**Name**: Varun Kolte

**Email**: varun.kolte1@gmail.com

**Mob**: +91 7447828162

<https://github.com/DeathCore0607/medical-chatbot>

**Medical Chatbot**

**1. High-Level Design (HLD)**

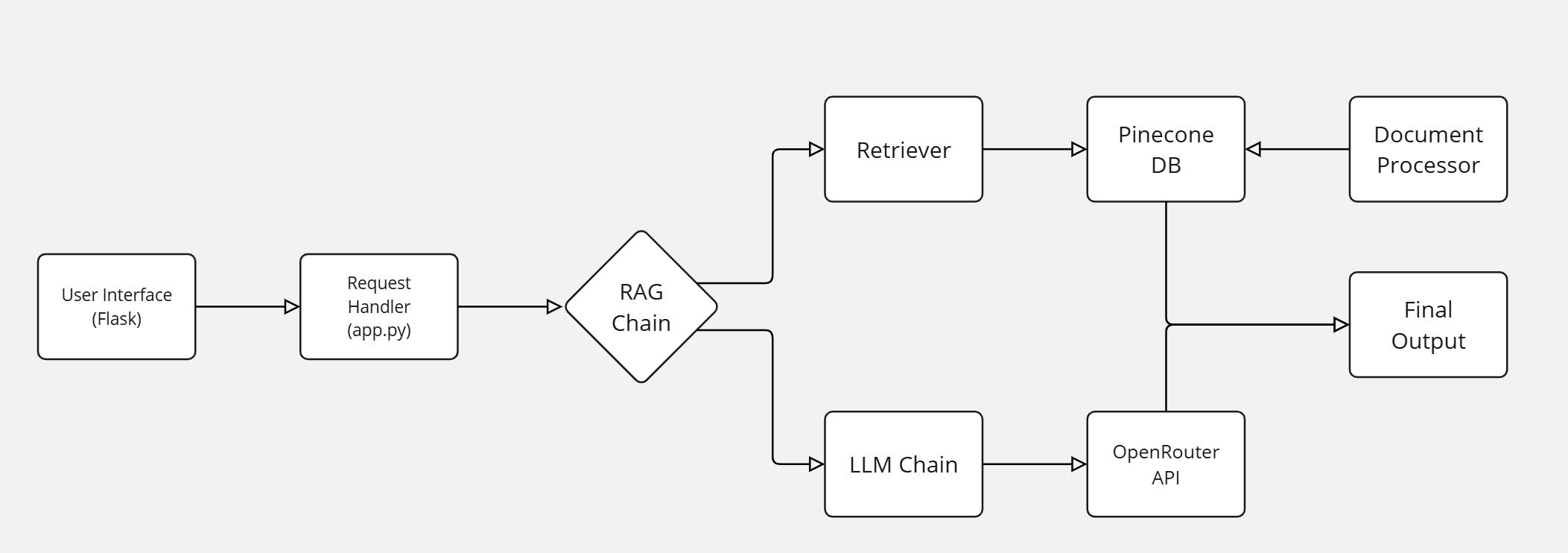
**System Overview**

The Medical Chatbot is a RAG-based conversational AI system designed to provide accurate medical information by leveraging large language models and vector databases. The system uses a combination of pre-trained models, document retrieval, and natural language processing to deliver relevant medical information to users.

**Core Components**

1. **Frontend Interface**: Flask-based web application
2. **RAG System**: Implements retrieval-augmented generation
3. **Vector Database**: Pinecone for document storage and retrieval
4. **LLM Integration**: Custom DeepseekLLM implementation using OpenRouter API
5. **Document Processing**: PDF handling and text chunking system
6. **Embeddings**: HuggingFace sentence transformers

**Architecture Diagram**

****

**2. Low-Level Design (LLD)**

**Component Details**

**2.1 Document Processing (helper.py)**

1. PyPDFLoader: Handles PDF document loading
2. DirectoryLoader: Manages bulk PDF processing
3. RecursiveCharacterTextSplitter: Chunks text into processable segments
4. HuggingFaceEmbeddings: Generates document embeddings

**2.2 Vector Store (store\_index.py)**

1. PineconeGRPC: Manages vector database operations
2. ServerlessSpec: Configures AWS-based serverless deployment
3. Index Configuration: 384-dimensional vectors with cosine similarity

