LabLens: AI-Powered Health Report Simplification

Project Team 14

1. Asad Ullah Waraich
2. Dhruv Rameshbhai Gajera
3. Mihir Harishankar Parab
4. Shahid Kamal
5. Shruthi Kashetty
6. Sri Lakshmi Swetha Jalluri

Table of Contents

1. Introduction

Every day, millions of patients are confused, anxious, and overwhelmed by their own medical results. Imagine a patient opening a lab report to see "ALT: 85 U/L" or "Neutrophils: 78%" flagged in red. This dense medical jargon and cryptic data act as an immediate barrier, preventing patients from understanding the urgency of their health status. This confusion leads to delayed treatment, unnecessary anxiety, and poor health decisions, turning patients into passive, ill-informed recipients of their own health data.

We are solving this comprehension crisis with an ML-powered platform that transforms confusing medical reports into clear, actionable health intelligence. Our primary mission is to break down the language and complexity barriers through instant summarization and multilingual support. Our solution is built on a structured two-stage framework that demystifies lab results and empowers patients with essential understanding:

First, Clarity through Intelligent Summarization: We fundamentally eliminate the jargon barrier by instantly translating complex medical terms—converting "elevated hepatic transaminases" into "your liver enzymes are higher than normal"—ensuring the patient grasps the basic findings. Our platform currently supports English and Hindi, addressing the needs of over 600 million Hindi speakers globally, with plans to expand to additional languages as we scale. This bilingual capability ensures that language barriers don't prevent patients from understanding their health data.

Second, Context through Risk Analysis: By analyzing result patterns, we highlight potential risk indicators and explain what these findings commonly suggest—such as clarifying that elevated white blood cells often indicate the body is fighting an infection. This provides patients with essential context about the significance of their results, helping them understand whether findings require urgent attention or routine follow-up, all while maintaining clear boundaries against providing medical diagnoses.

This framework converts complex documents into powerful educational tools. We are not just creating simpler text; we are fundamentally changing how patients interact with their own medical data, enabling them to become informed participants in their health journey. While our current focus is on delivering clear summaries and risk context, our roadmap includes expanding language support and potentially adding features like specialist recommendations based on user feedback and clinical validation. Through this focused approach, we aim to reduce patient anxiety, improve health literacy, and ultimately contribute to better healthcare outcomes.

Business Goals and Values:

- **Large Underserved Market**: Millions of patients struggle to understand medical reports daily, with English and Hindi speakers representing a significant global population that lacks access to simplified medical information
- **Multiple Revenue Streams**: Healthcare providers need solutions to improve patient satisfaction and reduce support costs, while diagnostic labs seek differentiation and patients want affordable tools to understand their health
- **Operational Efficiency**: Reduces time doctors spend explaining basic terminology, decreases panic-driven healthcare visits, and helps patients prepare better questions for consultations, creating value across the healthcare system
- **Competitive Advantage**: Building bilingual medical AI requires specialized expertise in both healthcare terminology and language processing, creating natural barriers for competitors and significant technical challenges to replicate
- **Scalable Technology Platform**: Once core summarization works, the platform can expand to additional languages, integrate with existing healthcare systems, and add

features based on user needs, enabling growth from pilot programs to enterprise deployments

2. Dataset Information

2.1 Dataset Introduction

We utilize the MIMIC-III database and MedMNIST-ChestMNIST collection, balancing comprehensive clinical data with CPU-optimized image processing capabilities. These datasets provide:

MIMIC-III: 2 million+ clinical notes including 59,652 discharge summaries for text summarization, with real-world medical complexity from 46,520 patients over 11 years

MedMNIST-ChestMNIST: 112,120 pre-processed 28×28 chest X-rays optimized for CPU training, with 14 pathology labels matching clinical standards

Integration Strategy: MIMIC-III provides gold-standard clinical text while MedMNIST enables feasible image classification on CPU-only infrastructure

2.2 Data Card

Dataset	Size	Format	Data Type
MIMIC-III v1.4	50 GB compressed, 100 GB uncompressed	26 CSV tables	• Text: Clinical notes, discharge summaries • Structured: ICD-9 codes, lab results, medications • Temporal: Timestamps for all events
MedMNIST-ChestMNIST	~100 MB	NPZ/PNG format, Pre-split datasets	• Images: 112,120 chest X-rays (28×28 pixels) • Labels: 14 pathology conditions • Metadata: Train/val/test splits included
Indian Medicine Data (Kaggle)	~1–2 MB (CSV, ~6,000+ entries)	CSV format	• Structured: Medicine name, category (Ayurvedic, Unani, Homeopathy) • Uses/indications (disease

			mapping) • Metadata: Source tags
Drugs@FDA & Drug Approvals	Continuously updated (hundreds of MBs)	Online relational database, XML/CSV	• Structured: Drug approval history, application numbers • Text: FDA drug labeling, prescribing information • Metadata: Therapeutic equivalence codes, marketing status

