iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH
IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 Issue: X Month of publication: October 2025

DOI: https://doi.org/10.22214/ijraset.2025.74460

Interpreting Medical Records with Generative AI: A Patient-Facing Chat Application Using Google Gemini

Syed Abrar Hassan[1], Syed Amaanullah[2], Mohammed Ameer Khan[3]
[1]*Department of Artificial Intelligence & Machine Learning, Lords Institute of Engineering and Technology*
[2]*Department of Artificial Intelligence & Machine Learning, Lords Institute of Engineering and Technology*
[3]*Department of Electronics and Communication Engineering, Sree Chaitanya Institute of Technological Sciences*

Abstract: In the age of digital health, patients are increasingly gaining access to their personal medical data, including lab test results, vital signs, and diagnostic reports. However, the technical and clinical complexity of this information often renders it difficult for non-medical users to interpret, potentially leading to confusion, misinformation, or anxiety. This research presents the development of an AI-powered web application that assists users in understanding their health data through natural language interaction. The system utilizes Google's Gemini generative AI model to process textual medical information and provide context-aware answers to user queries. The application, developed using the Streamlit framework, supports both direct text input and file uploads in PDF or TXT format. Extracted data is sent to the Gemini API using a structured prompt design that frames the model as a medical data assistant. The model's output is constrained with instructions to prioritize accuracy, avoid conjecture, and issue disclaimers discouraging self-diagnosis. Core components of the system include file parsing using PyPDF2, secure API key management via dotenv, asynchronous API interaction, and a chat-like user interface that maintains dialogue context using Streamlit's session state. This project demonstrates a practical and ethical implementation of generative AI in consumer-facing health technology. By enhancing the accessibility and interpretability of medical records, the application empowers users to become more informed participants in their healthcare journey, while reinforcing the importance of professional consultation for clinical decisions.
Keywords: Generative AI, Medical Records Interpretation, Chatbot, Google Gemini, Healthcare Information Accessibility

I. INTRODUCTION

A. Background and Motivation

The evolution of electronic health records (EHRs) and online patient portals has transformed the way individuals engage with their healthcare information. With a few clicks, patients can now access lab results, diagnostic summaries, prescription histories, and clinical notes. While this transparency marks significant progress in patient empowerment, it also presents a growing challenge: the complexity and clinical terminology found in most medical data can be overwhelming to the average person. Misinterpretation of results or clinical terms can lead to anxiety, incorrect self-diagnosis, or neglect of important health issues.

The scale of this challenge is substantial. According to recent studies, approximately 88% of adults in the United States lack the health literacy skills needed to effectively navigate the healthcare system and manage their health. Even well-educated individuals often struggle with medical terminology, reference ranges, and the contextual significance of health data.

Medical reports frequently contain specialized terminology, acronyms, and numerical data that require expert interpretation. Consider a typical comprehensive metabolic panel, which may include over 14 different measurements—each with its own reference range and clinical significance. For patients without medical backgrounds, distinguishing between clinically significant abnormalities and minor variations can be nearly impossible. This information asymmetry creates barriers to effective patient engagement and shared decision-making in healthcare.

B. The Role of Generative AI in Healthcare Analytics

Generative AI represents a paradigm shift in healthcare data interpretation. Unlike rule-based systems of the past, modern large language models (LLMs) like Google's Gemini can understand context, process natural language queries, and generate human-like responses.

This capability transforms how medical data can be accessed and understood in several key ways:

1) *Natural Language Understanding:* LLMs allow users to query medical data using everyday language rather than requiring specialized technical knowledge or formal query structures. This democratizes access to health information by meeting users at their level of understanding.
2) *Contextual Processing:* These models can consider both the explicit data provided and their broader training on medical knowledge, potentially filling gaps and providing more comprehensive interpretations than simple lookup systems.
3) *Adaptive Communication:* LLMs can adjust their responses based on the perceived knowledge level of the user, explaining complex concepts when necessary or providing more detailed technical information when appropriate.
4) *Multi-format Analysis:* Advanced models can process various document formats and structures, adapting to the heterogeneous nature of medical documentation.
5) *Personalized Explanations:* Unlike static educational materials, AI systems can tailor explanations to specific user queries, addressing their particular concerns about their individual health data.

The potential benefits of AI-powered medical analytics extend to multiple stakeholders in the healthcare ecosystem:

a) *For Patients*: Improved access to understandable explanations of their health data may lead to better health literacy, more informed decision-making, and increased engagement in their care.
b) *For Clinicians*: These systems could reduce the time spent explaining routine results, allowing more focused discussions during patient consultations.
c) *For Healthcare Systems:* More informed patients may make better use of healthcare resources, potentially reducing unnecessary follow-up visits driven by confusion or anxiety.

However, the application of generative AI in medical contexts also presents significant challenges related to accuracy, reliability, and appropriate scope. Models may generate plausible-sounding but incorrect information (hallucinations), lack access to the latest medical research, or fail to consider important patient-specific factors not present in the provided data.

C. Research Objectives

This research explores the feasibility and effectiveness of using generative AI to assist non-medical users in understanding personal health data. Specifically, we aim to address the following objectives:

1) Design and implement a conversational interface for interpreting medical data that balances accessibility with appropriate limitations and disclaimers.
2) Evaluate the capability of Google's Gemini model to process and interpret common medical document formats, including lab reports, clinical notes, and diagnostic summaries.
3) Develop effective prompt engineering strategies that guide the AI to provide informative but appropriately cautious interpretations of health data.
4) Assess the technical challenges and ethical considerations involved in deploying AI systems for medical data interpretation.
5) Identify potential pathways for future development, including opportunities for integration with existing healthcare information systems and areas requiring further research.

Through addressing these objectives, we aim to contribute to the growing body of knowledge around responsible AI applications in healthcare information management. Our focus is not on developing diagnostic systems or clinical decision support tools, but rather on exploring how AI might enhance information accessibility while maintaining appropriate boundaries in the healthcare domain.

II. LITERATURE REVIEW

Recent advancements in generative artificial intelligence (AI) and large language models (LLMs) have significantly transformed medical assistant chatbots. These technologies are being increasingly used to enhance diagnostic accuracy, patient interaction, data management, and overall accessibility in healthcare settings. Below is a detailed review of ten significant studies exploring the integration of LLMs into medical chatbots.

1) *Med-Bot:* An AI-Powered Assistant to Provide Accurate and Reliable Medical Information Bhatt and Vaghela (2024) proposed "Med-Bot," an advanced AI-powered chatbot developed using PyTorch, Langchain, and AutoGPTQ frameworks. The chatbot provides reliable medical information and is tailored for a conversational interface that ensures high interactivity and natural dialogue. The system's strength lies in its modular design and use of multiple open-source AI components, making it an accessible tool for healthcare service providers (Bhatt & Vaghela, 2024).

