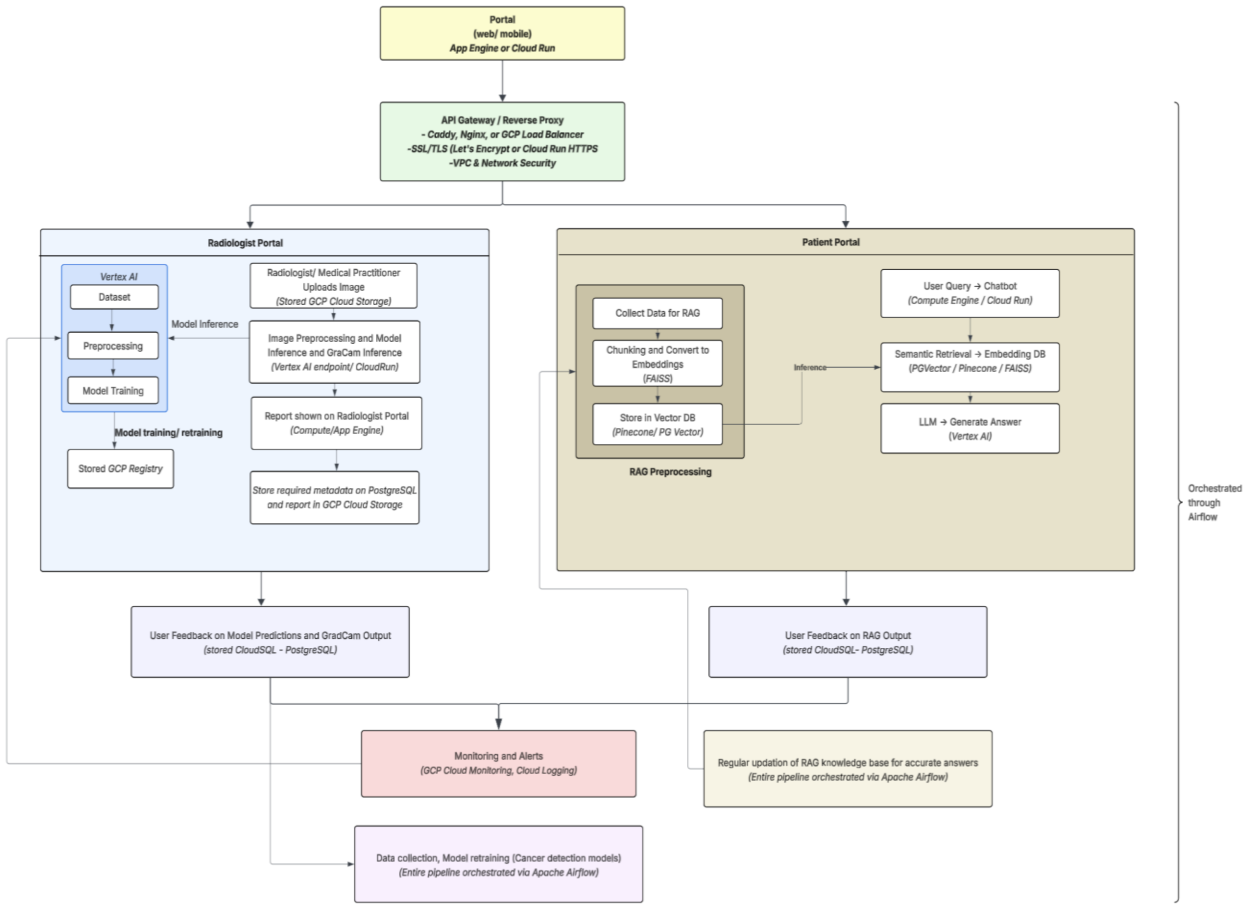
Apache Airflow (Cloud Composer) will orchestrate automated retraining pipelines triggered by data drift detection or scheduled intervals. DAGs will manage data validation, preprocessing, Vertex AI training, evaluation, and deployment.

9.9 Security and Networking (not part of MVP)

Services will operate within a **VPC** with **Cloud NAT** for secure access. **IAM** will enforce role-based access control. Data at rest will use default GCP encryption, while **Cloud Run's managed SSL certificates** will secure HTTPS endpoints with **Cloud Load Balancing** managing traffic. Automated **Cloud SQL backups** with point-in-time recovery and **GCS versioning** will ensure disaster recovery.