Combined Statistics:

- 59,652 discharge summaries for text summarization
- 112,120 chest X-rays optimized for CPU processing (5-30 minute training)
- 46,520 unique patients (MIMIC-III)
- Temporal coverage: 2001-2012 (MIMIC-III)
- ~6,000+ medicine entries with categories (Ayurvedic, Unani, Homeopathy) and mapped indications
- Thousands of FDA-approved drugs with detailed approval history and prescribing information

2.3 Data Sources

- **MIMIC-III:** PhysioNet Repository (https://physionet.org/content/mimiciii/1.4/)
- **MedMNIST-ChestMNIST:** https://huggingface.co/datasets/albertvillanova/medmnist-v2
- **Indian Medicine Data (Kaggle):** https://www.kaggle.com/datasets/mohneesh7/indian-medicine-data
- **Drugs@FDA & Drug Approvals:** https://www.fda.gov/drugs/development-approval-process-drugs/drug-approvals-and-databases

2.4 Data Rights and Privacy

MIMIC-III:

- Fully de-identified per HIPAA Safe Harbor provisions
- IRB approved by MIT and Beth Israel Deaconess Medical Center

- Requires signed Data Use Agreement (DUA)
- No re-identification attempts permitted
- Mandatory citation of dataset paper

MedMNIST-ChestMNIST:

- CC BY 4.0 license - permits academic and commercial use
- Pre-processed from multiple public sources (NIH ChestX-ray14, CheXpert)
- Already de-identified in source datasets
- No additional privacy restrictions beyond attribution

Indian Medicine Data (Kaggle):

- Openly available on Kaggle
- Intended primarily for academic and research purposes
- Subject to Kaggle's dataset terms of use
- Contains no personal health records (non-PHI data)
- Already anonymized since it only deals with drug/medicine metadata
- Attribution required to the dataset creator on Kaggle

Drugs@FDA & Drug Approvals:

- Public domain (U.S. Government data)
- Permits both academic and commercial use without restriction
- No personal data included (only regulatory and product information)
- No IRB approval or data use agreement required
- Standard citation of FDA Drugs@FDA recommended when using the database

3. Data Planning and Splits

Data Preprocessing Pipeline

Phase 1: Text Data (MIMIC-III)

1. Extract discharge summaries from NOTEEVENTS table:
   - Filter: CATEGORY = 'Discharge summary' (59,652 notes)
   - Remove de-identification markers [**]
   - Parse sections using regex patterns
2. Data splits (by patient ID to prevent leakage):
   - Train: 70% (41,756 notes)

    - Validation: 15% (8,948 notes)
    - Test: 15% (8,948 notes)

Phase 2: Image Data (MedMNIST-ChestMNIST)

1. Load pre-processed 28×28 images:
    - Direct numpy array loading
2. Utilize provided splits:
    - Train: 78,468 images (70%)
    - Validation: 11,219 images (10%)
    - Test: 22,433 images (20%)
    - Pre-stratified by pathology distribution

4. Github Repository

https://github.com/asad-waraich/mlops-group14

5. Project Scope

Problem Statement

The current process of understanding medical test results presents a significant comprehension barrier for patients who receive reports filled with complex medical terminology, abbreviations, and cryptic numerical values. Medical laboratories and hospitals generate reports using technical language designed for healthcare professionals, making them virtually inaccessible to patients who need to understand their own health data.

When patients access these reports through online portals or receive them directly from labs, they lack the tools to interpret whether findings are concerning or what the results mean for their health. This knowledge gap creates widespread patient anxiety, confusion about health status, and difficulty in making informed decisions about their medical care.

Current Situation

Patients receiving medical reports must navigate multiple challenges to understand their health information. They often resort to unreliable internet searches that provide generic or alarming information without proper context, struggle to identify which findings are significant versus routine variations, or remain passive in their healthcare journey due to inability to comprehend their medical data.