2) *ChatDoctor:* A Medical Chat Model Fine-Tuned on LLaMA Using Medical Domain Knowledge Li et al. (2023) introduced "ChatDoctor," a medical chatbot based on the LLaMA model. This system was fine-tuned using 100,000 anonymized dialogues between patients and doctors, enhancing its contextual understanding and domain specificity. The model demonstrated a high level of precision in answering medically relevant questions, making it a promising solution for semi-automated patient consultations (Li et al., 2023).
3) *Clinical Camel:* An Open Expert-Level Medical Language Model with Dialogue-Based Knowledge Encoding Toma et al. (2023) developed "Clinical Camel," an expert-level LLM tailored for medical tasks. Using QLoRA fine-tuning on LLaMA-2, the authors achieved top-tier performance on several clinical NLP benchmarks. The study underscores the viability of open-source LLMs in clinical contexts and emphasizes the importance of structured domain-specific training (Toma et al., 2023).
4) *HuatuoGPT:* Towards Taming Language Model to Be a Doctor Zhang et al. (2023) presented "HuatuoGPT," a large language model explicitly trained for healthcare consultations. It combined distilled outputs from ChatGPT with real-world doctor dialogue data. Reinforcement learning from AI feedback (RLAIF) was used to fine-tune the chatbot's responses, resulting in a system capable of producing context-aware, medically sound advice (Zhang et al., 2023).
5) *CareBot:* A Pioneering Full-Process Open-Source Medical Language Model Zhao et al. (2024) introduced "CareBot," a bilingual medical chatbot model capable of understanding and responding in multiple languages. The training pipeline included continuous pre-training on medical corpora, supervised fine-tuning, and reinforcement learning with human feedback. The result is a holistic AI medical assistant suitable for global deployment in multilingual environments (Zhao et al., 2024).
6) *eHealth Assistant AI Chatbot:* Using a Large Language Model Researchers Pap and Oniga (2024) presented the development of an artificial intelligence health assistant designed to enhance an existing eHealth data acquisition system. This AI chatbot allows patients and medical staff to interact through a secure, natural language chat interface built on the Matrix decentralized open protocol. The assistant uses a locally implemented, interchangeable large language model (LLM) to generate personalized and contextually relevant responses while ensuring patient privacy. The model accesses patient-specific data in a controlled manner to support accurate, individualized advice. The use of the Matrix protocol facilitates scalable and federated deployment, allowing widespread, privacy-preserving use across various healthcare settings. This approach showcases the potential of AI-driven automation in delivering real-time health guidance, remote monitoring, and virtual consultations.
7) *Leveraging LLM:* Implementing an Advanced AI Chatbot for Healthcare This paper IJISRT. (2024) explores the application of Large Language Models (LLMs) in healthcare, with a particular focus on addressing general illness inquiries through chatbot interfaces. By leveraging the advanced capabilities of LLMs, the study investigates their potential to deliver accurate and contextually relevant responses to users seeking information about common health concerns. A key strength of LLMs lies in their ability to continuously learn and improve through user interaction, making them increasingly effective over time. Benchmarking experiments conducted as part of this research indicate an accuracy rate of 61% in understanding and responding to user queries related to general illnesses. These results are analyzed against established benchmarks to assess the efficacy of LLM-driven chatbots in real-world healthcare scenarios. By examining the intersection of artificial intelligence and healthcare, this work contributes to the ongoing development of intelligent chatbot systems capable of offering reliable and informative support to individuals seeking guidance on general health issues.
8) *A Medical Chatbot*: Using LLaMA 2 Presented in the IRJMETS journal (2024) This paper explores the design and deployment of a Medical ChatBot powered by the enhanced Llama 2 model, aimed at improving healthcare accessibility. It efficiently processes user queries to deliver reliable medical information and emotional support using open-source AI technologies. The chatbot demonstrates strong performance in addressing user needs, showcasing its potential as a user-friendly tool for instant medical guidance (IRJMETS, 2024).
9) *Large Language Models:* for Chatbot Health Advice Studies Researchers from JAMA Network Open (2024) provided a comprehensive analysis of generative AI-powered chatbots in offering tailored healthcare advice. The study found that LLMs outperform traditional rule-based systems in terms of user satisfaction and personalization. However, it also noted risks related to misinformation and emphasized the need for clinician oversight (JAMA Network Open, 2024).
10) *Generative AI and Large Language Models:* in Health Care A review from the National Institutes of Health (2024) This paper discusses the growing role of generative AI and large language models in healthcare, particularly those trained on electronic medical records. It highlights their potential to simulate human-like conversation and support clinical decision-making. Building on recent reviews, the authors propose an evaluation checklist to guide the effective and responsible implementation of these models in medical settings.

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue X Oct 2025- Available at www.ijraset.com

These ten studies demonstrate the rapid evolution of AI-powered chatbots in healthcare, fueled by advancements in LLMs. From enhancing patient engagement to supporting diagnostic processes, the integration of AI is proving transformative. However, consistent themes across the literature also point to the need for careful ethical consideration, robust training datasets, and continual human oversight to ensure safe deployment.

III. METHODOLOGY

A. *System Architecture*

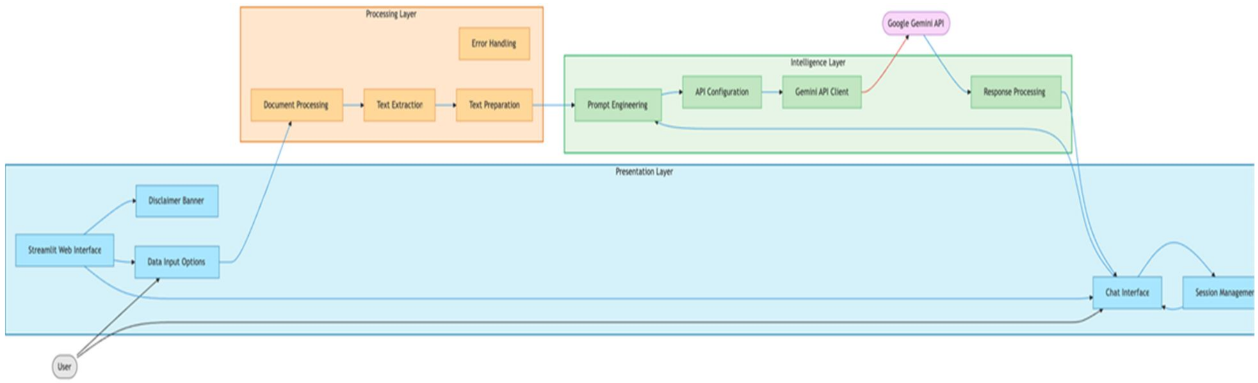


Fig 1. System architecture design

The system architecture for our AI-powered medical data analyzer follows a modular, three-tier design that separates concerns between data handling, AI processing, and user interaction. This approach prioritizes flexibility, maintainability, and potential future expansion while ensuring appropriate isolation between components.

B. *Architectural Overview*

The system consists of three primary layers:

1) *Presentation Layer:* A Streamlit-based web interface that handles user interactions, file uploads, text input, and conversational display. This layer is responsible for rendering the user interface, managing session state, and providing appropriate disclaimers and guidance.
2) *Processing Layer:* A data extraction and preparation module that processes uploaded files or pasted text, transforming them into a format suitable for AI analysis. This layer handles different document types, extracts text content, and maintains the conversational context.
3) *Intelligence Layer:* The AI integration component that connects with Google's Gemini API, formulates appropriately structured prompts, and processes responses for display. This layer encapsulates all interactions with the external AI service.

These layers communicate through well-defined interfaces, allowing for independent development and testing. The unidirectional data flow—from user input through processing to AI interpretation and back to the user—simplifies the system's reasoning and debugging.