9.10 Frontend

Frontend portals will be built using **React** or **Streamlit** based on prototyping results, prioritizing React for production-grade patient/radiologist interfaces and Streamlit for rapid internal dashboards. The application will be deployed on **Cloud Run**, with static assets served from **GCS** via **Cloud CDN** for low-latency delivery.


Portal
(web/ mobile)
App Engine or Cloud Run
API Gateway / Reverse Proxy
- Caddy, Nginx, or GCP Load Balancer
-SSL/TLS (Let's Encrypt or Cloud Run HTTPS
-VPC & Network Security
Radiologist Portal
Vertex AI
Dataset
Preprocessing
Model Training
Model Inference
Model training/ retraining
Stored GCP Registry
Radiologist/ Medical Practitioner Uploads Image (Stored GCP Cloud Storage)
Image Preprocessing and Model Inference and GraCam Inference (Vertex AI endpoint/ CloudRun)
Report shown on Radiologist Portal (Compute/App Engine)
Store required metadata on PostgreSQL and report in GCP Cloud Storage
Patient Portal
Collect Data for RAG
Chunking and Convert to Embeddings (FAISS)
Store in Vector DB (Pinecone/ PG Vector)
RAG Preprocessing
Inference
User Query → Chatbot (Compute Engine / Cloud Run)
Semantic Retrieval → Embedding DB (PGVector / Pinecone / FAISS)
LLM → Generate Answer (Vertex AI)
Orchestrated through Airflow
User Feedback on Model Predictions and GradCam Output (stored CloudSQL - PostgreSQL)
User Feedback on RAG Output (stored CloudSQL- PostgreSQL)
Monitoring and Alerts (GCP Cloud Monitoring, Cloud Logging)
Regular updation of RAG knowledge base for accurate answers (Entire pipeline orchestrated via Apache Airflow)
Data collection, Model retraining (Cancer detection models) (Entire pipeline orchestrated via Apache Airflow)

[10] Monitoring Plan

The monitoring framework ensures the reliability, performance, explainability, and compliance of both AI models and the RAG pipeline. This involves tracking **system-level health metrics, model inference metrics, data quality, and feedback pipelines,** enabling early detection of anomalies, model drift, and operational issues. The monitoring architecture leverages **GCP Cloud Monitoring, Cloud Logging, and optional experiment tracking tools such as Weights & Biases or MLflow to provide full observability across system,** model, and RAG pipeline metrics. Cloud Monitoring collects system health metrics, training pipeline metrics, inference metrics, and RAG pipeline statistics, while Cloud Logging centralizes application and infrastructure logs for detailed auditability. Alerting policies are configured for CPU/GPU saturation, API errors, inference accuracy drops, and anomalies in RAG feedback, with notifications delivered via email, SMS, or Pub/Sub triggers to ensure rapid incident response. Cloud Monitoring dashboards visualize metrics in real-time and provide drill-down capabilities, enabling detailed analysis from infrastructure-level performance to individual model-level insights.

Weights & Biases or MLflow is integrated to log fine-grained experiment data, including training and evaluation metrics, hyperparameters, dataset versions, and Grad-Cam visualizations, ensuring reproducibility of experiments and traceability of model performance. Incorporating MLOps-specific metrics such as model accuracy, precision, recall, data drift, and Grad-Cam correctness provides several benefits. It enables early detection of model degradation, allowing teams to spot accuracy drops or data drift before they impact clinical decisions. It ensures regulatory compliance by documenting model performance, explainability, and data handling practices, which is essential in healthcare. It promotes operational continuity by proactively monitoring system, pipeline, and model health to reduce downtime, supports continuous improvement by integrating feedback loops to adapt models to changes in input data and clinical workflows, and enhances explainability and trust, providing radiologists with confidence in AI outputs.

This framework ensures end-to-end observability for all stages of the AI lifecycle—from raw data ingestion, model training, inference, RAG outputs, and feedback integration to overall system performance. By tightly integrating system health, model performance, RAG pipeline metrics, and feedback ingestion using GCP-native monitoring and logging services, the strategy maintains operational, clinical, and regulatory standards, ensuring that the AI system remains reliable, transparent, and accountable.