Without a structured framework to interpret results, patients cannot determine the urgency of their findings or understand the implications of abnormal values. The complexity of medical terminology creates barriers for all patients—English speakers struggle with technical medical jargon even in their native language, while Hindi speakers face the dual challenge of both medical complexity and language translation, as most medical reports are generated only in English.

Healthcare providers frequently spend valuable consultation time explaining basic terminology and test interpretations rather than focusing on treatment strategies and preventive care. This inefficient information transfer affects healthcare quality, patient satisfaction, and the overall patient-provider relationship.

Proposed Solution

Implementing an ML-powered platform using advanced NLP models will transform how patients interact with their medical data through our focused two-stage framework: Clarity and Context.

The platform will instantly process uploaded medical reports, generating plain-language summaries that break down complex medical terminology into simple, understandable explanations. For English-speaking users, the system will convert technical medical jargon like "hyperlipidemia" into clear language such as "high cholesterol levels in your blood." For Hindi-speaking users, the platform will not only simplify but also translate these complex terms into accessible Hindi, transforming "elevated hepatic enzymes" into "आपके लिवर एंजाइम सामान्य से अधिक हैं" (your liver enzymes are higher than normal).

Using transformer-based models fine-tuned on medical datasets, the system addresses the unique needs of both language groups—helping English speakers understand medical complexity in simple English, while providing Hindi speakers with culturally appropriate translations that maintain medical accuracy. The platform analyzes result patterns to identify potential risk indicators, explaining what abnormal values might mean and whether findings require urgent attention or routine follow-up, all while maintaining clear boundaries against providing medical diagnoses.

This focused solution empowers patients regardless of their language preference, ensuring that medical complexity doesn't become a barrier to understanding one's own health. While our current scope concentrates on English and Hindi summaries with risk context, our architecture is designed to accommodate future enhancements such as additional language support and advanced features based on user feedback and clinical validation.

6. Current Approach Flow Chart and Bottleneck Detection

End User Flow

1. User accesses the medical report platform through web browser (Streamlit interface)
2. User uploads medical documents (PDF lab reports or photos of physical reports)
3. System validates file format and size, providing immediate feedback if issues detected
4. Upon successful upload, user selects preferred output language (English or Hindi)
5. User clicks "Analyze Report" button to initiate processing
6. Real-time progress indicator shows processing stages (Uploading → Analyzing → Generating Summary)
7. System displays results in two sections: Summary and Risk Indicators
8. Translation toggle allows switching between English and Hindi versions instantly
9. User can download formatted report as PDF or share via secure link
10. User can save results to personal dashboard for future reference

End-to-End Pipeline Flow

1. Medical documents are ingested through API endpoint and validated for supported formats
2. Text-based reports (PDF) are processed through PyPDF2 for text extraction
3. Image-based inputs (photos of reports) are processed through OCR (Tesseract/Google Vision API)
4. Text pipeline applies medical entity recognition to extract lab values and diagnoses
5. T5/BART model generates plain language summary from extracted medical entities
6. Risk analysis engine compares values against normal ranges and identifies patterns
7. If Hindi translation required, Google Translate API or mT5 converts the summary
8. Results are formatted as JSON response
9. Frontend receives structured data and renders visualization
10. All processing logs are sent to Cloud Monitoring for performance tracking

GCP Deployment

- Docker containers created with model artifacts and dependencies
- Images pushed to Google Artifact Registry with version tags
- Cloud Run used for serverless deployment (cost-effective for MVP)
- Model weights stored in Cloud Storage buckets
- Basic auto-scaling configured based on request rate
- Cloud Monitoring tracks latency and throughput metrics
- Budget alerts set to control costs

Bottleneck Detection and Mitigation

Text Extraction: OCR may fail on poor quality images

- Mitigation: Image preprocessing, user feedback for re-upload

Model Inference: T5/BART summarization may be slow

- Mitigation: Model optimization, caching common patterns

Translation Accuracy: Medical terms may translate poorly

- Mitigation: Custom medical dictionary, post-processing validation

Concurrent Users: System may slow with multiple users

- Mitigation: Request queuing, horizontal scaling

Memory Usage: Large models may cause OOM errors

- Mitigation: Model quantization, batch size optimization

API Rate Limits: Google Translate API has usage limits

- Mitigation: Request batching, fallback to local model

Response Time: Target <10 seconds per report

- Mitigation: Async processing, progress indicators

Cost Control: GCP costs may escalate

- Mitigation: Budget alerts, usage quotas, CPU-only inference where possible

7. Key Metrics, Objectives

Key Metrics

- Translation Accuracy: Percentage of medical jargon correctly converted to plain language
- Summary Quality: Clinical validation score for report summarization accuracy
- Patient Comprehension Improvement: Pre/post-platform understanding score increase
- User Satisfaction: Average patient rating (1-10 scale)
- Anxiety Reduction: Percentage decrease in patient-reported anxiety
- Response Time: Average seconds to process and generate reports