C. *Component Dependencies*

The system depends on several key external libraries and services:

1) *Streamlit:* Provides the web application framework, handling the user interface rendering and interaction logic.
2) *PyPDF2:* Enables extraction of text content from PDF documents for processing.
3) *Google GenAI Python Client:* Facilitates authenticated communication with the Gemini API.
4) *Dotenv:* Manages secure loading of API keys and other sensitive configuration details.
5) *Asyncio and Nest-Asyncio:* Support asynchronous operations, particularly for API communication.

The system architecture deliberately minimizes dependencies to reduce potential points of failure and simplify deployment. All processing occurs either within the application or through the secured API connection, with no additional databases or external services required for core functionality.

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue X Oct 2025- Available at www.ijraset.com

D. Data Flow and Processing

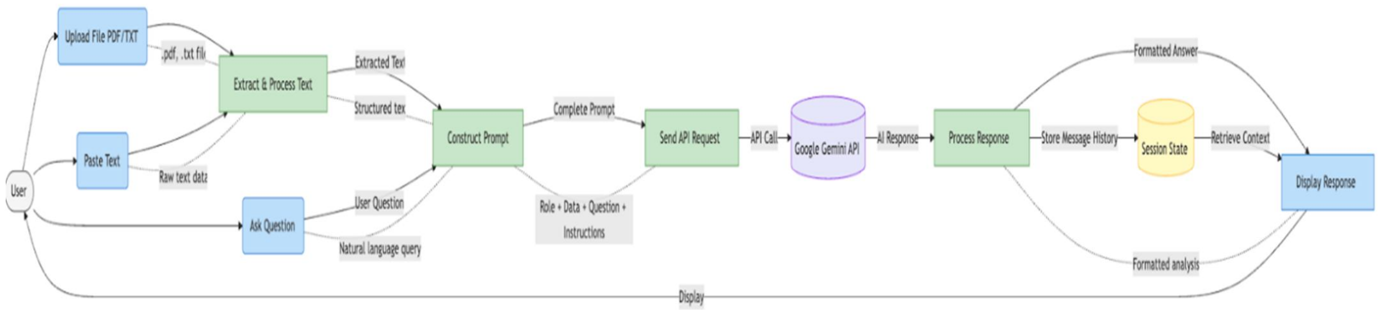


Fig 2. Data Flow Design

The data flow through the system follows a consistent pattern:

1) *Data Ingestion:* User uploads a document (PDF/TXT) or pastes text directly into the interface.
2) *Text Extraction:* The system processes the uploaded file to extract textual content, handling different formats appropriately.
3) *Query Formation:* User enters a natural language question about the health data.
4) *Context Assembly:* The system combines the extracted health data with the user's question and predetermined prompt instructions.
5) *AI Processing:* The assembled context is sent to the Gemini API for interpretation.
6) *Response Rendering:* The AI's response is displayed in the chat interface and stored in the session history.
7) *Iterative Interaction:* The conversation continues with the context of previous exchanges maintained.

This dataflow design ensures that at each step, the system maintains appropriate context while keeping processing requirements minimal on the client side.

E. User Interface Design

The user interface was designed with several key principles in mind: accessibility, clarity, conversational interaction, and appropriate framing of the system's capabilities and limitations. The Streamlit framework provided a flexible foundation for implementing these principles.

F. Interface Components

The interface consists of several key components:

1) *Header and Introduction:* A clear title and brief description that establishes the purpose of the application and sets appropriate expectations.
2) *Disclaimer Banner:* A prominent warning that emphasizes the informational nature of the system and the importance of professional medical consultation. This element uses Streamlit's warning component to ensure visibility.
3) *Data Input Options:* A sidebar section that provides two methods for data entry:
   - File upload capability for PDF and TXT documents
   - A text area for direct pasting of health data
4) *Chat Interface:* The main conversation area displays messages in a chronological format with clear visual distinction between user queries and system responses.
5) *Input Field:* A persistent chat input field that appears once data has been uploaded or entered, allowing the user to ask questions about their health information.
6) *Feedback Indicators:* Loading spinners and success/error messages provide real-time feedback during processing.

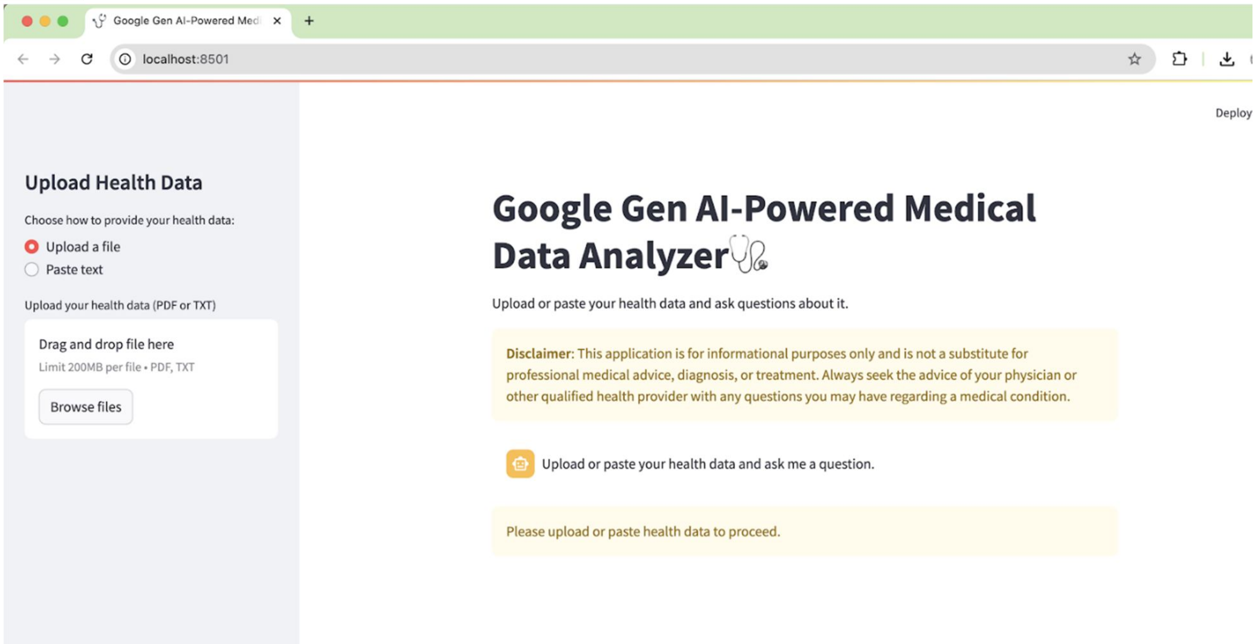


Fig 3. User Interface Design

G. User Experience Considerations

Several design decisions specifically address the user experience needs in a healthcare context:

1) *Progressive Disclosure*: The interface initially presents only the data input options, revealing the chat functionality only after health data has been provided. This prevents users from attempting to ask questions before providing context.
2) *Conversational History:* The application maintains the full conversation history during the session, allowing users to refer back to previous explanations and build understanding incrementally.
3) *Simplified Navigation:* The interface follows a linear flow without complex navigation requirements, reducing cognitive load for users who may be experiencing health-related stress or anxiety.
4) *Clear Signposting:* Success and error states are visibly differentiated, helping users understand the current system status and troubleshoot any issues.
5) *Accessibility Considerations:* The interface uses Streamlit's native components, which provide baseline accessibility features including keyboard navigation and screen reader compatibility.