Category / Subcategory	Metric	Why Monitor	Tools / Methods

System Health Metrics	CPU usage	Detect system bottlenecks affecting inference or training	**Cloud Monitoring VM metrics**
	GPU usage / memory	Ensure inference/training efficiency and prevent resource contention	**Cloud Monitoring GPU metrics / NVIDIA DCGM Exporter (optional)**
	RAM utilization	Prevent crashes due to insufficient memory	**Cloud Monitoring VM metrics**
	Disk I/O & storage	Detect slow read/write impacting data pipelines	**Cloud Monitoring disk metrics**
	API latency & throughput	Ensure real-time performance of inference & RAG APIs	**Cloud Monitoring / Cloud Trace**
	Error rates / exceptions	Early detection of software or pipeline failures	**Cloud Logging + Alerting policies**
	Uptime / availability	SLA adherence and operational reliability	**Cloud Monitoring uptime checks**
Model Performance Metrics	Accuracy, Precision, Recall, F1-score	Evaluate inference quality and detect model degradation	**Batch evaluation on validation sets; W&B or Vertex AI Model Monitoring**
	ROC-AUC / PR-AUC	Measure discrimination performance	**Periodic evaluation on holdout sets; Vertex AI metrics**
	Grad-Cam/ Explanation Correctness	Maintain explainability for clinical trust	**Custom scripts exporting explanations; W&B logging**
	Inference latency per scan	Ensure timely decision-making	**Cloud Monitoring custom metrics / Vertex AI endpoint metrics**
	Data drift / distribution shift	Identify changes in input data that can degrade model performance	**Statistical divergence metrics (KL divergence, PSI); Vertex AI Model Monitoring**
RAG Pipeline Metrics	Retrieval accuracy	Ensure the retrieved context matches queries	**Evaluate with known QA pairs or embedding similarity**
	RAG answer correctness	Detect hallucinations or inconsistencies	**Automated scoring or human feedback integration**

	Feedback ingestion rate	Ensure real-time integration of corrections	**Cloud Monitoring custom counters / Pub/Sub metrics**
	Alert on critical feedback	Immediate operational action	**Cloud Monitoring alerting policies (email/SMS/Slack)**
	Embedding cache hit/miss rates	Ensure efficient retrieval	**Cloud Monitoring custom metrics**
	Unauthorized access attempts	Ensure safety of the application	**Cloud Logging audit logs, Security Command Center / SIEM**
	Data drift detection	Detect shifts in patient questions overtime	**Statistical tests (Kolmogorov-Smirnov, PSI)**
	Data latency / ingestion health	Ensure pipelines process scans in real-time	**Cloud Monitoring metrics / Cloud Logging**
Training Pipeline Metrics	Epoch loss / accuracy	Detect underfitting, overfitting, or training failures	**Vertex AI Training job metrics; W&B or MLflow**
	Training duration & throughput	Optimize resource allocation	**Vertex AI Training job metrics**
	Experiment metadata / hyperparameters	Ensure reproducibility	**Weights & Biases or MLflow**

[11] Success and Acceptance Criteria

Prediction Model

1. F1 of 0.8 or above
2. Human in the loop – eval over sample test images – approve the prediction and Grad-Cam output

RAG

1. Human in the loop eval – check for tone consistency and answer relevance with source citations
2. Average similarity scores of retrieved contexts – 0.9 and above
3. Hallucination score (check how grounded the answer is within the context provided) - 0.7 and above

System

1. Latency – less than 1 second for disease prediction at inference
2. RAG – 15 seconds or less (end-to-end: from query to response)

[12] Timeline Planning

Phase	Name	Oct 1–7	Oct 8–14	Oct 15–21	Oct 22–28	Oct 29–Nov 4	Nov 5–11	Nov 12–18	Nov 19–25	Nov 26–Dec 2	Dec 3–9	Dec 10-12
1	Planning & Setup	🟩										
2	Core Model Dev (Detection + Explainability + Report Gen)		🟩	🟩	🟩	🟩						
3	Feedback Loop & Workflow Integration				🟩	🟩	🟩	🟩				
4	Patient Portal + RAG Chatbot		🟩	🟩	🟩	🟩	🟩	🟩				
5	Testing, Validation & Refinement								🟩	🟩	🟩	
6	Final Polish & Demo Prep											🟩

[13] Additional Information

AI for Healthcare & Trust

Trust in AI for healthcare is essential and shapes the future of medical practice, yet skepticism remains high among patients and providers due to concerns about reliability, bias, data privacy, transparency, and loss of human connection in care. Barriers include fears about accountability for AI errors, discrimination from biased data, and opaque decision-making, with patients wanting AI to support—not replace—clinicians while providers worry about liability. Building trust requires transparency about AI capabilities and limitations, robust governance, bias mitigation, human-centered design that preserves autonomy and empathy, and meaningful engagement of patients and clinicians in development. Trust is bidirectional—AI systems depend on quality human oversight while users must have confidence in AI tools—and failure to build it risks undermining not only AI adoption but confidence in healthcare institutions themselves.

