# Technical Report

---

## Team 16 : AN2I

**Collaborators** :

- **Ilyes DJERFAF**
- **Nazim KESKES**
- **Ahmed SIDI AHMED**
- **Azzedine AIT SAID**
- **Imed Eddine KENAI**

**Technical Report**

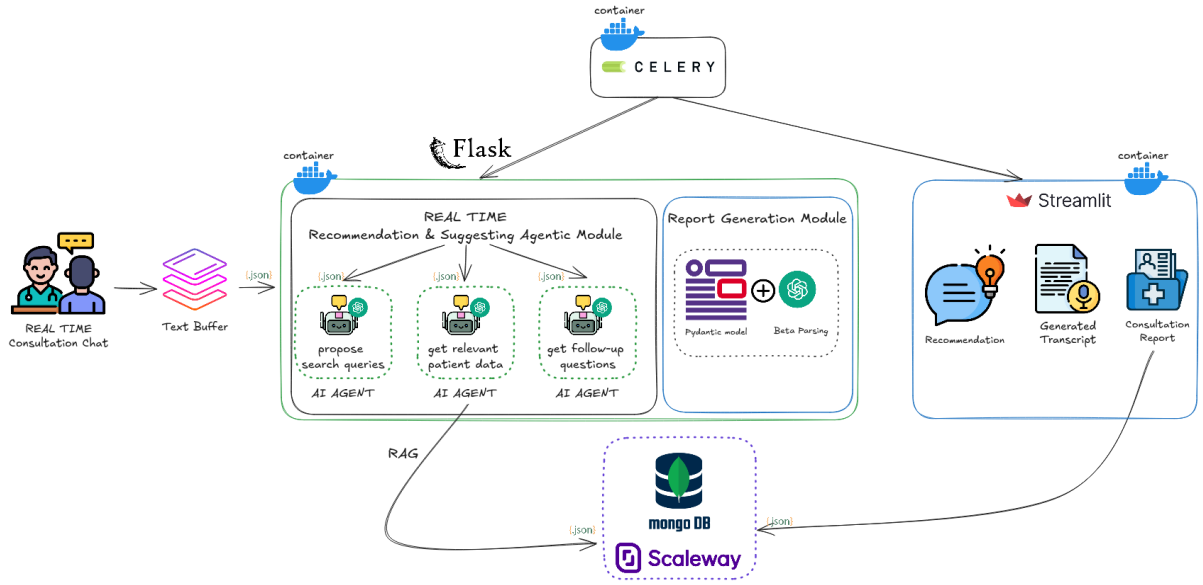
# 1. Summary

This report details the technical implementation of an AI-powered medical assistant designed to streamline clinical workflows, enhance diagnostic accuracy, and reduce administrative burdens on healthcare professionals. The system integrates real-time conversational analysis, automated report generation, and intelligent medical recommendations using modern AI frameworks and cloud-based infrastructure.

**Note :** please refer to the business plan before any further reading.

# 2. System Architecture

## Overview




The Agentic-AI medical assistant consists of multiple modules:

1. **Real-Time Speech-to-Text Engine**: Captures conversations between doctor and patient using buffered processing.

2. **Recommendation & Suggestions Engine**: Suggests follow-up questions based on doctor-patient conversation and the medical history of the patient and recommends treatments or medical adjustments. Each recommendation module is an “**Agent”**.
    - **Agent 1 :** propose search queries
    - **Agent 2 :** retrieve relevant patient data (RAG)
    - **Agent 3 :** get follow-up questions

3. **Report Generation Module**: Structures notes into medical documentation using advanced prompt-engineering in order to generate a **Pydantic** format including :
    - Symptoms
    - Pathology
    - Treatment
    - Intelligent summary & keywords

4. **Storage Module :** the generated reports as well as the patient information is stored in a **MongoDB** instance deployed in **Scaleway.**

5. **Referral letter generation:** generates a customizable referral letter instantly using AI-generated summaries based on previously stored reports.

### Data Flow

1. **Input**:
    - Doctor-patient conversation captured in Real-Time via microphone.
    - Summary of the historical data record of the patient.
2. **Processing**:
    - Audio translated to text and streamed using a buffered process.
    - AI Agents :
        - analyze text buffer to extract key medical insights.
        - Cross-reference with external medical data
        - retrieve historical patient information (RAG)
    - Report structured into Symptoms, Diagnosis, Treatment, and Summary.
    - Store summaries as an embedding version to be used for RAG-similar cases to retrieve relevant information
3. **Output**:
    - Generated report editable by the physician.
    - Summary stored in patient records for future reference.
    - Global Summary of patient history

---

# 3. Technology Stack

### Programming Languages

- Python

### Frameworks and Libraries

- Mistral AI (language model)
- OpenAI SDK (lightweight and avoided langchain)

- Flask (could’ve worked with fastAPI but flask is better for scalability)
- Streamlit (for the frontend implementation)

### Database and Storage

- MongoDB (for its flexibility to stores data in JSON-like format)

---

# 4. Data Management

### Data Collection

- Voice data recorded during consultations.
- Patient history and treatment plans.
- External medical knowledge sources for AI decision support.(can be defined by the doctor)

### Data Processing Pipeline

1. **Speech-to-Text Conversion**: AI model transcribes spoken words.
2. **Natural Language Understanding (NLU)**: Answer all what a doctor is expecting (Extracts medical terms, symptoms, and diagnoses and generates output based on the defined format) using an LLM agent.
3. **Post-Processing**: Summarizes key findings, formats report.

### Data Privacy & Compliance

- No code interaction between tasks (API calls)

**Note :** all the events, messages, data sent to the LLM, data extracted from the LLM are JSON format. This was possible thanks to the structured output function ( Beta-Parsing) implemented in the OpenAI sdk.

---

# 5. Deployment Strategy

### Infrastructure

- **Frontend** Streamlit app deployed a docker container
- **Backend** Flask API managing all the necessary functions

### Integration

- API-based communication for seamless integration
- the backend is dockerized
- Task manager celery is managing the API functions as tasks and run them when special events happen on the front

### Scalability Considerations

- Microservices-based architecture for modular and scalable expansion.
- For scalability, more containers can be created and managed with a load balancer.

---

# 6. Future Roadmap

## Short-Term Goals

#### Key Metrics

- **Transcription Accuracy**
- **Recommendation Relevance**: Measuring physician adoption of AI-suggested treatments.
- **Processing Latency**: Ensuring fast response time
- **User Satisfaction**: Surveys and feedback from early adopters.

#### Testing and Validation

- With real physician consultations.
- Continuous model retraining based on user feedback.

## Long-Term Goals

- Minimize LLMs dependencies in agents as possible by leveraging deterministic algorithms leaving only the managing part for the AI Agent
- implement and fine tune any model that is being used through an inference endpoint.
- Host & Deploy models reducing costs (token generation fees, database hosting ,etc..)

## 7. Conclusion

The Agentic AI system medical assistant is designed to alleviate physicians' administrative workload, enhance documentation accuracy, and improve patient care efficiency. With a secure, scalable, and regulation-compliant infrastructure, this solution represents the future of AI-driven healthcare support.