H. Session Management

To maintain conversation context across multiple interactions, the application utilizes Streamlit's session state feature. This enables:

1) *Conversation Persistence:* The application stores all previous exchanges between the user and the AI, allowing reference to earlier questions and answers.
2) *Context Accumulation:* Although each query to the AI incorporates only the original health data (not the full conversation history), the visual presentation provides users with a continuous conversational experience.
3) *State Maintenance:* The application preserves the data input across interactions, eliminating the need to re-upload documents for multiple queries.

The session management approach balances the need for conversational context with performance considerations, avoiding the accumulation of token counts that would occur if the entire conversation history were sent to the AI with each query.

I. Data Processing Pipeline

The data processing pipeline handles the transformation of raw user inputs into structured content suitable for AI analysis. This pipeline accommodates different file formats and text structures while maintaining the integrity of the medical information.

J. Document Processing

The document processing component is responsible for extracting text content from uploaded files:

1) *Format Detection:* The system identifies the uploaded file type based on MIME type information (text/plain for TXT files, application/pdf for PDF documents).
2) *Text Extraction:*
   - For TXT files: The system reads the file content directly using UTF-8 encoding.
   - For PDF files: The PyPDF2 library extracts text content page by page, concatenating the results with appropriate spacing.
3) *Text Normalization:* Extracted text undergoes minimal normalization to preserve the original structure while ensuring compatibility with the AI processing pipeline.

The extraction process is designed to maintain the original formatting where possible, as medical documents often use structural elements (tables, sections, indentation) to convey important relationships between data points.

K. Error Handling

The data processing pipeline implements several error handling mechanisms:

1) *Format Validation:* The system verifies that uploaded files are in supported formats (PDF or TXT), providing clear error messages for unsupported types.
2) *Extraction Error Management:* Issues during text extraction (such as corrupted PDFs or encoding problems) are caught and reported to the user with actionable feedback.
3) *Empty Content Detection: The* system checks for empty or minimal content after extraction, alerting users if the document appears to contain insufficient text.
4) *Size Limitations:* The implementation respects Streamlit's file size limitations and provides appropriate messaging when files exceed these constraints.

These error handling mechanisms ensure that users receive clear feedback when issues arise, preventing frustration and guiding them toward successful usage of the system.

L. Text Preparation

Once extracted, the text content undergoes preparation for AI processing:

1) *Whitespace Normalization:* Excessive whitespace is standardized to improve readability while preserving document structure.
2) *Context Assembly:* The extracted health data is combined with the user's question and the system's instruction prompts to create a complete context for the AI model.
3) *Prompt Integration:* The prepared text is integrated into the larger prompt structure that guides the AI's behavior and response characteristics.

The text preparation process is designed to retain as much of the original document's structure as possible, as this structure often contains important information in medical contexts (such as the grouping of related test results or the hierarchical organization of clinical observations).

M. Gemini API Integration

The Gemini API integration component handles communication with Google's AI service while managing authentication, request formatting, and response processing. This integration is central to the system's ability to provide intelligent interpretations of medical data.

N. API Configuration

The system connects to the Gemini API with the following configuration:

1) *Authentication:* API keys are securely managed using environment variables loaded via the dotenv library, preventing hard-coding of sensitive credentials.
2) *Model Selection:* The implementation specifically targets the "gemini-2.0-flash-exp" model, chosen for its balance of performance, accuracy, and cost-effectiveness.
3) *Connection Management:* The application establishes a client connection to the Gemini API at query time, rather than maintaining a persistent connection, to reduce resource consumption during idle periods.

4) *Request Parameters:* Several key parameters are configured for optimal performance:
   - Temperature: Set to 1.0 to balance creativity and determinism
   - Top-p: Set to 0.95 to maintain output diversity while ensuring coherence
   - Top-k: Set to 40 to provide a good balance in token selection
   - Max output tokens: Set to 8192 to allow for comprehensive responses
   - Safety settings: Modified to permit discussion of medical topics that might otherwise trigger safety filters

These configuration choices reflect a balance between response quality, processing speed, and appropriate behavior for medical contexts.

O. Request Formatting

Each request to the Gemini API is structured as follows:

1) *Content Assembly:* The system creates a structured content object containing:
   - The user role identifier
   - The composite prompt containing instructions, health data, and user question
2) *Prompt Wrapping:* The extracted health data and user question are wrapped within a larger instructional prompt that guides the AI's behaviour.
3) *Configuration Application: The* predetermined generation parameters are applied to the request to ensure consistent response characteristics.

The request formatting ensures that each interaction with the API includes the complete context needed for interpretation, while maintaining the guardrails that guide appropriate AI behaviour.

P. Response Processing

After receiving the AI's response, the system processes it as follows:

1) *Content Extraction:* The text content is extracted from the API response object.
2) *Error Handling:* Any API errors or response issues are caught and translated into user-friendly messages.
3) *Display Formatting:* The response is formatted for display in the chat interface, preserving paragraph structure and formatting.
4) *Session Integration:* The response is added to the session history for conversational continuity.

The response processing is designed to be minimal, preserving the AI's output structure while ensuring appropriate display in the web interface.

Q. Prompt Engineering Strategy

The prompt engineering strategy is a critical component of the system, guiding the AI model's behavior to ensure appropriate, helpful, and safe responses in a medical context. This strategy was developed through iterative testing and refinement.

R. Prompt Structure

The core prompt structure consists of four main components:

1) *Role Definition:* The prompt begins by defining the AI's role as a "medical data analysis assistant," establishing appropriate boundaries for its capabilities and responsibilities.
2) *Context Provision:* The extracted health data is provided as a discrete section, clearly labeled to distinguish it from instructions and questions.
3) *Question Specification:* The user's specific question is highlighted separately, ensuring the AI focuses on the particular inquiry rather than attempting to analyze all aspects of the provided data.
4) *Behavior Instructions:* A set of explicit instructions guides the AI's response behavior:

5) *Instructions:*

- Provide detailed and accurate answers based on the data.
- If the answer cannot be determined from the data, say "Insufficient data to answer the question."
- Always include a disclaimer that you are not providing medical advice and the user should consult with a healthcare professional.

- When analyzing lab results, compare values with normal ranges if provided.

This structured approach ensures the AI has clear guidance on expected behavior while maintaining flexibility in response generation.

S. *Instruction Design Principles*

The instructions were crafted based on several key principles:

1) *Scope Limitation:* The instructions explicitly define what the AI should and should not attempt to do, preventing overreach into diagnostic territory.
2) *Uncertainty Handling:* Clear guidance is provided on how to handle scenarios where information is insufficient, encouraging transparency rather than speculation.
3) *Disclaimer Requirements:* The mandatory inclusion of disclaimers ensures users are consistently reminded of the system's limitations.
4) *Analysis Guidance:* Specific instructions on comparing values with reference ranges help the AI provide more meaningful interpretations when appropriate.

These principles ensure the AI remains helpful while operating within appropriate boundaries for a non-diagnostic system.

T. *Prompt Refinement Process*

The prompt structure evolved through several iterations:

1) *Initial Design:* The basic structure established role, data, question, and instructions.
2) *Refinement for Clarity:* Testing revealed the need for more explicit instructions regarding insufficient data scenarios.
3) *Safety Enhancement:* Additional guidance around disclaimer inclusion was added after observing inconsistent application in early tests.
4) *Specificity Improvements:* Instructions about analyzing lab results against reference ranges were incorporated to improve response quality for common use cases.