- Platform Adoption: Number of reports processed per day/week/month
- User Retention: Percentage of patients returning for repeat use
- Consultation Efficiency: Minutes saved per provider visit
- System Uptime: Platform availability percentage
- Processing Success Rate: Percentage of reports analyzed without errors

Objectives

Primary Objectives:

- Transform complex medical reports into clear, plain-language summaries that patients can easily understand
- Provide instant bilingual support (English and Hindi) to eliminate language barriers in healthcare comprehension
- Reduce patient anxiety and empower informed health decisions through contextualized explanations
- Enable patients to become active participants in their healthcare through better health literacy

Secondary Objectives:

- Decrease provider consultation time by pre-educating patients on basic medical terminology and findings
- Improve quality of patient-provider conversations through better-prepared and informed patients
- Increase health equity by serving diverse populations including Hindi speakers and underserved communities
- Build a scalable platform architecture that can accommodate future feature additions

Technical Objectives:

- Achieve >80% accuracy in medical terminology simplification and report summarization
- Deliver real-time report processing with response times under 10 seconds per report
- Maintain high system reliability with >95% uptime for consistent patient access
- Support accurate bilingual medical processing with >85% quality scores for both languages

8. Failure Analysis

During Project Development

Data Quality Issues: Lab reports may contain missing values, inconsistent formats, or outdated reference ranges, leading to confusing or inaccurate summaries. *Mitigation:*

- Perform preprocessing to detect and flag missing or invalid entries instead of trying to interpret them
- Normalize units and test names into a consistent internal format
- Regularly review and update reference ranges using trusted medical sources to keep results relevant

Infrastructure and Resource Constraints: Training and serving ML models that process medical text and translations may demand high compute and storage resources. *Mitigation:*

- Use scalable cloud services (GCP) for training and inference
- Optimize preprocessing workflows and use lightweight inference models for production
- Implement model quantization and pruning techniques

Pipeline Failures: Breakdowns could occur between stages (data ingestion, ML model inference, translation, summary generation, and user interface). *Mitigation:*

- Write unit tests for each component and run end-to-end integration tests
- Automate builds and deployments using CI/CD pipelines (GitHub Actions)

During Deployment

API Downtime or Latency: The backend or ML inference APIs may slow down under heavy load, leading to delayed results for patients. *Mitigation:*

- Set up load balancing, caching, and failover systems
- Monitor system health with uptime and latency alerts

Model Drift: Over time, the model may misinterpret results as lab practices or medical reference ranges evolve. *Mitigation:*

- Retrain models regularly with updated medical guidelines and lab ranges
- Use drift detection tools to monitor performance and trigger retraining

Post Deployment

Inaccurate or Misleading Summaries: The platform may simplify lab data incorrectly, leading to patient anxiety or false reassurance. *Mitigation:*

- Add disclaimers and encourage follow-ups with healthcare professionals
- Route uncertain or complex cases to human support or a flagged review system

Scalability Issues: Large user traffic may overload servers and slow responses. *Mitigation:*

- Use Cloud Run auto-scaling to support traffic spikes
- Monitor real-time traffic and auto-adjust resources

9. Deployment Infrastructure

Cloud Platform Google Cloud Platform (GCP): Primary cloud provider for hosting services

Containerization

- Docker: For packaging the application into containers to ensure consistent deployments
- Cloud Run: For serverless container deployment with automatic scaling

Frontend Streamlit-based frontend for patient interaction and report visualization

ML Model Management MLflow: For tracking experiments, packaging code, and deploying models that perform medical jargon translation and risk interpretation

Core Framework

- **Clarity Module:** Summarizes and translates complex medical terms into patient-friendly explanations in their preferred language (English/Hindi)
- **Context Module:** Analyzes result patterns to highlight common risk indicators (e.g., "elevated white blood cells often mean your body is fighting an infection")

Continuous Integration/Continuous Deployment (CI/CD) GitHub Actions: To automate testing, building, and deployment of the application

Deployment Pipeline:

- Build Docker images
- Run unit and integration tests on summarization and translation modules
- Deploy to staging and production environments via Cloud Run

Supported Platforms Web browsers (desktop and mobile responsive)

10. Monitoring Plan

We will implement a multi-layer monitoring plan to ensure the platform remains reliable, accurate, and user-friendly after deployment.

