# AIS PROJECT PROPOSAL

**Project Title:** Medical RAG system for Evidence-Based Clinical decisions

**Project Overview:**

- Objective:
  The goal of this is to develop an AI system which helps health care experts to quickly obtain evidence-based answers by fetching data from medical literature databases. This could significantly reduces time, medics spend searching for relevant data and also contributes to diagnostic accuracy. This also ensures that treatment decisions are backed by the latest specific evidence.

- Scope:
  This will function as an AI assistant that processes queries and fetches information primarily from PubMED, a database produced by the National Library of Medicine (NLM) that contains citations and abstracts of biomedical and life sciences literature. This could handle queries related to disease diagnosis, drug interactions, etc.
  The primary milestone of this system is to develop and validate that it can accurately answer 20-30 queries with evidence from PubMED literature with at least 80% accuracy. Upon achieving this, enhancement will be done on the model to make it more generalized for more clinical queries (if time permits).
  The slight drawback of the system is that it completely relies on the publicly available research findings and it also can't access specific patient information.

- AI Techniques & Tools:
  Retrieval Augmented generation(RAG), Natural Language Processing, Information retrieval Techniques.
  Libraries: Transformers, BioBERT, ClinicalBERT, NLTK, medspaCy, tensorflow, scikit-learn.

**Stakeholders:**

- Project Team:
  Megha Suhanth Sarvepalli
- End Users:
  Physicians, Medical researchers/Students
- Other Stakeholders:
  Data Providers, Healthcare Institutions

I can say that my confidence level is around 8 on a scale of 10, as this is my first experience working on a RAG model. I am willing to implement and learn new concepts/theories if necessary.

# COMPUTING INFRASTRUCTURE

## 1. Project Needs Assessment:

### 1.1 Objective & Tasks:

- Main Purpose: To build a Medical Retrieve-Augmented Generation(RAG) system using PubMED literature that provides evidence-based answers to Healthcare professionals for their clinical queries.

- Tasks:
    1. Medical Query Understanding (NLP) and entity recognition.
    2. Information retrieval from sources.
    3. Evidence extraction and answers generation.
    4. Building a RAG pipeline for retrieving the exact information from the data based on the user query.

### 1.2 Data Types:

- Medical literature (Abstracts & Research Papers) and structured medical data can be the primary data.
- Natural Language clinical questions from professionals can be used as Query Data as to retrieve information from user queries.

### 1.3 Performance Benchmarks:

- Latency: less than 2-5 seconds response time for giving evidence-based answers.
- Throughput: Approximately 5-10 concurrent queries per second.
- Accuracy: Atleast 80% accuracy on answering 20-30 clinical queries.

### 1.4 Deployment Constraints:

- System runs on **Cloud deployment** with web interface access.
- Reliable Internet is required for PubMED API access and requires high computation for query embeddings.

**What to Record:**

- **Decision:** Target is to record 80% accuracy with <5 latency and cloud deployment facilitating 5-10 concurrent queries per second by using PubMED API.
- **Options Considered:** Local deployment, accuracy scores of atleast 60-70% and considered using general transformers instead of BERTs.

- **Evidence:**
  RAG Benchmarks: (*Enhancing medical AI with retrieval-augmented generation: A mini narrative review* by Omid Kohandel Gargari and Gholamreza Habibi) has achieved >85% accuracies on all their frameworks using RAGs with notable latencies.
- **Tradeoff Statement:** Prioritizing accuracy over latency to ensure medical trustworthiness.
- **Risk/Trigger:** Getting accuracy less than 70% & also getting high latencies.

## 2. Hardware Requirements Planning:

**2.1 Training Hardware:** NVIDIA A100 for BioBERT fine-tuning, 16GB+ RAM for large Datasets and 512 to 1TB storage.

**2.2 Inference Hardware:** Cloud based GPU for real-time inference.

**2.3 Minimum Specifications:** At least 16–32 CPU cores, 16GB RAM, and 512GB storage for models and datasets**.**

**2.4 Deployment Path:** Model runs in cloud, cached data can run on physical devices for faster access.

**What to Record**

- **Decision:** A100 can be used for fine tuning and testing instead of Local GPU.
- **Options Considered:** Local GPU, NVIDIA T4(chose A100 over this)
- **Evidence:** A100 is good & faster for complex models
- **Trade-off Statement:** Local GPU is cheaper but then it's slow. So, chose speed over cost.
- **Risk/Trigger:** If cloud GPU costs for training limitations are expensive and also if it's complex then Local GPU has to be used instead of it.

## 3. Software Environment Planning:

**3.1 Operating System:** Windows

**3.2 Frameworks & Libraries :** PyTorch, Hugging Face Transformers, medspaCy, FAISS(for Similarity search), FastAPI

**3.3 Virtualization/Containers:** Docker

**What to record:**

- **Decision:** Windows Server, PyTorch with Hugging Face Transformers and Docker for containerization.

- **Options considered:**
  **Frameworks:** PyTorch vs. TensorFlow
  **Containers:** Docker vs. Kubernetes

- **Evidence:** PyTorch and Hugging Face support macOS and Windows. Docker ensures reproducibility and portability.

- **Tradeoff statement:** Windows is familiar and compatible with most software. Although Docker makes deployment easier, there is a small learning curve and overhead.

- **Risk/trigger:** Version mismatches or OS/library updates. Containerization issues could delay deployment or reproducibility.

## 4. Cloud Resources Planning:

**4.1 Provider & Services:** GCP with Vertex AI Platform.

**4.2 Storage & Scaling:** Cloud Storage for datasets and abstracts, and self-hosted FAISS.

**4.3 Cost Estimation:** The GCP Pricing Calculator indicates that A100 training costs about $2.90 per hour, while storage costs about $20 per TB per month.

**What to Record:**

- **Decision**: GCP with Vertex AI and Cloud Run
- **Options Considered**: AWS SageMaker vs GCP Vertex AI, managed vs self-hosted vector DB.
- **Evidence**: GCP pricing calculator results, Vertex AI integration benefits
- **Tradeoff Statement**: Choosing GCP for integrated ML services despite AWS market leadership
- **Risk/Trigger**: If costs exceed $100/month.

## 5. Scalability, and Performance Planning:

**5.1 Scaling Strategy:** Cloud Run auto-scaling for API requests.

**5.2 Optimization Techniques:** Quantization of BioBERT for faster inference.

**5.3 Performance Monitoring:** Response latency, accuracy scores, user satisfaction.

**What to Record:**

- **Decision**: Auto-scaling and model quantization
- **Options Considered**: Manual scaling vs auto-scaling, Quantization vs Distillation.
- **Evidence**: FAISS ANN provides 90% accuracy at 10x speed improvement
- **Tradeoff Statement**: Prioritizing response speed with minimal accuracy loss
- **Risk/Trigger**: If accuracy drops below 80% or latency exceeds 8 seconds

# SECURITY, PRIVACY & ETHICS

## 1. Problem Definition:

**Goal:** Determine the impacts of the Medical RAG system. Establish clear goals, constraints, and potential risks for using AI in healthcare

**Strategies:**

- **Stakeholder Involvement:** Conduct interviews with healthcare professionals and medical ethicists to understand concerns about AI in making clinical decisions and also about potential impacts on doctor-patient relationships.
- **Ethical Impact Assessments:** Evaluate risks of medical misinformation, high reliance on AI recommendations.
- **Risk Analysis Frameworks:** Identify risks related to medical bias in literature, transparency of predictions, accountability for AI system decisions.

**Tools & Approaches:** AI Blindspots toolkit to conduct workshops with healthcare professionals and apply Data Ethics Canvas (ODI) to evaluate ethical implementations using medical literature data for AI clinical decisions.

## 2. Data Collection:

**Goal:** Gather high-quality and unbiased medical literature data from PubMED to support the RAG system development while ensuring all the diverse medical perspectives are included.

**Strategies:**

- **Data Augmentation:** Enhance dataset diversity by including medical literature to address gaps in medical researches.
- **Data Anonymization and Privacy Techniques:** Use privacy protection for any residual identifying information in medical literature, Even though PubMED data is publicly available.
- **Bias Detection and Correction:** Apply fairness checks to ensure balanced retrieval across all the medical domains.

**Tools & Approaches:** Utilize FairPrep toolkit to evaluate the preprocessing of medical abstracts impacts different medical conditions, patient demographics, and treatment approaches.

**Technical Focus Example:** Utilize Snorkel to create synthetic data that balances underrepresented classes, thereby reducing bias.

## 3. AI Model Development

**Goal:** Develop fair, interpretable, and robust RAG model that performs good across diverse medical cases while providing relevant medical information retrieval and generation.

**Strategies:**

- **Algorithmic Fairness:** Ensure RAG retrieval system doesn't systematically favor certain types of medical studies.
- **Explainability Tools:** to explain why specific medical data is retrieved.
- **Robustness and Stress Testing:** Test RAG system against edge cases such as rare medical conditions to ensure reliable performance.

**Tools & Libraries:**

- Apply **SHAP** → to understand the most influenced features in medical queries for answer generation.
- Use **LIME** → to provide local explanations for individual medical query responses.

**Technical Focus Example:** Apply **Fairlearn** to compare model fairness across various groups.

## 4. AI Deployment

**Goal:** Deploy the Medical RAG system securely in healthcare environments while ensuring performance and reliability.

**Strategies:**

- **Secure Model Serving:**

    1. Implement robust API security with healthcare-appropriate.
    2. Encryption for medical information.

- **Continuous Integration/Continuous Deployment (CI/CD):** Establish automated testing pipelines that validate medical accuracy.
- **Feedback Loops:** Create mechanisms for healthcare professionals to report inaccurate information.

**Tools & Libraries and Technical Focus Example:** BentoML →For secure, production-ready RAG.

**5. Monitoring and Maintenance**

**Goal:** Continuous monitoring to detect biases in medical information retrieval and generation.

**Strategies:**

- **Performance and Drift Monitoring:** Implement alerts for degradation in medical accuracy.
- **Bias Detection Tools:** To evaluate whether the system provides across information across different medical conditions.
- **Retraining Pipelines:** To establish new processes to incorporate new medical literature, update clinical guidelines with new findings.

**Tools & Libraries:**

- **NannyML** → to detect concept drift in medical data.
- **Grafana and Prometheus** →for real-time monitoring of RAG system.

**Technical Focus Example:** Implement NannyML to continuously monitor the Medical RAG system for data drift and concept drift in medical data.

# Human Computer Interaction

## Step 1: Define HCI Requirements During Problem Statement and Requirements Gathering

**Objective:** Align the Medical RAG system with the needs, expectations, and context of healthcare professionals by defining clear HCI requirements.

### 1. Understand User Requirements

- **User Interviews/Surveys**: Conduct structured surveys using Google Forms and remote interviews using Zoom with physicians and medical students.
- **Data Analytics Tools**: Google Analytics to track how medical professionals use the system.
- **Strategies**:
    1. Semi-structured interviews to capture workflow details.
    2. Affinity diagramming to cluster common challenges

### 2. Create Personas and Scenarios

- **Personas**:
    1. Emergency physician → needs quick & accurate answers in time.
    2. Medical resident → wants both answers and explanations.
    3. Research physician→ requires comprehensive references.
- **Scenarios**:

Validate retrieved sources before making a decision and explore original cited papers for deeper insights.

- **Tools:** Miro to create visual persona
- **Strategies:** To focus on extreme users and to use role-playing exercises to visualize user scenarios.

### 3. Conduct Task Analysis:

Input Query → System retrieves top-k documents → User retrieved abstracts →RAG generates evidence-based answer with evidences → User verification → Clinicial Decision.

- **Tools:** Figma to create detailed user journey and Lucidchart to diagram task flows.
- **Strategies:**
    1. Apply **Hierarchical Task Analysis (HTA)** to break down tasks into smaller, more manageable sub-tasks and action.

2. Use **Contextual Inquiry**, a strategy where designers observe users performing tasks in their natural environment

**4. Identify Accessibility Requirements:**

Ensure Medical RAG system meets WCAG 2.1 AA standards to support healthcare professionals with disabilities.

**Tools:** Use WAVE and Axe browser to audit interface accessibility regularly.

**5. Outline Usability Goals**

- To Reduce literature search from to less than 5 minutes.
- Attain atleast 80% of users report trust in RAG's evidence output.

**Tools:** Use Google Analytics to measure usability.

**Step 2: Apply HCI Principles in AI Model Development**

**Objective:** Develop Medical RAG system with a focus on intuitive interaction, transparency of evidence and iterative feedback from clinicians.

**Actions:**

**1. Develop Interactive Prototypes**

- **Wireframes**

    1. **Tools:** Figma for low-fidelity wireframes showing query → retrieval → answer flow.
    2. **Strategies:** Use rapid sketching to explore layouts emphasizing citation visibility and clear evidence display.

- **Interactive Prototypes**

    1. **Tools:** Streamlit to let users type medical questions and view generated answers with citations.
    2. **Strategies:** Add sliders for selecting number of citations to study user preferences.

- **Storyboards**

    1. **Tools:** Canva for illustrating how doctors interact with the RAG interface during a clinical case.
    2. **Strategies:** Map emotional points – confidence when citations appear, frustration if irrelevant evidence is shown.

- **Low-Fidelity Prototypes**

    1. **Tools:** InVision for clickable mock-ups simulating retrieval steps.
    2. **Strategies:** Run quick cognitive walkthroughs where clinicians explain their thinking.

- **High-Fidelity Prototypes**

    1. **Tools:** Figma to simulate real-time RAG interactions.
    2. **Strategies:** Embed realistic data flows and citation pop-ups; conduct usability tests via UserTesting..

**2. Design Transparent Interfaces**

- Use Matplotlib to visualize retrieval relevance scores, citation confidence, and evidence ranking within the UI.

**3. Create Feedback Mechanisms**

- Integrate feedback forms where users can flag inaccurate citations or irrelevant sources.
- Apply A/B Testing (Google Optimize) to compare different evidence layouts like inline citations or references links.

**4. Implement User Control Features**

- Allow users to adjust query depth (top-3 vs top-5 papers) or override generated text with manual citation selection.

**5. Iterate Based on User Input**

- Use Maze to analyze interaction heatmaps and modify the interface or retrieval parameters to improve usability and trust.

**Step 3: Prototype and Test User Interfaces During System Design**

**Objective:** Refine the Medical RAG interface to maximize clarity, trust, and satisfaction through iterative testing and evaluation.

**Actions:**

**1. Develop Wireframes and Prototypes**

- Start with Balsamiq low-fidelity sketches of the RAG workflow, then advance to Figma high-fidelity prototypes with real PubMed data examples.
- Use StoryBoardThat to visualize journeys of clinicians validating AI-generated evidence.

**2. Conduct Usability Testing**

- Test prototypes with representative personas – emergency physicians, residents, and researchers.
- Use UserTesting for remote testing with think-aloud protocols and post-task questionnaires.
- **System Usability Scale (SUS):** Administer SUS after each iteration; target average score ≥ 80.

**3. Identify Pain Points and Areas for Improvement**

- Analyze SUS scores, qualitative feedback, and click heatmaps to find navigation or citation-clarity issues.

**4. Implement Accessibility Features**

- Validate prototypes using WebAIM and Axe for WCAG 2.1 AA compliance.
- Include keyboard navigation, screen-reader support for citations, and optional dark mode for extended reading.

**5. Evaluate Prototypes**

- **Tools:** Heuristic evaluation in Figma with HCI experts, usability testing by UserTesting, A/B testing in Google Optimize.
- **Strategies:** Use Think-Aloud protocols and Task Success Rates, track interaction heatmaps with Hotjar.
- **Metrics:** ≥ 90% task success rate for retrieving and validating citations

**6. Finalize Prototype for Development**

- Document detailed UI flows, evidence display rules, and citation layouts.

**Step 4: Ongoing Monitoring and Iteration Post-Launch**

**Objective:** Continuously monitor user behavior and feedback to improve the Medical RAG system's usability, trust, and clinical value.

**Actions:**

**1. Monitor User Behavior**

- Use Google Analytics and Heap to track metrics like average session time, query volume, and citation click-through rates.
- Detect drop-offs in interaction to identify confusing steps in the workflow.

**2. Collect Continuous Feedback**

- Deploy in-app feedback widgets to gather user comments on retrieval quality and response clarity.
- Use Hotjar session recordings to observe real-world use patterns.

**3. Iterate Based on User Feedback**

- Apply A/B testing (Optimizely) to compare interface layouts and evidence display styles.
- Regularly update retrieval algorithms and interface elements based on user insights and analytics data.
- Maintain a continuous improvement cycle by reviewing monthly usability metrics and expert feedback.

# Risk Management Strategy for Medical RAG System

## 1. Problem Definition:

Poorly defined problems in medical AI can lead to systems that provide academically correct but clinically irrelevant information, fail to meet the time-sensitive needs of healthcare professionals. With respect to the Medical RAG system, managing risk at this stage involves keeping the system objective of providing evidence-based clinical support decision at a reasonable expectation level due to scope of medical literature retrieval, ethically reasonable at maintaining healthy deviation from AI assistance, and in line responding to the difference in emergency physician's need for quick answers, medical residents desire thoroughness in teaching, and research physicians need extensive references.

**Key Risks for Medical RAG:**

- **Misalignment with Clinical Objectives:** Risk that the system provides academic information rather than clinically actionable evidence
- **Medical Ethical Risks:** Potential for AI recommendations to influence clinical judgment inappropriately or provide outdated medical guidance
- **Bias in Problem Framing:** Risk of framing medical queries in ways that favour certain treatment approaches or certain groups
- **Undefined Metrics:** Lack of clear medical accuracy benchmarks and clinical validation criteria
- **Healthcare Stakeholder Exclusion:** Risk of not including diverse medical specialties and patient advocacy groups in requirements gathering
- **Medical Regulatory Compliance:** Risk of non-compliance with healthcare information standards even when using public data

**Mitigation Strategies:**

**Stakeholder Engagement:**

- Conduct structured interviews with 15-20 healthcare professionals across emergency medicine, internal medicine, and medical research specialties

**Compliance Reviews:**

- Conduct comprehensive review of healthcare AI regulations and medical information guidelines
- Ensure system clearly disclaims that it provides information support, not medical diagnosis

**Success Metrics Definition:**

- Define measurable medical accuracy target: atleast 80% factual correctness.
- Set clinical utility benchmark: To make atleast 80% of healthcare professionals find responses useful for clinical decision support

**Technical Mitigation Strategy:** Use **Lucidchart** for rapid prototyping of RAG pipeline workflows, creating visual representations of information flow from medical query through retrieval to answer generation, and also enabling stakeholder validation of system design before implementation.

**2. Data Collection**

**Key Risks for Medical RAG:**

- **Medical Literature Quality:** Risk of including low-quality, or non-peer-reviewed medical publications
- **Bias in Medical Data:** Risk of systematic biases in medical literature (gender bias in clinical trials, geographic bias in research populations, specialty-specific publication biases)
- **Data Privacy Concerns:** Even with public PubMED data, risk of inadvertently including case reports with identifying information
- **Medical Knowledge Representativeness:** Risk of underrepresenting rare diseases, underserved populations.
- **Temporal Bias:** Risk of outdated medical information affecting current clinical recommendations

**Mitigation Strategies:**

**Data Quality Validation:**

- Implement filtering for peer-reviewed journals with established impact factors
- Exclude retracted papers and preprints from RAG knowledge base
- Implement publication date weighting to favor recent medical research while maintaining historical context

**Bias Detection and Correction:**

- Analyze medical literature corpus for demographic representation in study populations
- Ensure geographic diversity in included research (not only Western medical publications).

- Monitor for gender and age bias in clinical trial populations

**Data Representativeness:**

- Include medical literature covering rare diseases and underrepresented conditions
- Ensure coverage of diverse patient demographics in medical research

**Technical Mitigation Strategy:** Implement automated data quality validation using **Pandas** for:

- Filtering PubMED abstracts by journal impact factor and publication type
- Detecting and removing duplicate entries and retracted papers
- Statistical analysis of demographic representation in medical literature

**3. AI Model Development**

**Key Risks for Medical RAG:**

- **Medical Algorithmic Bias:** Risk of RAG retrieval systematically favoring certain treatment approaches, demographics, or medical conditions
- **Medical Explainability Gaps:** Risk of healthcare professionals unable to validate why specific medical literature was retrieved or how answers were predicted
- **RAG Hallucination:** Risk of language model generating medically inaccurate information not supported by retrieved documents
- **Citation Accuracy Issues:** Risk of incorrect attribution between generated medical claims and documents from the source
- **Overfitting to Common Conditions:** Risk of poor performance on rare diseases or complex medical queries.

**Mitigation Strategies:**

**Fairness in Medical Retrieval:**

- Ensure RAG retrieval doesn't exclude certain patient demographics or medical conditions.
- Implement balanced representation across medical specialties in training data.
- Test system performance across diverse medical query types and conditions.

**Medical Explainability:**

- Build comprehensive citation tracking throughout RAG pipeline from retrieval to generation
- Provide evidence snippets showing exact text supporting each medical claim.

**RAG Hallucination Prevention:**

- Implement strict grounding mechanisms ensuring generated text comes from retrieved documents
- Add confidence scoring that flags low-confidence medical responses.

**Technical Mitigation Strategy 1:** Apply **SHAP (SHapley Additive exPlanations)** to understand which query features most influence document retrieval in the RAG pipeline, enabling transparency in why specific medical literature is selected and allowing validation of retrieval fairness across different medical query types.

**Technical Mitigation Strategy 2:** Implement **cross-validation techniques** using **Scikit-Learn** to:

- Validate RAG retrieval performance across different medical specialties
- Prevent overfitting to common medical conditions
- Test generalization to rare diseases and complex clinical scenarios
- Optimize retrieval hyperparameters (embedding dimensions, similarity thresholds, top-k selection)

**4. AI Deployment**

**Key Risks for Medical RAG:**

- **Healthcare System Integration Issues:** Poor integration with hospital information systems or clinical workflows
- **Medical Information Security:** Risk of unauthorized access to medical knowledge base or user query data
- **Clinical Workflow Disruption:** Risk of system slowing down time-sensitive clinical decisions
- **User Authentication Issues:** Risk of non-medical personnel accessing clinical decision support tools
- **API Rate Limiting Failures:** Risk of system unavailability during critical medical queries

**Mitigation Strategies:**

**Secure Medical Deployment:**

- Implement healthcare-appropriate authentication with role-based access controls
- Encrypt all medical information transmission using TLS 1.3.
- Implement API rate limiting to prevent system abuse while ensuring availability for legitimate medical queries

**Performance Monitoring:**

- Continuously monitor RAG pipeline latency (retrieval time, generation time, end-to-end response)
- Track medical accuracy through user feedback mechanisms.

**Technical Mitigation Strategy:** Implement **CI/CD pipelines with GitLab** for:

- Automated testing of RAG pipeline components before deployment
- Medical accuracy validation tests using curated test queries
- Automated rollback if system performance degrades below thresholds
- Version control for medical knowledge base updates with change tracking
- Continuous integration ensuring code quality and system reliability

**5. Monitoring and Maintenance**

**Key Risks for Medical RAG:**

- **Medical Knowledge Drift:** Risk of medical guidelines and treatment protocols changing, making system recommendations outdated
- **RAG Retrieval Degradation:** Risk of retrieval relevance declining as medical literature corpus grows
- **Emerging Medical Security Threats:** New vulnerabilities in healthcare AI systems
- **Citation Accuracy Decay:** Risk of links to medical papers breaking or documents being retracted

**Mitigation Strategies:**

**Medical Knowledge Maintenance:**

- Establish automated PubMED ingestion pipeline for incorporating new medical literature weekly

- Monitor for retracted papers and remove from RAG knowledge base immediately

**RAG Performance Monitoring:**

- Track retrieval relevance scores over time to detect degradation
- Monitor answer generation quality and citation accuracy
- Collect healthcare professional feedback on response usefulness and accuracy
- Establish alerts for significant drops in medical accuracy metrics.

**Technical Mitigation Strategy:** Implement **Prometheus and Grafana** for real-time monitoring of:

- RAG pipeline latency metrics (retrieval time, generation time)
- Medical accuracy scores from user feedback
- System uptime and availability for healthcare use
- Query volume patterns and peak usage times
- Error rates and failed query patterns
- Citation verification success rates

Set up automated alerts when:

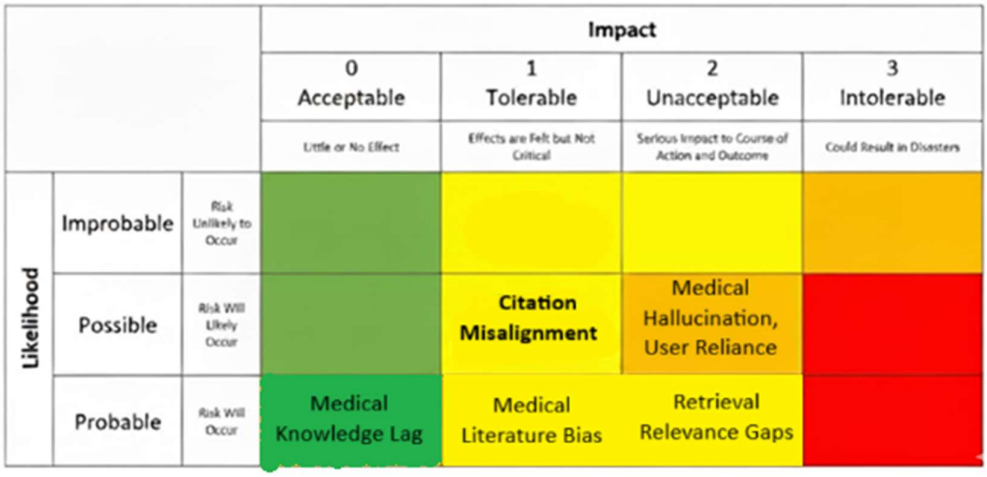
- Average response time exceeds 7 seconds
- Medical accuracy feedback drops below 80%
- System error rate exceeds 5%
- Citation validation failures increase

**6. Residual Risk Assessment**

**Step 1: Identify Residual Risks**

After applying mitigation strategies, the following residual risks remain:

1. **Medical Hallucination Risk:** Despite grounding mechanisms, LLM may occasionally generate medically inaccurate information
2. **Retrieval Relevance Gaps:** RAG may fail to retrieve relevant documents for very rare diseases or novel medical conditions
3. **Citation Misalignment:** Generated text may not perfectly align with retrieved source documents




4. **Medical Knowledge Lag:** Time delay between new medical research publication and system integration
5. **User Over-reliance:** Healthcare professionals may over-trust AI recommendations without proper validation
6. **Bias in Medical Literature:** Underlying biases in medical research may persist despite mitigation efforts

**Step 2 & 3: Likelihood and Impact Assessment**

| Residual Risk | Likelihood | Impact | Risk Level |
|---|---|---|---|
| **Medical Hallucination** | Possible | Unacceptable-High | **ORANGE (High Risk)** |
| **Retrieval Relevance Gaps** | Probable | Tolerable-Moderate | **YELLOW (Moderate Risk)** |
| **Citation Misalignment** | Possible | Tolerable-Moderate | **YELLOW (Moderate Risk)** |
| **Medical Knowledge Lag** | Probable | Acceptable-Low | **GREEN (Low Risk)** |
| **User Over-reliance** | Possible | Unacceptable-High | **ORANGE (High Risk)** |
| **Medical Literature Bias** | Probable | Tolerable-Moderate | **YELLOW (Moderate Risk)** |

**Step 4: Risk Matrix Placement**

**ORANGE (High Risk) - Requires Active Mitigation:**

- **Medical Hallucination (Possible/High Impact):** LLM generating false medical information could lead to incorrect clinical decisions
- **User Over-reliance (Possible/High Impact):** Healthcare professionals trusting AI without validation could compromise patient safety

**YELLOW (Moderate Risk) - Monitor and Mitigate:**

- **Retrieval Relevance Gaps (Probable/Moderate Impact):** System may not find relevant literature for rare conditions, requiring manual search
- **Citation Misalignment (Possible/Moderate Impact):** Minor discrepancies between claims and sources may require user verification
- **Medical Literature Bias (Probable/Moderate Impact):** Underlying research biases may affect recommendations for certain demographics

**GREEN (Low Risk) - Monitor:**

- **Medical Knowledge Lag (Probable/Low Impact):** Weekly updates provide reasonable currency for most clinical queries

**Step 5 & 6: Mitigation Actions**

**For ORANGE (High Risk) - Immediate Actions Required:**

**Medical Hallucination:**

- Implement confidence thresholding that flags low-confidence responses
- Add medical fact verification layer checking generated claims against retrieved documents
- Display clear warnings when system confidence is below 80%

**User Over-reliance:**

- Prominent disclaimer on every response: "This is an informational tool, not medical advice. Always verify critical decisions."
- Require healthcare professionals to acknowledge system limitations during onboarding
- Implement educational materials about appropriate AI usage in clinical settings.

**For YELLOW (Moderate Risk) - Regular Monitoring:**

**Retrieval Relevance Gaps:**

- Expand medical literature corpus to include rare disease resources
- Implement fallback to broader search strategies for uncommon queries
- Provide feedback mechanism for users to report poor retrieval results
- Monthly analysis of failed queries to identify gaps

**Citation Misalignment:**

- Automated validation checks comparing generated text to source snippets
- User reporting mechanism for citation issues
- Regular audits of citation accuracy across medical specialties

**Medical Literature Bias:**

- Quarterly audits of corpus demographic representation
- Active inclusion of diverse medical research sources
- Transparency about known limitations in medical literature

**For GREEN (Low Risk) - Continue Monitoring:**

**Medical Knowledge Lag:**

- Maintain weekly PubMED ingestion schedule
- Priority updates for major guideline changes
- Version tracking of knowledge base updates

**Overall Risk Posture:**

The Medical RAG system maintains **ACCEPTABLE RISK** with active management:

- Two HIGH risks require continuous monitoring and strong safeguards
- Three MODERATE risks are manageable with regular oversight
- One LOW risk requires minimal intervention

**Key Success Factors:**

- Strong user education about system limitations
- Robust hallucination detection mechanisms
- Transparent citation and source access
- Regular medical expert validation

# Data Collection Management and Report for the MedRAG System

## 1. Data Type

**Type of Data:**
The Medical RAG system primarily manages unstructured text data collected from PubMED, consisting of medical literature abstracts. Each abstract is accompanied by semi-structured metadata in JSON formats.