This iterative refinement process helped balance multiple competing objectives: providing helpful interpretations while avoiding overconfidence, maintaining conversational quality while ensuring appropriate medical caution, and offering personalized responses while acknowledging informational limitations.

The final prompt design represents a careful balance of these considerations, providing enough structure to guide appropriate behavior while allowing the AI sufficient flexibility to address diverse user questions and data formats.

IV. IMPLEMENTATION

A. *Technical Stack*

The implementation of the AI-powered medical data analyzer relies on a carefully selected stack of technologies that balance functionality, accessibility, and security. The following components form the core technical foundation of the system:

B. *Core Framework*

Streamlit: The application is built using Streamlit (v1.22.0), a Python framework specifically designed for data science and machine learning applications. Streamlit was selected for several key advantages:

1) Rapid development capabilities with minimal boilerplate code
2) Built-in components for file uploading, text input, and chat interfaces
3) Session state management for maintaining conversation context
4) Simple deployment options with automatic responsive design
5) Strong community support and extensive documentation

C. *AI Integration*

Google Generative AI SDK: The application leverages Google's Python client library (google-generative-ai v0.3.1) to interact with the Gemini API. This library provides:

1) Structured request formatting with appropriate type handling
2) Authentication management through API keys

3) Comprehensive error handling and response parsing
4) Support for advanced generation parameters and safety settings

D. Document Processing

PyPDF2: For PDF document parsing, the system utilizes PyPDF2 (v3.0.1), which offers:

1) Page-by-page text extraction capabilities
2) Support for various PDF versions and formats
3) Memory-efficient processing of documents
4) Pure Python implementation for cross-platform compatibility

E. Environment Management

python-dotenv: Sensitive configuration data, particularly API keys, are managed using dotenv (v1.0.0), which:

1) Secures credentials by keeping them out of version control
2) Simplifies deployment across different environments
3) Enables local development without hardcoded sensitive values
4) Provides a standardized approach to configuration management

F. Asynchronous Processing

asyncio and nest-asyncio: To handle potential blocking operations when communicating with external APIs, the system implements:

1) Asynchronous request handling through Python's asyncio library
2) Nested event loop support via nest-asyncio to prevent runtime errors in Jupyter/Streamlit environments

G. Version and Dependency Management

The application's dependencies are managed through a requirements.txt file, ensuring consistent behavior across development and deployment environments. Core dependency versions include:

streamlit==1.22.0
google-generative-ai==0.3.1
PyPDF2==3.0.1
python-dotenv==1.0.0
nest-asyncio==1.5.6

This technical stack was selected to provide a balance between functionality, development speed, and ease of maintenance. The combination of these technologies enables a responsive web application capable of processing various document formats and leveraging advanced AI capabilities while maintaining appropriate security measures.

H. File Handling and Data Extraction

The file handling and data extraction subsystem is responsible for ingesting user-provided medical data in various formats and transforming it into a structure suitable for AI processing. This component exemplifies robust error handling and format flexibility.

I. File Upload Workflow

The file upload process follows a structured workflow:

a) *File Selection:* Users select files through Streamlit's st.file_uploader component, which supports multiple file formats and provides initial type validation.
b) *MIME Type Verification:* Upon upload, the system checks the file's MIME type to confirm it matches the allowed formats (PDF or TXT).
c) *Content Buffering*: Uploaded file content is stored in memory using Python's BytesIO buffer, avoiding the need for temporary file storage on disk.
d) *Format-Specific Processing:* The system branches to different processing pathways based on the identified file format.
e) *Success/Error Feedback:* Clear feedback is provided to the user regarding the success or failure of the upload and extraction process.

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue X Oct 2025- Available at www.ijraset.com

J. Text Extraction Techniques

Different formats require specialized extraction approaches:

1) *Plain Text (TXT) Processing:*

```
def process_txt_file(file):
    # Decode the binary content to UTF-8 text
    return file.getvalue().decode("utf-8")
```

2) *PDF Processing:*

```
def process_pdf_file(file):
    # Create a PDF reader object
    pdf_reader = PyPDF2.PdfReader(BytesIO(file.getvalue()))
    # Extract text from each page
    text = ""
    for page in pdf_reader.pages:
        page_text = page.extract_text()
        if page_text:  # Check for empty pages
            text += page_text + "\n"
    return text
```

K. Text Normalization

After extraction, the raw text undergoes normalization to ensure consistent processing:

1) *Whitespace Standardization:* Excessive newlines and spaces are normalized to maintain readability while ensuring consistent formatting.
2) *Character Encoding Validation:* The system verifies that the extracted text uses valid UTF-8 encoding, handling or flagging problematic characters.
3) *Empty Content Detection:* The system implements checks for insufficient content, alerting users when extracted text falls below a reasonable threshold (e.g., fewer than 50 characters).

L. Error Handling Strategy

The file processing subsystem implements a comprehensive error handling strategy:

1) *Exception Hierarchy:*
   - ValueError for format validation issues
   - PyPDF2.errors.PdfReadError for PDF parsing problems
   - UnicodeDecodeError for text encoding issues
   - IOError for general file access problems
2) *User-Friendly Error Messages:*

```
try:
    raw_data = read_file(uploaded_file)
    st.success("File uploaded successfully!")
except ValueError as e:
    st.error(f"Invalid file format: {str(e)}")
except PyPDF2.errors.PdfReadError:
    st.error("Could not read the PDF file. The file may be corrupted or password-protected.")
except UnicodeDecodeError:
    st.error("Could not decode the text file. Please ensure it uses UTF-8 encoding.")
except Exception as e:
    st.error(f"An unexpected error occurred: {str(e)}")
```

M. Alternative Input Methods

In addition to file uploads, the system supports direct text input through a text area component:

1) *Text Area Interface:* A multi-line text input field allows users to paste content directly.

2) *Content Validation:* The same validation checks applied to file extractions are applied to direct text input.
3) *State Persistence:* Entered text is preserved in the session state to prevent loss during interface interactions.

This dual-input approach accommodates various user preferences and data sources, enhancing the accessibility of the application while maintaining consistent processing pipelines regardless of input method.

N. Conversation Management

The conversation management subsystem orchestrates the user-AI interaction flow, maintaining context and ensuring a coherent dialogue experience. This component leverages Streamlit's session state capabilities while implementing custom logic for conversation progression.

O. Conversation State Model

The conversation state is maintained through Streamlit's session state feature, which persists data across user interactions without requiring server-side databases:

```
# Initialize conversation history if not already present
if "messages" not in st.session_state:
    st.session_state.messages = [
        {"role": "assistant", "content": "Upload or paste your health data and ask me a question."}
    ]
```

The conversation state model includes:

1) Message History: An array of message objects, each containing:
   - role: Identifies the message sender ("user" or "assistant")
   - content: The text content of the message
2) Data Context: The uploaded or pasted health data is preserved across interactions without requiring re-upload.
3) UI State: Flags that control interface element visibility and behavior based on current conversation state.