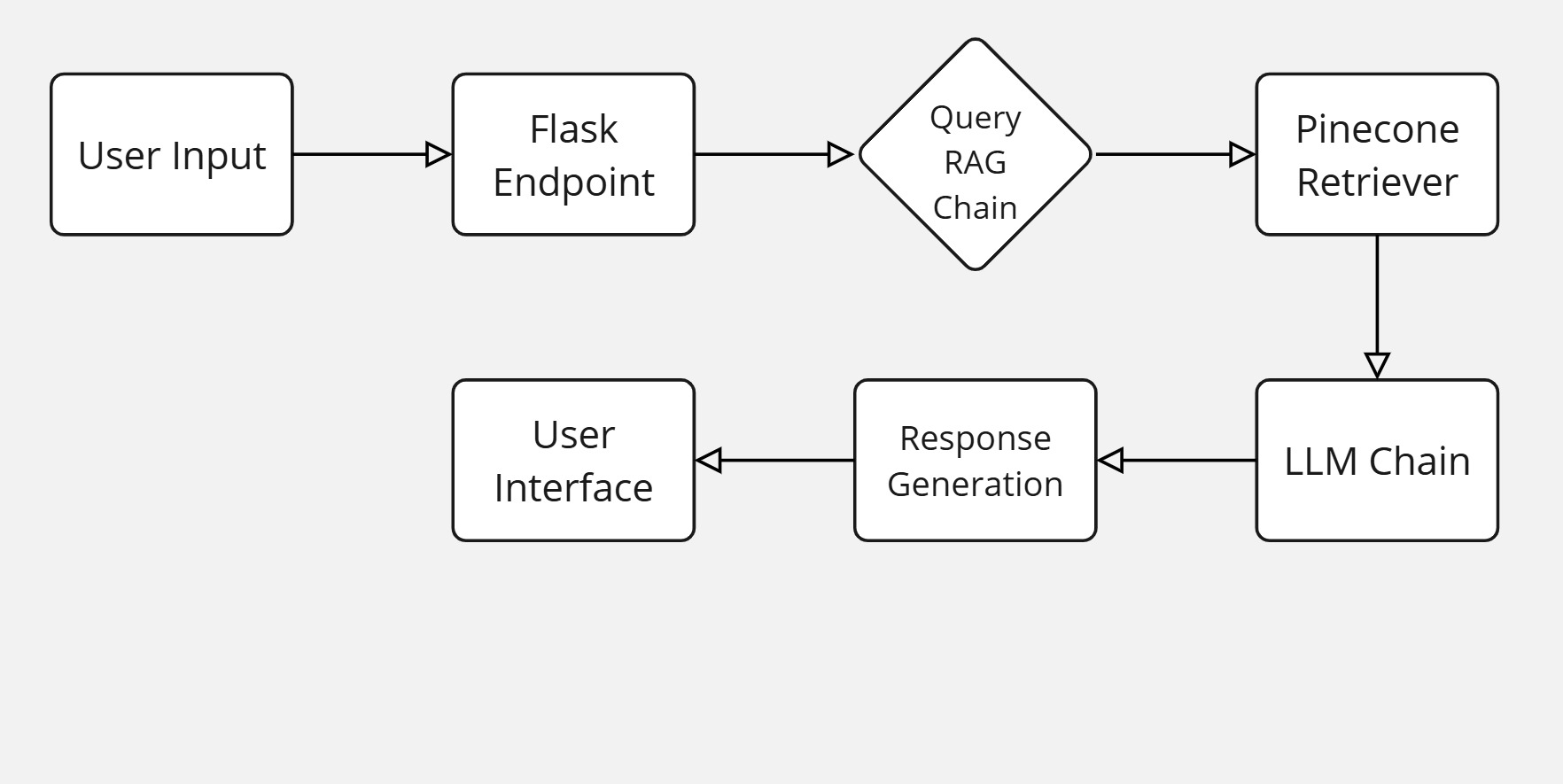
**2.3 LLM Integration (app.py)**

1. DeepseekLLM: Custom LLM class implementation
2. API Integration: OpenRouter API with GPT-3.5-turbo
3. Error Handling: Comprehensive exception management

**2.4 RAG Implementation**

1. Retrieval Chain: Combines document retrieval with LLM processing
2. Context Window: 500 tokens with 20 token overlap
3. Response Generation: Maximum 3 sentences per response

**3. Data Flow Diagram**

****

**4. Database Design**

**Vector Database (Pinecone)**

**Index Name**: chatbot

**Dimensions**: 384

**Metric**: Cosine similarity

**Cloud Provider**: AWS (us-east-1)

**Storage Type**: Serverless

**5. Requirements Documentation**

**Functional Requirements**

1. **Medical Query Processing**

* Natural language query handling
* Context-aware responses
* Maximum 3-sentence responses