Hallucination Detection

Vectara's Hughes Hallucination Evaluation Model (HHEM) is a state-of-the-art system that detects hallucinations in LLM outputs using a dedicated classification model rather than "LLM-as-a-judge" approaches, making it faster, more reliable, and cost-effective for sensitive domains where accuracy and trust are critical. HHEM-2.3, the commercial version, provides an interpretable Factual Consistency Score (0-1 probability, e.g., 0.98 = 98% factual accuracy) indicating how well responses align with context and is optimized for production RAG pipelines and mission-critical deployments, while the open-source HHEM-2.1-Open supports multiple languages and is freely available via Hugging Face. The model powers Vectara's Hallucination Leaderboard that benchmarks LLMs by hallucination rates and is widely used across research and enterprise to validate AI outputs, ensuring transparency and reducing risks in real-world applications.

GRAD-CAM

Grad-CAM (Gradient-weighted Class Activation Mapping) explains how CNNs make decisions by generating heatmaps that highlight which image regions were most important for predictions, transforming black-box models into interpretable systems. It computes gradients of class scores relative to the last convolutional layer's feature maps, weights them by importance, and overlays the resulting localization map on the original image to show where the model focused. This technique helps humans understand and trust AI by revealing when models rely on irrelevant details—aiding debugging and bias detection—and works with many CNN architectures without requiring retraining, making it especially valuable in high-stakes domains like healthcare and autonomous driving where human-interpretable explanations are critical.

Cost Estimation

Data Storage & Management

1. **Google Cloud Storage (GCS):** $0.30/month for raw DICOM scans, preprocessed images, and RAG documents (~5-7 GB)
2. **Cloud SQL for PostgreSQL:** $7.00/month (db-f1-micro) for patient metadata, user data (patient portal)
3. **PGVector (included in Cloud SQL):** $0/month for RAG vector embeddings and semantic retrieval

Model Development & Training

1. **Vertex AI Workbench:** $15.00/month for Jupyter notebooks, data exploration, and experimentation
2. **Vertex AI Training:** $6.00/month for training tuberculosis and lung cancer detection models using preemptible Tesla T4 GPUs (increased allocation for faster iterations)
3. **Vertex AI Model Registry:** $0/month (included) for model versioning, metadata tracking, and lineage management
4. **DVC (Data Version Control):** $0/month for dataset and model version tracking (uses GCS storage already accounted for)

Model Serving & APIs

1. **Cloud Run:** $12.00/month for FastAPI REST API, RAG chatbot backend, and frontend hosting (200K requests/month)
2. **Compute Engine (GPU inference):** $25.00/month for GPU-intensive medical image analysis using preemptible Tesla T4 instances (~80 hours/month)
3. **Google Kubernetes Engine (GKE):** $0/month (deferred for MVP; Cloud Run sufficient for current scale)

Orchestration & CI/CD

- **Cloud Functions:** $3.00/month for data ingestion triggers and preprocessing automation
- **Cloud Scheduler:** $0.10/month for scheduled retraining triggers and pipeline jobs (3 jobs)
- **GitHub Actions:** $0/month for CI/CD automation, Docker builds, and deployments (2,000 minutes/month free tier)

LLM & RAG Pipeline

1. **Hugging Face Models:** $0/month for open-source LLM inference (self-hosted)
2. **OpenAI API:** $8.00/month for GPT-based RAG responses for higher quality outputs (estimated 15K tokens/day)
3. **LangChain:** $0/month (open-source library for prompt orchestration and retrieval workflows)

Monitoring & Logging

1. **Cloud Monitoring:** $5.00/month for system health metrics, API performance, model drift detection
2. **Cloud Logging:** $4.00/month for application logs, inference tracking, audit trails, and error analysis
3. **MLflow (self-hosted):** $10.00/month for experiment tracking, hyperparameter logging, and Grad-CAM visualizations (e2-small VM)
4. **Prometheus & Grafana:** $8.00/month for custom dashboards, fine-grained model performance metrics, and real-time alerting (e2-micro VM)

Networking & Security

1. **Cloud NAT:** $2.00/month for secure outbound connectivity from VPC
2. **Cloud Load Balancing:** $8.00/month for HTTPS endpoints, SSL certificate management, and traffic distribution
3. **Cloud CDN:** $2.00/month for static asset delivery and low-latency frontend performance

Frontend Development

1. **React/Streamlit hosting:** $0/month (included in Cloud Run estimate above)