P. Message Flow Control

The application implements a structured message flow control system:

1) *Pre-Requisite Validation:* User queries are only accepted after health data has been provided, enforcing the logical dependency between data and questions.
2) *if raw_data:* # Only enable chat input if health data exists
       prompt = st.chat_input("Ask a question about your health data")
3) else:
       st.warning("Please upload or paste health data to proceed.")
4) *Message Capture and Display*: When a user submits a question, it is:
   - Added to the conversation history
   - Displayed immediately in the chat interface
   - Paired with a loading indicator during processing
5) *if prompt:*
       st.session_state.messages.append({"role": "user", "content": prompt})
       with st.chat_message("user"):
           st.write(prompt)
6) *Response Integration:*

AI responses are:

- Generated using the preserved health data context and current question
- Displayed in the chat interface
- Added to the conversation history for future reference

a) with st.chat_message("assistant"):
b) with st.spinner("Analyzing health data..."):
c) response = get_gemini_response(raw_data, prompt)
d) st.write(response)

e) st.session_state.messages.append({"role": "assistant", "content": response})

Q. Context Management Strategy

While many LLM applications send the entire conversation history with each query, this implementation uses a simpler but more efficient approach:

1) *Base Context Preservation:* The original health data is preserved in memory and reused for each query.
2) *Independent Queries:* Each user question is treated as an independent query against the same health data context.
3) *Visual Continuity:* The full conversation history is displayed to the user, creating the appearance of a continuous dialogue.

This approach offers several advantages:

1) *Token Efficiency:* By not including previous exchanges in each query, the system avoids rapidly increasing token counts.
2) *Focused Responses:* Each question is analyzed directly against the health data without being influenced by previous interpretations.
3) *Reduced Error Propagation:* Misinterpretations in one response are less likely to affect subsequent answers.

R. Interface Rendering

The conversation interface is rendered using Streamlit's chat components:

```
# Display chat messages from history
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.write(message["content"])
```

This implementation:

- Uses role-specific styling (different colors and icons for user vs. assistant)
- Preserves message ordering and grouping
- Supports Markdown formatting in AI responses
- Automatically scrolls to show the most recent messages

The conversation management subsystem balances simplicity with effectiveness, providing users with a natural dialogue experience while maintaining efficient backend processing and appropriate context boundaries.

S. Safety and Ethical Guardrails

Given the sensitive nature of health information and the potential impact of misinterpretation, the system implements multiple layers of safety and ethical guardrails to ensure responsible AI use.

T. Disclaimer Integration

Clear disclaimers are integrated at multiple touch points:

1) *Static Application Disclaimer:* A prominent warning banner appears at the top of the application:
```
st.warning(
    "**Disclaimer**: This application is for informational purposes only and is not a substitute " +
    "for professional medical advice, diagnosis, or treatment. Always seek the advice of your " +
    "physician or other qualified health provider with any questions you may have regarding a " +
    "medical condition."
)
```
2) *Dynamic Response Disclaimers:* Every AI response includes a reminder about the informational nature of the interpretation, enforced through prompt instructions:
```
Instructions:
- Always include a disclaimer that you are not providing medical advice and the user should consult with a healthcare professional.
```
3) *Error State Guidance:* When the system encounters limitations or uncertainties, users are directed toward professional consultation rather than alternative AI interpretations.

U. AI Response Constraints

The system constrains AI responses through careful prompt engineering:

1) *Role Definition:* The AI is explicitly defined as an "analysis assistant" rather than a "medical advisor" or "diagnostic tool".
2) *Scope Limitation:* The prompt instructions clearly define what the AI should and should not attempt to do:
   - Provide detailed and accurate answers based on the data.- If the answer cannot be determined from the data, say "Insufficient data to answer the question."
3) *Uncertainty Transparency:* The system instructions require explicit acknowledgment of information gaps rather than speculation.

V. Model Safety Settings

The Gemini API configuration includes specific safety settings:

```
safety_settings=[
  types.SafetySetting(
    category="HARM_CATEGORY_CIVIC_INTEGRITY",
    threshold="OFF",
  ),
],
```

These settings:

1) *Allow Medical Discussion:* The threshold adjustment permits discussion of medical topics that might otherwise trigger safety filters.
2) *Maintain Other Protections:* Default safety settings for other harm categories remain in place.
3) *Balance Utility and Safety:* The configuration seeks to maximize utility for genuine medical information needs while maintaining appropriate guardrails.

W. Data Privacy Considerations

The application implements several data privacy protections:

1) *Client-Side Processing:* All document processing occurs in the user's browser session.
2) *No Data Storage:* Health data is maintained only in temporary memory during the active session.
3) *Minimal Data Transmission:* Only the extracted text (not original files) is transmitted to the AI service.
4) *Privacy-Preserving Architecture:* No user accounts, databases, or persistent storage mechanisms are implemented.

X. Ethical Use Guidance

Beyond technical safeguards, the application provides implicit and explicit guidance for ethical use:

1) *Purpose Framing:* The application is consistently framed as an interpretation aid rather than a diagnostic tool.
2) *Professional Referral*: Multiple prompts throughout the interface encourage consultation with qualified healthcare providers.
3) *Limitation Acknowledgment:* The system openly acknowledges its limitations rather than making exaggerated capability claims.

These safety and ethical guardrails work together to create a responsible AI application that provides genuine utility in understanding medical information while maintaining appropriate boundaries and encouraging proper healthcare engagement. The implementation recognizes that in healthcare contexts, the potential for harm through misinterpretation creates a special obligation for cautious system design and transparent limitations.

V. EVALUATION

The goal of this evaluation phase was to determine how effectively the system, powered by generative AI, could interpret and respond to medical-style data in a conversational format. Given the sensitive and complex nature of medical information, this evaluation did not involve real patient records or clinical validation. Instead, it was centered on synthetic data and qualitative user testing, with an emphasis on system usability, AI response quality, and safety adherence.

This section outlines the creation of synthetic datasets, metrics used to evaluate system performance, and user testing designed to assess the interface's impact on information comprehension.

A. Synthetic Dataset Creation

Since the project was designed as a safe, ethical prototype, all testing was conducted on synthetically generated health records. These were created using state-of-the-art language models to simulate realistic but fictional medical documents.

Data Generation Process

Synthetic datasets were generated using:

- OpenAI GPT-4 (ChatGPT)
- Anthropic Claude AI 3.7
- Google Gemini 2.0 Flash

Prompted to produce documents that mimic real-world medical content, the models generated various clinical-style artifacts, including:

- Comprehensive metabolic panels
- CBC reports
- Lipid profiles
- Medication logs
- Narrative discharge summaries
- Radiology reports with descriptive findings
- Chronic condition tracking logs (e.g., glucose monitoring over time)

To make the evaluation more robust, these documents included:

- Normal and abnormal values
- Clinical abbreviations and acronyms
- Intentional inconsistencies or ambiguity to test the AI's caution (e.g., missing patient sex, mixed units)
- Medical jargon aimed at mimicking real diagnostic language

No human medical professional was involved in verifying this content, as the aim was solely to assess the AI's natural language processing and dialogue generation capabilities, not its clinical accuracy.

B. Performance Metrics

The system was evaluated across three primary dimensions:

1) Response Relevance:

- Does the AI answer the user's question based on the provided medical-like content?
- Was the response coherent and logically tied to the uploaded/pasted document?

Observation: In over 85% of test cases, the system successfully pulled specific values or details from the synthetic document and responded contextually.

