

Lebanese University

Faculty of Engineering III

Electrical and Electronic Department

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Hospital ai dashboard – ai powered report summarization and risk detection

using rule based & nlp logic

by

**Alaa Fadlallah, 6445**

**Batoul Abbas, 6502**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Spring 2024-2025**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Mini Project**

**Dedicated to : Dr. Mohammad Aoude**

**1. Introduction**

**1.1 Problem Statement**

Doctors often deal with hundreds of lengthy and complex patient reports. Manually reading and identifying risks is time-consuming and error-prone.  
This project aims to build an AI-powered system that automatically **summarizes**, **flags risks**, and **suggests recommendations** based on uploaded reports. It simulates hospital workflows and leverages concurrent API handling and model inference.

**1.2 Motivation**

Medical AI tools are usually black-box and slow, and in case of emergencies every second matters, We wanted to build a modular, explainable system that uses both **AI models** and **rule-based logic**, backed by modern concurrency tools like FastAPI’s asynchronous routing and model parallelization, in a way the Dr. benefits from their very own smart assistant.

**1.3 Objectives**

* Build a web app (React frontend + FastAPI backend)
* Upload and analyze medical pdfs / text files
* Use HuggingFace models for summarization and NER
* Apply a rule engine to detect medical risks
* Visualize data and allow PDF export

**2. Design and Implementation**

**2.1 Architecture Overview**

* **Frontend:** React + Tailwind CSS + Chart.js
* **Backend**: FastAPI routes for async analysis
* **Models**: BART summarization, ClinicalBERT embedding, BioBERT-NER
* **Rule Engine:** Keyword to risk logic (prolog style)
* **Database:** PostgreSQL for report storage

**2.2 AI Models Used:**

Our project uses three specialized transformer-based models from HuggingFace Transformers, each contributing a unique functionality to the report analysis pipeline:

**1. BART Large CNN**  
BART (Bidirectional and Auto-Regressive Transformers) is used for **abstractive summarization** of medical reports.  
It generates concise, human-readable summaries by **rephrasing content**, not just extracting sentences. This ensures more natural and clinically meaningful summaries.

**2. ClinicalBERT**  
Derived from BERT, ClinicalBERT is pretrained on **clinical notes and medical literature**.  
Although its output isn't directly shown to the user, we use its **medical-aware embeddings** internally for tasks like semantic similarity and understanding the report context more accurately.

**3. BioBERT NER**  
BioBERT NER is fine-tuned for **Named Entity Recognition (NER)** in biomedical texts.  
It identifies important entities such as **diseases, symptoms, medications, and anatomical terms**. These extracted entities are passed to the risk engine to trigger relevant alerts.

All three models are **downloaded and cached locally** using HuggingFace’s Transformers library.  
This ensures **reliable offline performance**, **fast inference**, and most importantly, **compliance with hospital privacy policies** by avoiding external API calls.  
By keeping all processing on local infrastructure, we minimize data exposure and maintain the confidentiality of sensitive medical records.

**3. Implementation Details**

**3.1 Technical Challenges and Solutions**

* Managing large model downloads
* Ensuring tokenizer/model consistency across runs
* Extracting clean medical text from PDFs
* Combining AI + rule-based output without delay

**3.2 Code Architecture**

backend/

├─ main.py (FastAPI entry point)

├─ services/

│ ├─ summarize.py

│ ├─ ner.py

│ └─ flagger.py

├─ routers/analyze.py

├─ model/report.py

frontend/

├─ pages/

│ ├─ Landing.jsx

│ ├─ Home.jsx

│ ├─ Recommendations.jsx

│ ├─ Analytics.jsx

**4. Testing Methodology**

**4.1 Correctness Verification**

* Verified summaries manually for clinical accuracy
* Tested rule engine against known diseases

**4.2 Performance Testing Protocol**

* Model response time logged + reports saved in database after summarization
* Frontend tested for uploads

**5. Results and Discussion**

* Summaries were accurate
* NER correctly extracted entities
* Rule based engine added interpretability
* Response time: under 5 seconds per full report

**6. Pros and Cons**

* **Pros:**

Fast, transparent AI + rules, clean exportable output

* **Cons:**

Heavy model download, Rule logic needs constant medical update.

**7. Real-world Applications**

* AI triage assistant for emergency departments
* Automated patient record summarizer
* Risk alert system for chronic conditions
* Health data analytics and dashboards for hospitals

**8. Conclusion and Future Work**

* Add full doctor authentication and dashboards
* Integrate LLM (e.g., Mistral or MedPalm) for explanation
* Real-time recommendation generator (e.g., suggest specialists)
* Long-term: integrate vitals, labs, and time-series data

We successfully built a concurrent, modular system that accepts reports, summarizes them, extracts medical entities, flags risks using logic rules, and visualizes the results in a clean web interface. It can serve as a base prototype for smart hospital assistants.