Performance Metrics

- Response time: Ensure fast generation of summaries and risk indicators
- Query success rate: Percentage of reports correctly summarized
- Translation accuracy: Quality of English-Hindi conversions

User Interactions

- Logs of uploaded reports, summaries provided, and user feedback
- Frequency of fallback cases (when the platform cannot process a report)
- User satisfaction metrics (thumbs-up/down, ratings)

Model Metrics

- Accuracy and relevance of plain-language summaries
- Drift detection on medical terminology and lab code distributions
- Frequency and performance of retrained models

System Health

- API uptime and latency
- Memory and CPU usage of deployed services
- Storage utilization and cost monitoring

Why Monitor

- Maintain reliability and scalability of the platform under real-world usage
- Detect and resolve performance bottlenecks or pipeline failures early
- Keep the knowledge base (medical terms, lab codes) current to avoid outdated outputs
- Improve patient experience by iteratively optimizing clarity and response quality

Tools for Monitoring

- Google Cloud Monitoring & Alerts: Track uptime, latency, resource usage
- Logs Explorer / BigQuery: Analyze user interactions and error logs
- MLflow: Monitor model performance, drift, and version lifecycle
- Evidently AI: Detect data drift and output degradation
- Custom Dashboards (Grafana/Streamlit): Visualize user metrics, feedback, and bottlenecks

11. Success and Acceptance Criteria

The project will be considered successful and accepted when it meets the following:

- **Accuracy:** ≥80% of summaries are medically accurate and clearly understandable in pilot testing
- **Translation Quality:** ≥85% accuracy in English-to-Hindi medical term translations
- **Performance:** Average API response time <10s; system supports ≥100 concurrent users without failures
- **Reliability:** ≥95% uptime post-deployment with active monitoring and alerting
- **User Satisfaction:** ≥85% positive feedback (thumbs-up/ratings ≥4/5) on clarity and usefulness of explanations
- **Integration:** End-to-end workflow (upload → summarize → risk analysis → translation) works without critical pipeline errors
- **Compliance:** Data remains HIPAA-safe and fully de-identified; outputs include clear non-diagnostic disclaimers
- **Risk Identification:** ≥75% accuracy in identifying abnormal values and potential risk indicators
- **Language Support:** Fully functional bilingual support with seamless switching between English and Hindi
- **Processing Success Rate:** ≥90% of uploaded reports successfully processed without errors

12. Project Timeline (Tentative)

Phase	Duration	Start Date	End Date
Phase 1: Project Scoping & Setup	1 day	09/25/25	10/01/25
Phase 2 : Data Acquisition & Preprocessing Pipeline	3 Weeks	10/02/25	10/26/25
Phase 3: Model Fine-tuning & Training Pipeline	2 Weeks	10/27/25	11/08/25
Phase 4 : MLOps Pipeline & Experiment Tracking	2 Weeks	11/09/25	11/22/25
Phase 5: CI/CD & Deployment Automation	2 Weeks	11/23/25	12/06/25

Phase 6: Final Testing, Demo Prep & Expo Dry Run	4 Days	12/07/25	12/10/25
Phase 7: Monitoring & Post-Deployment Support	Ongoing	12/11/25	Ongoing

13. Additional Information

- **Hugging Face Medical Models Repository: https://huggingface.co/models?search=medical Access to pre-trained medical NLP models including BioBERT, ClinicalBERT, and medical translation models that can accelerate development**

- **Google Cloud Healthcare API Documentation: https://cloud.google.com/healthcare-api/docs Comprehensive guide for building HIPAA-compliant healthcare applications on GCP with best practices for medical data handling**

- **MIMIC-III Clinical Database Documentation: https://mimic.mit.edu/docs/iii/ Essential reference for understanding MIMIC-III table structures, data types, and clinical coding systems used in the dataset**

- **Medical Translation Resources - UMLS Metathesaurus: https://www.nlm.nih.gov/research/umls/index.html Unified Medical Language System providing standardized medical terminology across multiple languages, useful for building medical dictionaries for Hindi translation validation**