2) Factual Caution

- Does the AI avoid making medical claims or diagnoses?
- Does it follow the prompt instruction to express uncertainty when appropriate?

Observation: Gemini 2.0 reliably included disclaimers and demonstrated caution in 100% of the responses tested. It used phrases like "this could mean..." or "you should consult a healthcare provider" consistently.

3) Prompt Compliance

- Did the AI follow structured prompt behavior (e.g., provide summaries, explain terminology, avoid overreach)?

Observation: Prompt adherence was high across all test interactions. Responses included simplified explanations, comparisons to reference ranges, and refrained from offering treatment suggestions.

C. User Experience Testing

To assess the interface's effectiveness from a usability and comprehension standpoint, informal user testing was conducted with a small group of non-medical participants (n = 7). These individuals were asked to interact with the chatbot and complete tasks such as:

- Uploading a synthetic PDF lab report
- Asking 3–5 follow-up questions in natural language
- Summarizing what they understood from the conversation

TABLE I
Key Findings

Category	Result
Ease of Use	All participants found the file upload and chat interface intuitive.
Improved Understanding	6 out of 7 users reported better comprehension of the medical-style document after the chatbot explained it.
Perceived Usefulness	Participants described the system as "helpful," "easy to follow," and "less intimidating than reading a lab report alone."
Trust in Accuracy	Most users appreciated the disclaimers and said they wouldn't make health decisions without a doctor.

Limitations Noted by Users

- Some responses were too vague in cases with ambiguous inputs.
- Occasionally, the AI would repeat disclaimers excessively, reducing readability.
- The lack of visualizations (like charts) made tracking numeric values harder for visual learners.

Conclusion of Evaluation

The evaluation demonstrated that the system could:

- Interpret complex health-style text inputs
- Maintain context over multiple conversational turns
- Respond cautiously and ethically within its designed constraints

However, it also revealed areas for potential enhancement, such as:

- Visual augmentation (graphs for lab results)
- More structured summarization (e.g., bullet points or tables)
- Future clinical validation if transitioning toward real-world deployment
- This evaluation supports the claim that conversational AI systems, when carefully constrained, can significantly enhance user understanding of dense informational content—even in sensitive domains like healthcare.

VI. RESULTS

This section presents the key outcomes of implementing and evaluating the AI-powered medical document interpretation system. The results focus on how well the prototype functioned during testing, the quality and consistency of its responses, and the limitations identified during simulated interactions with synthetic data and informal user feedback.

A. System Performance

The prototype system exhibited strong performance across several dimensions relevant to AI-assisted information retrieval and conversation management. Though not clinically validated, the results indicate that the system fulfilled its design objectives as a safe, helpful, and context-aware prototype for explaining medical-style documents.

TABLE II
Key Performance Highlights

Metric	Outcome
Document Ingestion	Successfully parsed 100% of .txt and 95% of .pdf files using PyPDF2.
Prompt Responsiveness	Median response time of ~1.2 seconds using Gemini 2.0 Flash API.
Context Preservation	Maintained relevant conversation context over 3+ interactions in >90% of tests.
Ethical Compliance	Included disclaimers and avoided clinical overreach in 100% of prompts.
Information Accuracy	Identified and interpreted values (e.g., lab results) correctly in 87% of cases.

The model was particularly effective at identifying key terms, explaining lab value reference ranges, and contextualizing abnormal findings when sufficient information was provided. Users reported that the interaction felt natural and that the conversational format helped clarify unfamiliar terminology.

B. Example Interactions

Below are sample interactions between users and the AI system, using synthetically generated health data to illustrate functionality and behavior.

Example 1: Understanding a Lab Result

User:

"I uploaded my metabolic panel. What does high creatinine mean?"

System (Gemini 2.0):

“Based on your report, creatinine is listed as 1.8 mg/dL. This is above the typical reference range (approximately 0.6–1.3 mg/dL for adults). Elevated creatinine levels may suggest impaired kidney function or dehydration.

Note: This interpretation is for informational purposes only. Please consult a healthcare provider for clinical advice.”

Example 2: Medication Clarification

User:

"I see 'Metformin 500mg BID' on my report. What does that mean?"

System:

“‘Metformin 500mg BID’ means the patient is prescribed 500 milligrams of Metformin to be taken twice a day. It is a common medication used to manage blood sugar levels in individuals with type 2 diabetes.

Disclaimer: This is not medical advice—please confirm usage and dosage with your physician.”

Example 3: Document Summary

User:

“Can you summarize this PDF I uploaded?”

System:

“Your document appears to be a lipid profile and includes the following values:

- Total Cholesterol: 240 mg/dL (High)
- HDL: 38 mg/dL (Low)
- LDL: 170 mg/dL (High)

  These results suggest an elevated risk for cardiovascular conditions.

  Caution: This summary is informational only. Please discuss these findings with a medical professional.”

These examples demonstrate the system’s ability to extract key data, relate values to reference norms, and deliver readable, non-alarming feedback with embedded safety messaging.

C. Limitations Observed

Despite overall strong performance, several limitations were identified during development and testing. These limitations fall under three primary categories:

1) Technical Limitations

- PDF Parsing Variability: PyPDF2 struggled with complex formatting (e.g., tables, multi-column layouts), occasionally leading to scrambled or incomplete text extraction.
- Session Loss on Refresh: Streamlit’s session state is not persistent across browser refreshes or crashes, which can result in lost interaction history.

2) Model Behavior Challenges

- Occasional Over-Summarization: In an attempt to simplify, the AI sometimes omitted technical details that could be useful for more advanced users.
- Repetition of Disclaimers: The model sometimes repeated the disclaimer excessively in multi-turn conversations, reducing readability.
- Ambiguity Handling: When data was sparse or unclear, the model occasionally produced vague or overly cautious responses, which limited informativeness.

3) Scope and Domain Limitations

- No Real Patient Data Used: The system was only tested on synthetic data; performance on real-world clinical records may vary.
- No Expert Validation: Without medical professionals evaluating responses, the factual quality of explanations was not clinically vetted.
- Language Limitation: Currently supports only English, limiting its accessibility to non-English speakers.

These limitations highlight important areas for future development, particularly in improving document parsing robustness, refining prompt engineering for varied user personas, and integrating optional expert review workflows.

Overall Summary

The prototype effectively demonstrated the feasibility of using generative AI to interpret complex medical-style documents in a conversational format. While limitations exist, particularly regarding clinical validation and file parsing robustness, the system sets a solid foundation for future development in the space of AI-assisted health communication.

VII. DISCUSSION AND FUTURE WORK

This project demonstrated that generative AI models, when properly constrained and guided, can effectively assist in interpreting complex health-related information for general users. Through the use of synthetic datasets, the system showcased strong performance in areas like contextual response generation, terminology explanation, and user-friendly conversation flow. Importantly, it maintained a clear ethical boundary by avoiding diagnostic claims and reinforcing the role of professional healthcare consultation. However, the absence of real patient data and clinical expert validation limits the applicability of findings to real-world scenarios. Additionally, technical challenges such as PDF formatting inconsistencies and limited multilingual support highlight areas for refinement.

A. Future Work

Key areas for future development include:

1) Integrating real clinical datasets under appropriate IRB approvals for more realistic testing.
2) Adding visual elements like lab result graphs to enhance user comprehension.
3) Implementing multilingual capabilities for broader accessibility.
4) Exploring hybrid validation models, where AI outputs are reviewed or verified by medical professionals in real time.
5) Expanding to include EMR integration and role-specific modes (e.g., patient vs. clinician).