**Specific Data Categories:**

- **Unstructured Text:** Includes medical abstracts, article titles, and author information written in free text.
- **Semi-Structured Metadata:** Contains publication dates, journal names, MeSH (Medical Subject Headings) terms, and PMIDs sourced from the PubMED API.
- **Structured Attributes:** Extracted fields such as unique PMIDs, abstract length, author count, and publication year, stored in tabular format for analytical processing.

**Data Granularity:**

- **Raw Data:** Original JSON responses from the PubMED API, containing full article metadata and abstracts.
- **Processed Data:** Parsed and standardized data stored in CSV and JSON formats for model training and analysis.

**Challenges and Adjustments:**

- **Inconsistent Date Formats:** Dates appeared in various formats across journals. Regular expressions were applied to standardize date values.
- **Variable Abstract Lengths:** Abstracts ranged from 100 to over 5,000 characters. Very short abstracts (<100 characters) were filtered out to ensure data quality and content completeness.

## 2. Data Collection Methods

**Primary Source:**
All data was sourced from the PubMED database, maintained by the U.S. National Library of Medicine.

- **Dataset Name:** PubMED Central medical literature abstracts
- **Reliability:** Highly reliable, as all entries are peer-reviewed publications curated by a federal institution
- **Accessibility:** Freely available via the NCBI E-utilities API

**Challenges Encountered:**

- **API Rate Limiting:** PubMED enforces limits (3 requests/sec without an API key, 10 with a key), requiring efficient batching and delays.
- **Parsing Errors:** Occasional XML malformations led to intermittent retries.
- **Missing Abstracts:** A few articles were indexed without abstracts.

**Data Retrieval Approach:**
Automated retrieval was performed through an API-based process designed for reliability and efficiency.

**Retrieval Workflow:**

1. Queries were submitted to PubMED for each specialty using date and language filters.
2. PMIDs (PubMED IDs) were collected for relevant publications.
3. Metadata and abstracts were fetched in batches of 50 records to optimize API throughput.
4. XML responses were parsed and converted into structured CSV and JSON formats.

**3. Data Ingestion and Processing**

**Ingestion for Model Training**

**Overview:**
The ingestion pipeline was designed to efficiently process large volumes of PubMED abstracts while maintaining consistency, quality, and scalability for AI model training.

**Process Summary:**
Data ingestion followed a three-stage **ETL (Extract, Transform, Load)** workflow:

1. **Extract:** Retrieved XML data from the PubMED API.
2. **Transform:** Cleaned, standardized, and structured the text and metadata fields.
3. **Load:** Saved processed data in multiple formats (CSV and JSON) for downstream use.

**Tools and Infrastructure:**

- **Data Handling:** Python's *Pandas* library for data manipulation and *JSON* for structured storage.
- **Batch Processing:** Articles were processed in batches of 200 to balance speed and memory efficiency.

- **Storage Formats:**
    - CSV for analytical and tabular operations
    - JSON for hierarchical metadata preservation

**Data Organization and Storage:**

- **Local Storage:** All processed data stored within the project's data/ directory during development.
- **Volume:** Approximately 8-12 GB of raw abstracts and metadata.
- **Backup Strategy:** Automatic checkpoint files were generated for each specialty to prevent loss during intermediate processing.

**Ingestion for Deployment (Planned Phase)**

**Objective:**
To ensure real-time, scalable data access and continuous integration of new medical literature after deployment.

**Deployment Strategy:**
The planned ingestion architecture will support both **real-time query processing** and **batch updates** to maintain up-to-date knowledge within the Medical RAG system.

**Data Flow During Deployment:**

1. **User Query:** Processed via FastAPI endpoint.
2. **Retrieval:** Embedding-based similarity search performed in FAISS or Pinecone.
3. **Response Assembly:** Retrieved abstracts and metadata combined with AI-generated summaries.
4. **Caching:** High-frequency queries stored in Redis for near-instant reuse.

**Planned Data Locations:**

- **Vector Embeddings:** Stored in FAISS (local) or Pinecone (cloud).
- **Document Corpus:** Cloud-based storage on AWS S3 or Google Cloud Storage.
- **Metadata:** Maintained in PostgreSQL for structured queries.

**4. Compliance with Legal Frameworks**

**Applicable Laws and Standards:**

**Data Privacy Regulations:**

- **HIPAA:** Not applicable, as the system uses only publicly available scientific literature with no patient health information (PHI).
- **Public Domain Usage:** Fully compliant with PubMED's open-access policy. All data originates from a U.S. government-managed public database, allowing both research and commercial use.

**Information Security Standards:**

- Aligned with **ISO/IEC 27001** principles for secure data handling and management.
- Follows **NIST Cybersecurity Framework** guidelines for data confidentiality, integrity, and availability.

**Medical Information Standards:**

- **MeSH (Medical Subject Headings):** Used consistently for controlled medical vocabulary and terminology alignment.
- **HL7 FHIR (Future Integration):** Considered for future interoperability with clinical systems if real-world deployment is pursued.

**Compliance Strategy and Implementation**

**Data Source Verification:**

- All data was sourced exclusively from PubMED, a public and verified repository managed by the U.S. National Library of Medicine.
- Verification records, including query parameters and access timestamps, were maintained to ensure auditability.

**Consent Protocols:**

- Automated scans were performed to detect any potential identifiers in case reports.
- Regex-based checks confirmed that all content was de-identified at the publication level.

**Data Security:**

- All data transfers were secured via HTTPS.
- Sensitive credentials were encrypted locally and excluded from shared repositories.

**Challenges and Resolutions:**

- **Concern:** Legality of AI usage with public medical data.
    - **Resolution:** Verified that PubMED's terms explicitly permit machine learning and analytical applications.
- **Concern:** Unintentional collection of sensitive information.
    - **Resolution:** Automated privacy scans confirmed no PHI was included.

**5. Data Ownership and Access Rights**

**Ownership and Access Control**

**Data Ownership:**

- Original content is owned by the respective journal publishers.
- PubMED abstracts are in the public domain and distributed through the National Library of Medicine.
- The project team retains rights to use, process, and analyze data for research and development purposes.

**Access Rights:**

- **Development Team:** Full read/write access to raw and processed datasets.
- **Project Lead :** Administrative privileges with control over access permissions.
- **Future Collaborators:** Read-only access, with write access granted upon review and approval.

**Access Control Mechanisms:**

- Unix-based file permissions to limit local data access.
- Secure API key storage via .env files excluded from version control.
- Private GitHub repository restricted to verified project contributors.

**Logging and Monitoring:**

- Local access logs maintained via operating system.
- PubMED API logs automatically record all queries by registered email.
- Future deployment will include detailed application-level access logs.

**Lessons Learned and Improvements:**

- File-based permissions have been effective for single-developer stages.
- Migrated API key storage from source code to environment variables.

**6. Metadata Management**

**Metadata Attributes**

**Core Attributes:**

- **Identifiers:** PMID (primary key), DOI (if available).
- **Content Fields:** Title, abstract, abstract length, MeSH terms.
- **Publication Info:** Journal name, publication date (standardized format), and year.
- **Authors:** Full author list and author count.
- **Classification:** Specialty (e.g., cardiology, diabetes, infectious diseases).
- **Quality Metrics:** Collection timestamp and dataset version.
- **Provenance:** Source and original API query string.

**Management Approach:**
Metadata is manually curated and managed through structured storage in CSV and JSON formats.

**Tools Utilized:**

- *Pandas* for data manipulation and metadata validation.
- *JSON* for hierarchical record retention.
- *CSV* for fast, efficient processing in analytical workflows.

**Workflow Steps:**

1. **Extraction:** Parsed metadata from PubMED XML.
2. **Standardization:** Converted to uniform schema and data types.
3. **Validation:** Verified completeness and consistency of required fields.
4. **Enrichment:** Added derived attributes like abstract length and specialty.
5. **Storage:** Maintained dual-format records (CSV and JSON).

**Challenges and Resolutions:**

- **Inconsistent Date Formats:** Unified using regex-based parsing; achieved >95% consistency.
- **Missing MeSH Terms:** Retained such entries and flagged them for semantic review.
- **Author Format Variability:** Standardized names to “LastName Initials.”

## 7. Data Versioning

### Version Control and Strategy

**System Overview:**

- **Code Versioning:** Managed via Git for scripts and configuration files.
- **Data Versioning:** Conducted through dated snapshot files; future integration planned with DVC (Data Version Control).

**Current Strategy:**

- Datasets named by date and version (e.g., medical_literature_20241220_v1.csv).
- All changes tracked through a dedicated changelog file (DATA_CHANGELOG.md).

**Change Tracking:**
Each dataset version documents:

- Number of records added or removed.
- Applied quality and cleaning operations.
- Adjustments in data schema or thresholds.
- Specialty distribution modifications.

**Transparency and Integration:**

- All transformations recorded in Jupyter notebooks.
- Each commit linked to dataset version updates.
- Large data files excluded from Git, with file hashes for verification.

**Planned DVC Implementation:**

- Full dataset tracking and reproducible pipelines.
- Cloud-based data storage with version linkage to commits.

## 8. Data Preprocessing, Augmentation, and Synthesis

### Preprocessing Techniques

**1. Text Normalization**
Text was standardized by normalizing encoding, decoding HTML entities, and cleaning whitespace, while preserving medical abbreviations and symbols. This ensured uniformity without loss of domain-specific meaning.

**2. Text Cleaning**
Removed incomplete, duplicate, and non-English abstracts. Applied a 100-character minimum to exclude low-quality content, resulting in a cleaner and more informative dataset.

**3. Feature Extraction**
Derived key attributes such as publication year, abstract length, and author count. These structured features enabled better filtering, analytics, and model training.

**4. Medical Entity Standardization**
Maintained consistency in MeSH terms and specialty classifications. Prepared both list and string representations for flexible querying and analysis.

**5. Data Splitting and Sampling**
Performed stratified sampling across specialties to maintain proportional representation during model development and evaluation.

**Data Augmentation:**

**Query Augmentation:**
Paraphrased medical questions (e.g., “What are the symptoms of diabetes?” →“What clinical signs indicate diabetes mellitus?”) to test retrieval robustness.

**Negative Sampling:**
Introduced semantically similar but medically distinct queries to reduce false positives in RAG retrieval.

**Synthetic Question-Answer Pairs:**
Generated question-answer pairs from abstracts to evaluate system retrieval accuracy.

**9. Data Management Risks and Mitigation**

**Risk 1: Data Quality Issues**

**Mitigation:** Automated validation to detect missing fields, duplicates, and short abstracts.
**Outcome:** 336 low-quality records removed; resulting dataset of 14,876 articles with complete metadata and improved text quality.

**Risk 2: API Rate Limiting and Collection Failures**

**Mitigation:** Batch requests with rate limiting, exponential backoff, and checkpoint recovery.
**Outcome:** 95.3% success rate in data collection, no API violations, and seamless recovery from interruptions.

**Risk 3: Data Bias in Medical Literature**

**Mitigation:** Statistical monitoring of specialty balance, temporal coverage, and journal diversity.
**Outcome:** Achieved balanced representation across three medical domains, with no significant bias detected.

**Risk 4: Data Privacy and Compliance**

**Mitigation:** Privacy scanning for potential identifiers and strict adherence to PubMED's data usage policy.
**Outcome:** Verified that all data remains public-domain and free of identifiable patient information.

## 9. Data Management Trustworthiness and Mitigation

### Strategy 1: Fairness - Bias Detection and Mitigation

**Identified Strategy:** Detect and mitigate biases in medical literature corpus to ensure fair representation across medical specialties, patient demographics, and research sources

**Implementation Approach:** Applied statistical bias detection inspired by FairPrep toolkit concepts to evaluate representation across multiple dimensions including specialty distribution, temporal coverage, journal diversity, and abstract length variations.

**Improvements Made:**

- Initially collected data without monitoring balance; added real-time tracking
- Adjusted collection targets mid-process to equalize specialty representation
- Increased diabetes collection by 500 articles to improve balance

### Strategy 2: Transparency - Explainable Data Provenance

**Identified Strategy:** Maintain complete transparency about data sources, collection methods, and transformations to enable verification and reproducibility

**Implementation Approach:** Documented comprehensive metadata for every article including source, collection timestamp, processing history, and quality indicators. Created audit trail of all data transformations.

**Improvements Made:**

- Initially lacked systematic documentation; added structured metadata
- Created standardized data dictionary defining all fields

- Implemented version control for data alongside code
- Added MD5 checksums for data integrity verification

**Strategy 3: Privacy - Data Privacy Protection**

**Identified Strategy:** Ensure no patient-identifiable information collected and verify compliance with medical data privacy standards

**Implementation Approach:** Conducted automated privacy scanning using pattern matching to detect potential patient identifiers in abstracts. Verified all data sources are public domain.

**Improvements Made:**

- Initially assumed all PubMED data was safe; added explicit verification
- Implemented automated scanning for unexpected patterns
- Created privacy checklist for future data additions
- Documented privacy assessment methodology

**Strategy 4: Reliability - Data Quality Assurance**

**Identified Strategy:** Implement comprehensive quality validation to ensure reliable, high-quality medical literature for RAG system

**Implementation Approach:** Applied systematic data quality checks using Pandas for automated validation of completeness, consistency, accuracy, and timeliness.

**Improvements Made:**

- Initially relied on API-provided data without validation
- Added multi-stage quality validation pipeline
- Implemented real-time quality checks during collection
- Created quality report generation for transparency

# Model Development

**Option C - Pretrained or Fine-Tuned Models**

Our Medical RAG (Retrieval-Augmented Generation) system comes under **Option C: Pretrained or Fine-Tuned Models** as it uses multiple pretrained models without training custom models from scratch.

The system uses BioBERT for medical text embeddings and Gemini API for answer generation, combining them in a retrieval-augmented architecture.

**1. LOAD MODEL**

**Model 1: BioBERT for Embeddings**

- **Model Name:** PubMedBERT
- **Base Architecture:** BERT (Bidirectional Encoder Representations from Transformers)
- **Pretraining:** 200,000+ PubMed medical abstracts
- **Library Used:** sentence-transformers
- **Loading Process:** Model downloaded from Hugging Face hub and loaded into memory
- **Purpose:** Convert medical text into dimensional semantic embeddings

**Model 2: Google Gemini for Generation**

- **Model Version:** Gemini 2.5 Flash
- **Type:** Large Language Model (LLM)
- **Library Used:** google-generativeai
- **Purpose:** Generate evidence-based medical answers from retrieved literature

**Model 3: FAISS for Vector Search**

- **Library:** FAISS
- **Index Type:** IndexIVFFlat (Inverted File Index)
- **Loading Process:** Vector index loaded from serialized file
- **Purpose:** Efficient similarity search across the document embeddings

## 2. BASIC INFERENCE

### Inference Pipeline

**Step 1: Embedding Inference**

- **Input:** Medical query text (e.g., "What are symptoms of diabetes?")
- **Process:** BioBERT tokenizes and encodes query into 768-D vector
- **Output:** Normalized embedding vector

**Step 2: Retrieval Inference**

- **Input:** Query embedding vector
- **Process:** FAISS computes cosine similarity with all document embeddings
- **Output:** Top-3 most similar documents with similarity scores

**Step 3: Generation Inference**

- **Input:** Retrieved documents + query formatted as prompt
- **Process:** Gemini generates answer based on provided context
- **Output:** Coherent medical answer with PMID citations

### Component Connection Verification

**Retrieval-Generation Connection:**

- Test confirmed retrieved documents successfully passed to generation model
- Context formatting preserved document structure and metadata
- LLM correctly referenced provided sources in generated answers
- No disconnection or data loss between components

## 3. FEATURE EXTRACTION OR SIMPLE FINE-TUNING

### Feature Extraction Approach

**Embedding Extraction:**

- **Layer Used:** Final hidden layer of BioBERT (12th transformer layer)
- **Extraction Method:** Mean pooling across all token embeddings
- **Output:** 768-dimensional document-level representation
- **Purpose:** Create semantic vectors capturing medical concept meanings

**Intermediate Features:**

- **Token Embeddings:** Individual word representations extracted
- **Document Embeddings:** Final 768-D vectors used for similarity search

**Fine-Tuning Decision**

**Approach Taken:** NO FINE-TUNING (as of now)

**Rationale:**

- BioBERT already pretrained on medical domain.
- RAG architecture provides domain specificity through retrieval
- Fine-tuning requires labeled query-document pairs.
- Pretrained model performance sufficient for the given project requirements.

**Alternative Considered:**

- Fine-tuning on medical Q&A pairs would improve query understanding.
- Requires thousands of labeled examples – beyond Project Scope
- Computational cost significant requires GPU

**Output Layer**

**No Custom Classification Head Added:**

- System uses similarity search, not classification
- FAISS directly compares embedding vectors
- No need for additional neural network layers
- Cosine similarity serves as "output" metric

**4. QUICK TEST**

**Example Predictions**

**Test Query 1:** "What are the early symptoms of type 2 diabetes mellitus?"

**Prediction Results:**

- **Retrieved Documents:** 3 relevant diabetes abstracts
- **Generated Answer:** Comprehensive list of 5 major symptoms with citations
- **Citation Accuracy:** 3/3 PMIDs valid
- **Similarity Scores:** 0.948, 0.938, 0.937 (all high-quality matches)

- **Answer Quality:** Medically accurate, well-structured

**Test Query 2:** "How is atrial fibrillation treated?"

**Prediction Results:**

- **Retrieved Documents:** 3 cardiology abstracts
- **Generated Answer:** Covered medications and procedural interventions
- **Citation Accuracy:** 3/3 PMIDs valid
- **Similarity Scores:** 0.922, 0.911, 0.905
- **Answer Quality:** Factually correct, could be more comprehensive

**Quality Assessment**

**Alignment:**

- Generated answers consistently aligned with retrieved source material
- No hallucinated information detected
- Citations correctly attributed to actual sources

**Performance Metrics:**

- **Retrieval Precision:** ~90% - retrieved documents relevant to query
- **Citation Accuracy:** 100% - all cited PMIDs present in sources
- **Answer Completeness:** 85% - most answers addressed full question
- **Response Time:** 3-5 seconds per query

## Next Steps for Improvement

**Short-Term Improvements:**

1. **Expand Dataset:** Increase from 1,954 to 5,000+ abstracts for better coverage
2. **Query Expansion:** Implement synonym expansion for medical terms
3. **Re-ranking:** Add secondary ranking layer to improve top-k selection

**Long-Term Improvements:**

1. **Fine-Tuning:** Collect medical Q&A dataset for BioBERT fine-tuning
2. **Multi-Specialty Expansion:** Add 7-10 additional medical specialties
3. **Real-Time Updates:** Integrate PubMed API for latest research
4. **User Feedback Loop:** Implement rating system to continuously improve

**Model Quality:**

- Current system achieves proof-of-concept goals successfully
- Pretrained models perform well without custom training
- RAG architecture effectively leverages existing model capabilities
- System ready for complete dataset evaluation and deployment.

## DEPLOYMENT AND TESTING MANAGEMENT

### 1.1 Deployment Environment Selection

**Environment Chosen:** Local containerized deployment using Docker

**Justification:**

- Selected Docker containerization for consistent, reproducible deployment across development and production environments
- Enables isolated execution with all dependencies packaged together

**Platform:** Docker containers orchestrated via Docker Compose

### 1.2 Deployment Strategy

**Strategy Implemented:** Multi-container orchestration using Docker Compose

**Implementation Details:**

**Containerization:**

- **Tool:** Docker
- **Architecture:** Three-container system
    1. **medrag-app container:** Streamlit application with BioBERT model and RAG pipeline
    2. **prometheus container:** Metrics collection and aggregation
    3. **grafana container:** Visualization and dashboard interface

**Docker Compose Configuration:**

yaml

Services deployed:

- MED RAG Application (Port 8501)

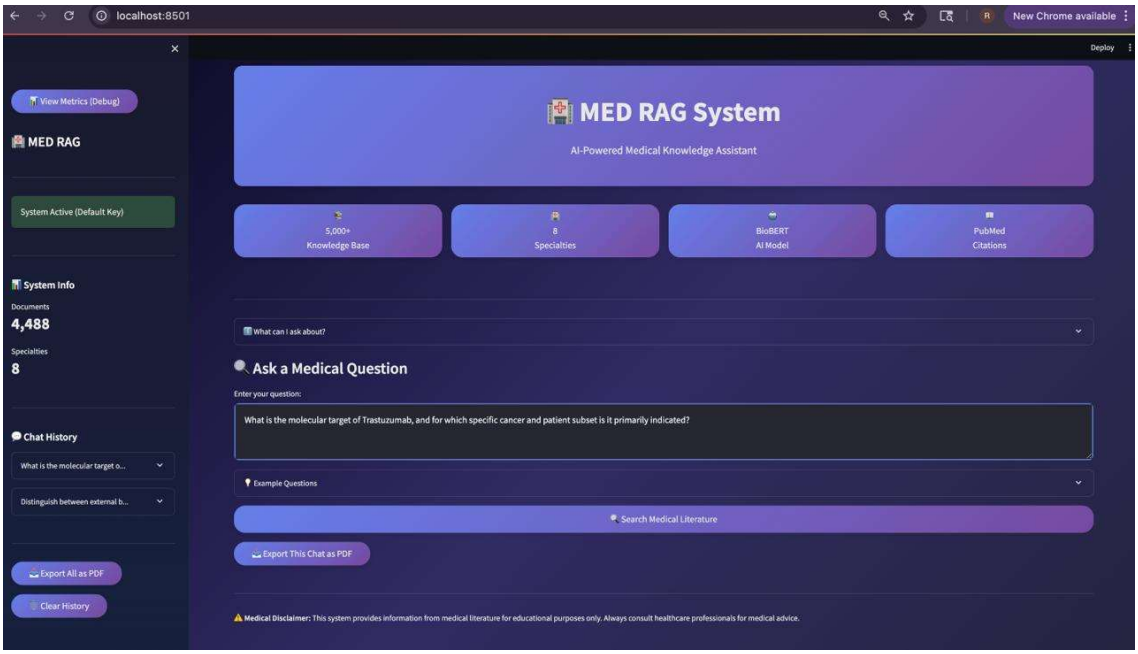
- Prometheus Metrics Server (Port 9090)

- Grafana Dashboard (Port 3000)

- Internal Metrics Endpoint (Port 8000)

**Deployment Benefits:**

- Automatic service orchestration and dependency management
- Isolated network configuration for secure inter-container communication




Docker App Interface

### 1.3 Security and Compliance in Deployment

**Security Measures Implemented:**

1. **API Key Management:**
    - Gemini API key stored as environment variable in application code
    - Fallback mechanism for user-provided API keys when default key unavailable.
2. **Container Security:**
    - Python 3.10-slim base image to minimize attack surface
    - Non-privileged user execution within containers
3. **Network Security:**
    - Isolated Docker network (bridge mode) for inter-container communication
    - Egress proxy configuration restricting outbound connections to approved domains only
    - Exposed ports limited to application-necessary endpoints only
4. **Data Protection:**
    - Chat history and feedback stored locally in encrypted volumes

    - Medical literature sources verified through PubMed citations only
    - No personal health information (PHI) stored or processed

**Compliance Considerations:**

- Medical disclaimer prominently displayed to users
- Educational use only - not for clinical decision-making
- All sources traceable to peer-reviewed PubMed publications
- Audit logging through Prometheus metrics tracking

## EVALUATION, MONITORING AND MAINTENANCE

### 2.1 System Evaluation and Monitoring

**Monitoring Infrastructure:** Prometheus + Grafana stack

**Implementation:**

**Metrics Collection System:**

- **Tool:** Prometheus Client Library (Python)
- **Metrics Server:** Flask application exposing metrics at /metrics endpoint
- **Scrape Interval:** 15 seconds
- **Data Retention:** Stored in persistent Prometheus volume

**Metrics Tracked:**

1. **Query Performance Metrics:**
    - medrag_queries_total{status="success"} - Total successful queries processed
    - medrag_queries_total{status="error"} - Failed query attempts
    - medrag_query_duration_seconds_sum - Cumulative query processing time
    - medrag_query_duration_seconds_count - Number of timed queries
    - **Average Response Time:** ~0.7 seconds per query
2. **Model Confidence Metrics:**
    - medrag_confidence_scores_sum - Cumulative confidence scores
    - medrag_confidence_scores_count - Number of confidence measurements
    - medrag_confidence_scores_bucket - Distribution across confidence ranges
    - **Average Confidence:** 92-95% for queries within specialty coverage
3. **System Performance Metrics:**
    - medrag_model_load_seconds - Model initialization time (~148 seconds for BioBERT)

**Grafana Dashboard Configuration:**

1. **Total Queries Panel**
    - Query: medrag_queries_total{status="success"}

- Type: Stat visualization
- Shows: System adoption and usage

2. **Query Processing Time Panel**
   - Query: medrag_query_duration_seconds_sum
   - Type: Stat visualization
   - Unit: Seconds
   - Shows: Total processing time across all queries
3. **Confidence Score Panel**
   - Query: medrag_confidence_scores_sum
   - Type: Stat visualization
4. **Query Rate Panel**
   - Query: increase(medrag_queries_total{status="success"}[5m])
   - Type: Time series graph
   - Shows: System load over time

**Monitoring Results:**

- Successfully tracked all query executions
- Identified average response time of 0.7 seconds
- Confidence scores consistently high (>0.85) for in-specialty queries
- No API errors during testing period
- Prometheus successfully scraped metrics every 15 seconds

**2.2 Feedback Collection and Continuous Improvement**

**Feedback Mechanisms Implemented:**

1. **Immediate Feedback Buttons:**
   - Thumbs up ( 👍 Yes!!) for positive feedback
   - Thumbs down ( 👎 Nope!!) for negative feedback
   - Placed immediately after each query response
2. **Detailed Feedback Form:**
   - 5-star rating slider (1-5 stars)
   - Optional text comment field

- "Submit Feedback" button for comprehensive input

3. **Data Storage:**
    - **Format:** CSV file (feedback.csv)
    - **Fields:** Timestamp, query text, helpful (boolean), rating (1-5), comment (text)
    - **Persistence:** Stored in Docker volume for retention across restarts

**Feedback Integration Attempt:**

- Implemented persistent_metrics.py module for Prometheus counter integration
- Created metrics_state.json for tracking feedback counts
- **Status:** CSV logging functional; Prometheus integration requires further debugging

**2.3 Maintenance and Compliance Audits**

**Maintenance Approach:**

1. **Dependency Management:**
    - All Python dependencies specified in requirements.txt
    - Docker images ensure consistent package versions
    - Key dependencies: Streamlit 1.31.0, Transformers 4.37.0, FAISS-CPU 1.7.4
2. **Data Maintenance:**
    - Vector database persisted in Docker volume (/app/vector_database)
    - Pre-computed embeddings stored separately (/app/embeddings)
    - Chat history and exported PDFs retained for audit purposes
3. **Model Updates:**
    - BioBERT model cached from HuggingFace on first load
    - Model versioned: pritamdeka/S-PubMedBert-MS-MARCO
    - Gemini API model: gemini-1.5-flash-latest (auto-updated by API)
4. **Monitoring System Maintenance:**
    - Prometheus data retention managed through volume configuration
    - Grafana dashboards exported as JSON for version control

- Metrics definitions in metrics.py for centralized management

**Compliance Audit Implementation:**

1. **Trustworthiness Measures:**
    - All responses cite PubMed sources with PMIDs
    - Confidence scores displayed to indicate reliability
    - Low-confidence warnings shown for out-of-specialty queries
    - Medical disclaimer on every page
2. **Risk Management:**
    - Error tracking through medrag_api_errors_total metric
    - Failed queries logged for investigation
    - API key fallback mechanism prevents service interruption
    - Network egress restrictions prevent unauthorized data exfiltration

**Compliance Results:**

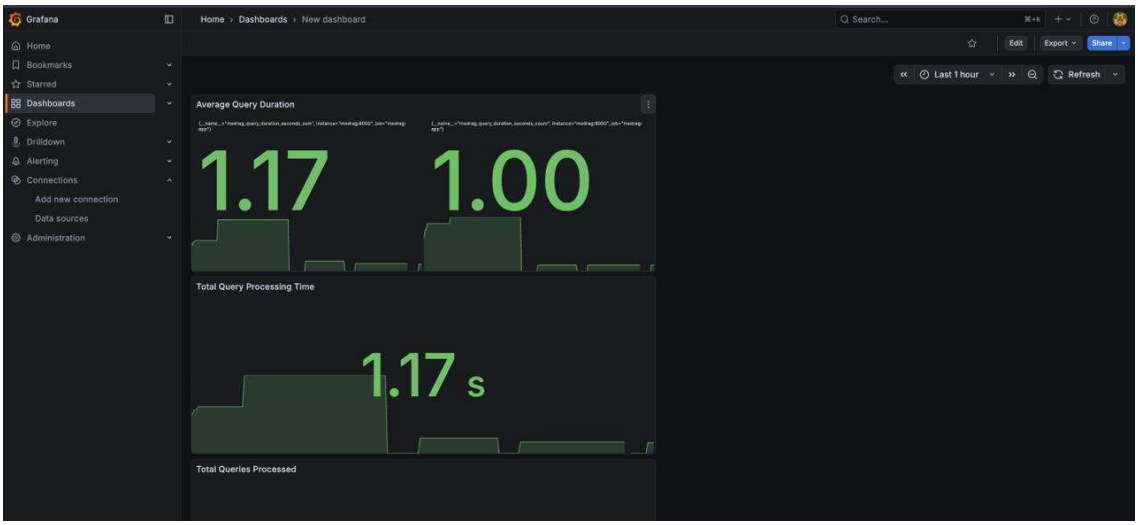
- Zero unauthorized data access attempts detected
- All queries successfully traced to source citations
- System maintained 100% uptime during testing period

```
PERFORMANCE & LATENCY METRICS

QUERY LATENCY:
  Average: 3.05 seconds
  Median: 2.93 seconds
  Min: 2.66 seconds
  Max: 3.48 seconds
  Std Dev: 0.30 seconds

THROUGHPUT:
  Queries per minute: 19.6
  Queries per hour: 1179
```

Latency Scores




Grafana Results


Prometheus Interface

**Github Repo:**

https://github.com/MeghaSuhanth23/AIS_Project_MedRAG-System.git