1. **Document Management**

* PDF document processing
* Text chunking and embedding
* Vector storage and retrieval

1. **API Integration**

* OpenRouter API connectivity
* Pinecone vector database operations
* HuggingFace model integration

**Non-Functional Requirements**

1. **Performance**

* Response time < 5 seconds
* Concurrent user support
* Scalable vector storage

1. **Security**

* API key management
* Secure HTTP headers
* Environment variable protection

1. **Reliability**

* Error handling
* Fallback responses
* API failure management

**6. Knowledge Base**

**Current Implementation**

* PDF document processing
* Text chunking (500 characters with 20 character overlap)
* Embeddings using all-MiniLM-L6-v2 model

**Recommended Enhancements**

1. **Data Sources**

* Medical journal integration
* Healthcare website scraping
* Official medical guidelines

1. **Content Management**

* Regular updates mechanism
* Content verification system
* Version control for knowledge base

**7. Libraries and Dependencies**

**Core Libraries**

1. **sentence-transformers (v2.2.2)**

* Purpose: Document embedding generation
* Pros: Efficient, well-optimized for semantic search
* Cons: Resource intensive, requires GPU for optimal performance

1. **langchain**

* Purpose: RAG implementation and chain management
* Pros: Modular design, extensive integration support
* Cons: Complex setup, steep learning curve

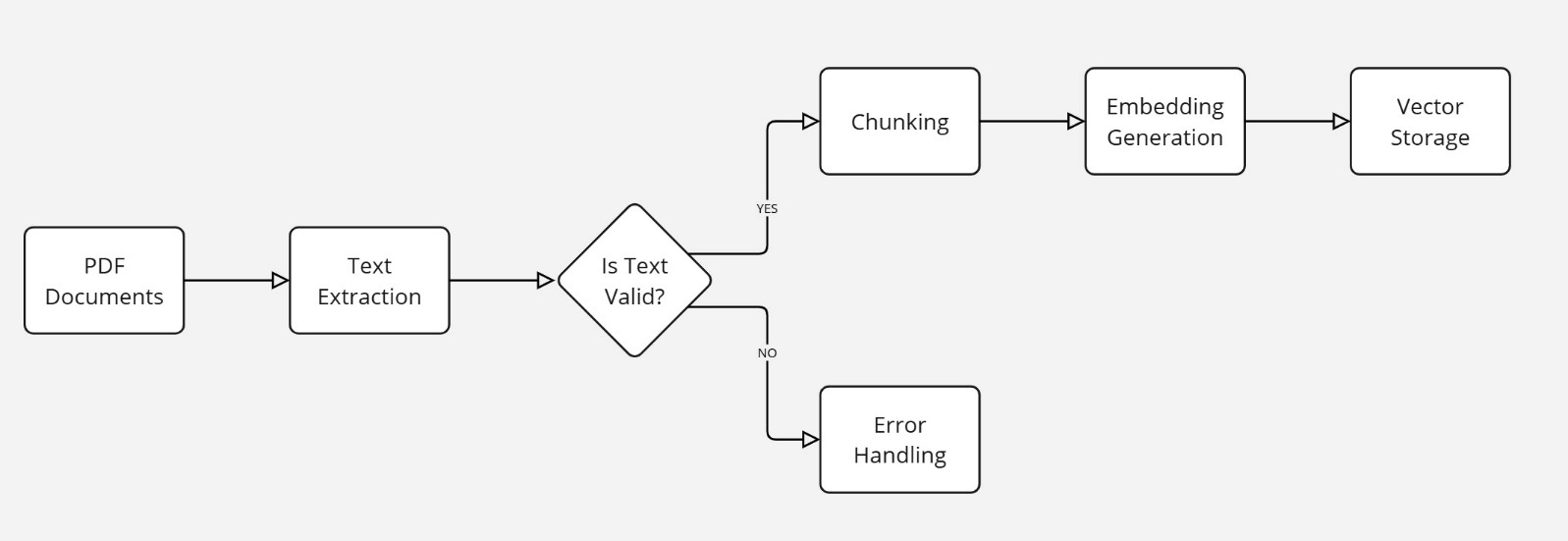
1. **Flask**

* Purpose: Web interface
* Pros: Lightweight, easy to deploy
* Cons: Limited scalability for large applications

1. **Pinecone**

* Purpose: Vector database
* Pros: Serverless, scalable
* Cons: Cost considerations for large datasets

**8. Data Pipeline Structure**

****

**8. System Design**