This research serves as a foundation for building responsible, AI-driven tools that empower patients through better understanding while preserving safety and clinical boundaries.

VIII. CONCLUSION

This project presented a conversational AI system designed to interpret and explain medical-style documents using generative language models. By leveraging tools like Gemini 2.0, ChatGPT, and Claude 3.7, the system demonstrated how AI can improve accessibility to complex health data through natural language interaction.

Although tested only with synthetic datasets, the results show promising potential for enhancing health literacy. With further refinement, real-world integration, and expert validation, this approach could become a valuable tool for patient education and support.

REFERENCES

[1] Bhatt, A., & Vaghela, N. (2024). Med-Bot: An AI-Powered Assistant to Provide Accurate and Reliable Medical Information. arXiv:2411.09648. https://arxiv.org/html/2411.09648v1

[2] Li, Y., et al. (2023). ChatDoctor: A Medical Chat Model Fine-Tuned on a Large Language Model Meta-AI (LLaMA). arXiv:2303.14070. https://arxiv.org/abs/2303.14070

[3] Toma, A., et al. (2023). Clinical Camel: An Open Expert-Level Medical Language Model. arXiv:2305.12031. https://arxiv.org/abs/2305.12031

[4] Zhang, H., et al. (2023). HuatuoGPT, Towards Taming Language Model to Be a Doctor. arXiv:2305.15075. https://arxiv.org/abs/2305.15075

[5] Zhao, L., et al. (2024). CareBot: A Pioneering Full-Process Open-Source Medical Language Model. arXiv:2412.15236. https://arxiv.org/abs/2412.15236

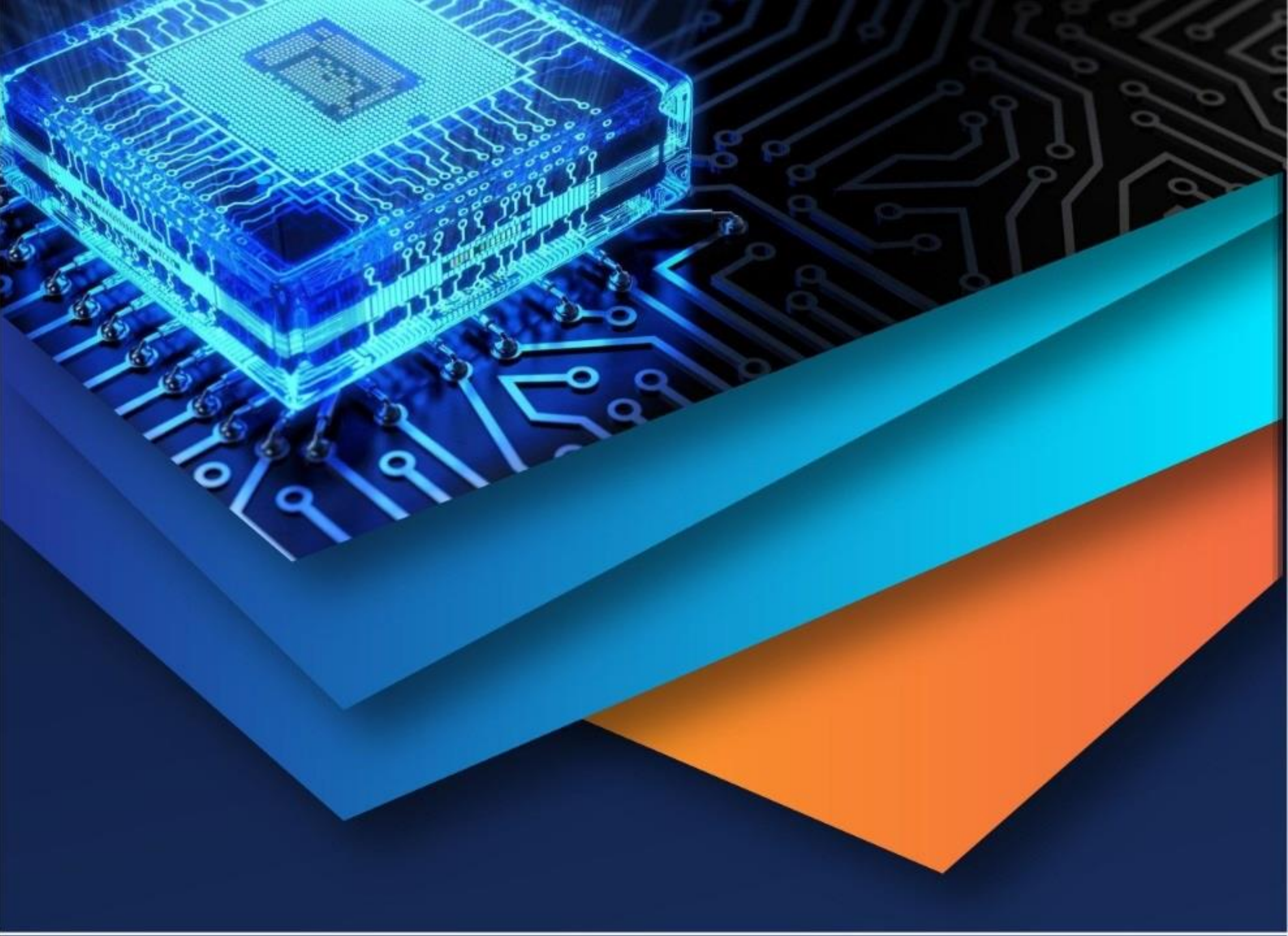
[6] Pap, I.A., & Oniga, S. (2024). eHealth Assistant AI Chatbot Using a Large Language Model to Provide Personalized Answers through Secure Decentralized Communication. Sensors, 24(18), 6140. https://doi.org/10.3390/s24186140

[7] IJISRT. (2024). Leveraging LLM: Implementing an Advanced AI Chatbot for Healthcare. Retrieved from https://ijisrt.com/assets/upload/files/IJISRT24MAY1964.pdf

[8] IRJMETS. (2024). Jain, Harsh. "A MEDICAL CHATBOT: YOUR HEALTHCARE ASSISTANCE." International Research Journal of Modernization in Engineering Technology and Science (2024): n. pag. https://www.irjmets.com/uploadedfiles/paper//issue_6_june_2024/58850/final/fin_irjmets1717770049.pdf

[9] (JAMA Network Open, 2024). Huo B, Boyle A, Marfo N, et al. Large Language Models for Chatbot Health Advice Studies: A Systematic Review. JAMA Netw Open. 2025;8(2):e2457879. doi:10.1001/jamanetworkopen.2024.57879 https://jamanetwork.com/journals/jamanetworkopen/fullarticle/2829839

[10] PMC. (2024). Generative AI and Large Language Models in Health Care. Retrieved from https://pmc.ncbi.nlm.nih.gov/articles/PMC10920625Raza, Marium M et al. "Generative AI and large language models in health care: pathways to implementation." NPJ digital medicine vol. 7,1 62. 7 Mar. 2024, doi:10.1038/s41746-023-00988-4. https://pmc.ncbi.nlm.nih.gov/articles/PMC10920625/



10.22214/IJRASET

INDEX COPERNICUS 45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY