

ABSTRACT

In today's healthcare landscape, there is a continuous generation of massive amounts of medical data in the form of patient reports, lab results, diagnostic summaries, and prescriptions. Much of this information is unstructured, filled with technical terms, and written in complex language that is often difficult for patients—and even sometimes for general practitioners—to understand. As a result, effective communication and timely decision-making can be compromised.

This project introduces an **AI-powered medical report summarization system** that leverages cutting-edge Natural Language Processing (NLP) to convert lengthy and complex medical reports into short, coherent, and easy-to-understand summaries. The system accepts input in diverse formats, including plain text, PDF, and scanned images. If the input is an image or scanned document, Optical Character Recognition (OCR) tools such as **Tesseract** or **Google Vision API** are used to extract the text content.

Once the text is extracted, it is processed through pre-trained transformer-based NLP models like **BERT**, **T5**, or **GPT** from the Hugging Face library to generate meaningful summaries. The system is also equipped with **Named Entity Recognition (NER)** using **spaCy**, which highlights key medical terms such as disease names, medications, and laboratory values. Moreover, the system can translate the summaries into multiple languages using APIs like **Google Translate** or **DeepL**, thus making it inclusive and globally accessible.

The automation of medical report summarization through this AI-powered system is expected to enhance patient understanding, improve healthcare communication, and assist doctors by reducing the time spent reviewing long reports. It is a step toward personalized and accessible healthcare data interpretation.

Table of Contents

S. No.	Table of Contents	Page No.
1.	Introduction	9
1.1	Overview of Medical Report Complexities	9
1.2	Importance of Medical Summarization	9
1.3	Role of AI and NLP in Healthcare	9
1.4	Objectives of the Project	10
1.4	System Input and Output Overview	11
2.	Literature Review and Analysis	12
2.1	Existing Systems and Their Limitations	12
2.2	Importance of NLP in Medical Summarization	12
2.3	Key Papers and Models in NLP-Based Summarization	13
2.4	Summary of Literature Insights	13
3.	System Design	14
3.1	System Components and Their Roles	14
3.2	System Workflow	16
3.3	Future Extensibility and Collaboration	17
4.	Technology Stack	20
4.1	Frontend Technologies	20
4.2	Backend Technologies	20
4.3	AI/ML Tools	21
4.4	OCR Tools	21
4.5	Storage Technologies	22
4.6	Deployment Infrastructure	22-23
5.	System Requirements	24
5.1	Hardware Requirements	24
5.2	Software Requirements	24
5.3	Environmental Requirements	25
5.4	User-End Requirements	25
5.6	Optional Tools for CI/CD and Monitoring	26
6.	Output	27
	Source Code	27-30
	Sample Input	31
	Sample Output	32

1. Introduction

1.1 Overview of Medical Report Complexities

In the modern healthcare ecosystem, medical reports serve as vital documentation of patient health history, diagnostic findings, treatments administered, and follow-up recommendations. These documents are typically generated by healthcare professionals and include clinical notes, laboratory results, radiology reports, prescriptions, and discharge summaries. However, such reports often tend to be extensive and filled with medical jargon, abbreviations, and domain-specific language that is not easily comprehensible to non-specialists.

Furthermore, the structure of these documents can vary greatly depending on the medical institution, the department, and the physician writing them. Some reports are highly structured, following a defined template (e.g., SOAP notes), while others may be semi-structured or completely unstructured, such as narrative summaries. This variability, coupled with the use of technical language, makes it difficult for patients to interpret the content accurately. Even for physicians, going through numerous lengthy reports daily is a time-consuming task, which may lead to delays in diagnosis, patient management, and decision-making.

1.2 Importance of Medical Summarization

Medical summarization is the process of condensing a long, detailed report into a shorter version that captures the most relevant information. The need for effective summarization arises from the ever-increasing volume of clinical data being generated and stored. By summarizing key medical findings and diagnoses, clinicians can access a concise snapshot of the patient's condition, thereby enabling faster and more accurate clinical decisions.

For patients, summarized versions of medical reports serve as an important bridge between clinical content and layman understanding. When patients can understand their health conditions, treatment plans, and follow-up steps more clearly, it significantly boosts health literacy and compliance with medical instructions. Medical summarization is thus pivotal for enhancing communication between healthcare providers and patients, reducing cognitive overload for clinicians, and ensuring that critical information is not overlooked.

1.3 Role of AI and NLP in Healthcare

Artificial Intelligence (AI), particularly Natural Language Processing (NLP), has revolutionized the way unstructured medical text is interpreted and utilized. NLP enables machines to read, understand, and derive meaning from human language, which is critical in processing clinical narratives and reports.

Recent advances in NLP—especially transformer-based models like BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), and T5 (Text-to-Text Transfer Transformer)—have shown exceptional performance in tasks such as summarization, translation, and named entity recognition. These models can understand medical context, identify relevant sentences, and generate human-like summaries without losing critical information.

Additionally, integrating AI-powered NLP into medical systems enables real-time processing of large volumes of patient records, enhances data-driven decision-making, and supports digital transformation in healthcare. When combined with tools like Optical Character Recognition (OCR) for reading scanned documents, and Named Entity Recognition (NER) for highlighting key terms (diseases, drugs, procedures), the AI pipeline becomes a robust solution for handling diverse clinical data.

1.4 Objectives of the Project

This project aims to bridge the gap between complex medical reports and user-friendly interpretation through the following specific objectives:

- **To develop an AI-based medical summarization tool** that can process input in various formats such as plain text, PDF, and scanned images.
- **To integrate OCR capabilities** using tools like Tesseract or Google Vision API to extract text from non-editable image or scanned documents.
- **To incorporate Named Entity Recognition (NER)** using spaCy or similar frameworks to automatically highlight and tag essential medical terms including disease names, medications, diagnostic results, and procedures.
- **To enable multilingual support** through integration with translation APIs (such as Google Translate or DeepL), thus making the summaries accessible to non-English speakers and users from different linguistic backgrounds.

1.5 System Input and Output Overview

Input Example:

A scanned medical report (in PDF format) containing the following unstructured text:

"Mr. Ramesh Kumar, aged 56, presented with complaints of persistent chest pain and shortness of breath. ECG findings revealed ST-elevation myocardial infarction (STEMI). The patient was immediately started on Aspirin and Atorvastatin and underwent a successful coronary angioplasty. Blood pressure at admission was 150/95 mmHg. Discharged on 12/02/2024 with instructions to continue Clopidogrel, Ramipril, and Metoprolol."

Output Summary Generated by the System:

Summary (English):

Patient Ramesh Kumar (56 years old) was diagnosed with a heart attack (STEMI). Treated with Aspirin and Atorvastatin and underwent coronary angioplasty. On discharge, advised to continue Clopidogrel, Ramipril, and Metoprolol.

Highlighted Medical Terms:

- STEMI
- Aspirin, Atorvastatin, Clopidogrel, Ramipril, Metoprolol
- Blood pressure: 150/95 mmHg

This example shows how the system:

- Converts an unstructured scanned report into clean text using OCR,
- Summarizes the key clinical events using NLP,
- Extracts important medical entities using NER,
- And translates the summary if needed.

2. Literature Review and Analysis

2.1 Existing Systems and Their Limitations

In the domain of medical report summarization, traditional systems have primarily relied on **rule-based approaches**, where handcrafted templates or syntactic parsing rules are used to extract key information. These systems were effective in tightly controlled environments but lacked the scalability, adaptability, and contextual understanding required for complex and unstructured medical data.

Such systems typically function by identifying pre-defined keywords or sections in the text, which works well only if the reports strictly follow the same format. Any deviation—such as narrative-style documentation, image-based inputs, or varied report templates—can cause these systems to fail or produce inaccurate results.

Furthermore, many of these conventional tools are **restricted to processing only plain text formats**. They do not support **image-based documents or scanned PDFs**, which are common in hospitals and diagnostic centers. Moreover, there is minimal support for **multilingual processing**, making these systems inaccessible to non-English-speaking patients and caregivers.

Hence, the limitations in traditional summarization systems include:

- Inability to adapt to different document structures or writing styles
- Lack of semantic understanding of medical context
- No support for scanned/image-based inputs
- Absence of multilingual support
- Minimal use of deep learning or AI

2.2 Importance of NLP in Medical Summarization

Natural Language Processing (NLP) plays a transformative role in healthcare data interpretation. Unlike rule-based systems, **modern NLP uses machine learning—especially deep learning—to learn from vast amounts of text**, capturing both syntax and semantics.

In medical summarization, NLP is used to:

- Understand the relationship between symptoms, diagnosis, and treatments
- Identify contextually relevant information, even when phrased differently
- Generate human-like summaries that preserve clinical meaning
- Handle domain-specific terms and abbreviations effectively
- Extract named entities such as diseases, medications, and lab values
- With the advent of **transformer-based architectures**, NLP has taken a quantum leap. Unlike RNNs or LSTMs, transformers handle longer sequences better and model relationships across entire documents more effectively. This has made them ideal for summarization tasks.

2.3 Key Papers and Models in NLP-Based Summarization

Recent advancements in transformer-based NLP models have greatly enhanced the ability to automatically summarize complex documents. The following models are particularly relevant to this project:

T5 (Text-to-Text Transfer Transformer)

Paper: *"Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer"*

Developed by Google Research, T5 treats every NLP task as a text generation task. It excels at summarization by learning how to generate simplified versions of complex input while retaining meaning. T5 is particularly powerful for multi-task learning and supports multilingual summarization with fine-tuning.

BERT (Bidirectional Encoder Representations from Transformers)

Paper: *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"*

BERT introduced bidirectional context understanding by reading the entire sentence before predicting masked words. Although BERT is not a generative model like GPT or T5, it is widely used for **extractive summarization**, where the most relevant sentences are selected from the input document.

GPT Series (Generative Pretrained Transformer)

Papers: OpenAI’s GPT-1, GPT-2, GPT-3, and GPT-4 series

GPT models use a unidirectional transformer decoder to generate human-like text. They are highly effective for **abstractive summarization**, where the model writes new content based on the essence of the original text. GPT models can handle medical domain content, especially when fine-tuned or prompted correctly.

2.4 Summary of Literature Insights

Model	Type	Application in Summarization	Strengths
T5	Abstractive	Converts complex input into coherent summaries	Unified framework, multilingual support
BERT	Extractive	Selects most relevant sentences	Strong contextual embedding
GPT	Abstractive	Generates summaries from scratch	High fluency and coherence

3: SYSTEM DESIGN

This chapter provides a detailed overview of the architecture and flow of the AI-powered medical report summarization system. The system is designed to process a variety of medical report formats—such as PDFs, images, and text files—and generate clear, concise summaries that include highlighted medical terminology and optional language translation. Each component in the architecture plays a specific role in the overall pipeline, and together they ensure a seamless user experience with high accuracy and flexibility.

3.1 System Components and Their Roles

1. Frontend Interface

The **Frontend Interface** acts as the user interaction layer of the system. Built using modern JavaScript frameworks like **React.js** or **Next.js**, it enables users to:

- Upload medical files (text, image, or PDF)
- Select preferred language (for multilingual summary)
- View or download the generated summary
- Highlight extracted medical terms for easy comprehension

It is styled using **Tailwind CSS** or Bootstrap for a responsive and visually appealing layout. The frontend communicates with the backend through REST API calls.

2. OCR Module

The **Optical Character Recognition (OCR) module** comes into play when users upload scanned medical reports or image-based files (e.g., JPEG, PNG, or scanned PDFs). This module uses tools such as:

- **Tesseract OCR** (open source)
- **Google Vision API** (cloud-based, highly accurate)

It extracts unstructured text from the scanned input and sends it to the next stage in the pipeline. OCR accuracy is vital, as it directly affects the quality of the summarization.

3. Summarization Engine

The **core of the system**, this engine is responsible for converting long-form medical documents into shorter, meaningful summaries. It leverages **transformer-based NLP models** such as:

- **T5** or **BART** for abstractive summarization
- **BERT** for extractive summarization
- **GPT** (like GPT-3.5 or GPT-4) for fluent, natural language generation

These models are fine-tuned for medical data or used via pre-trained weights from platforms like **Hugging Face**.

4. NER (Named Entity Recognition) Module

After summarization, the system identifies key medical terms within the summary using **NER**. It employs models from **spaCy**, **ScispaCy**, or even custom-trained NER systems to extract:

- Diseases (e.g., Diabetes, Tuberculosis)
- Medications (e.g., Metformin, Aspirin)
- Body parts, lab values, and procedures

These terms are then highlighted in the summary output for improved readability and user focus.

5. Translation Layer (Optional)

For accessibility and inclusivity, the system supports **multilingual output**. This optional module allows users to translate summaries into languages such as Telugu, Hindi, Tamil, or international languages like Spanish and French.

APIs used:

- **Google Translate API**
- **DeepL API** (for more context-aware translations)

This module ensures that patients from diverse linguistic backgrounds can understand their health information without confusion.

6. Database and File Storage

The system needs robust storage to:

- Save uploaded reports
- Log generated summaries
- Maintain audit trails and history for future retrieval

Recommended choices:

- **PostgreSQL / MongoDB** for metadata and logs
- **Firebase / AWS S3** for storing PDFs, images, and generated files

This modular design also allows easy export or sharing of summaries.

7. Deployment Infrastructure

To ensure high availability, fast response times, and scalability, the system is deployed on modern cloud infrastructure using:

- **Docker** for containerization
- **Kubernetes** for orchestration and auto-scaling
- **AWS EC2 / GCP Compute Engine** for hosting services
- **Vercel / Netlify** for frontend hosting

These tools ensure that the system is production-ready and can handle concurrent users effectively.

3.2 System Workflow

The system follows a streamlined and modular workflow:

Input:

User uploads a medical file via the frontend. File formats supported:

- PDF (digital or scanned)
- JPG/PNG images
- Plain text (.txt)

Processing Pipeline:

1. **OCR** (if scanned/image input)
2. **Summarization** using transformer-based NLP model
3. **NER** tagging of medical terms
4. **Translation** (optional, based on user input)

Output:

- A **clean, readable summary** of the original report
 - Key **medical terms highlighted**
 - **Downloadable PDF or web view** of the output
 - Optionally **translated** version for non-English users
-

3.3 Future Extensibility and Collaboration

The system has been designed with modularity in mind, allowing easy integration of new components such as:

- Voice-to-text modules for dictated medical notes
- Integration with Electronic Health Record (EHR) systems
- Team-based dashboard for hospital use (e.g., doctor-nurse collaboration)
- Patient-specific timeline generation based on summaries

The codebase and data flow are structured to support **collaborative development**, version control, and continuous integration/deployment pipelines (CI/CD).

System Architecture

Frontend (UI)

- React.js / Next.js
- Tailwind CSS
- Upload Files (PDF, Text, Image)

API Gateway (FastAPI)

- Accepts uploads
- Handles REST routes
- Triggers async tasks

Background Worker (Celery + Redis)

- Handles OCR if image or scanned PDF
- Calls summarization model
- Applies NER on summary
- Applies Translation if enabled
- Stores result in DB / File storage

Processing Modules

1. OCR Module
 - Tesseract / Google Vision API
2. Summarizer
 - HuggingFace Transformers (BERT, T5, GPT)
3. NER Module
 - spaCy models to extract entities
4. Translator
 - Google Translate / DeepL APIs

Storage / Database Layer

- PostgreSQL / MongoDB for metadata
- Firebase / AWS S3 for report storage

Output Delivery

- Final summarized report
- Key medical terms highlighted
- Multi-language options (if needed)
- Downloadable file + Web View

Frontend (UI)

React.js / Next.js
Tailwind CSS

- Upload Files (PDF, Text)
Image



API Gateway

(FastAPI)

- Accepts uploads
- Handles REST routes



Background Worker

(Celery + RabbitMQ)

- Handles OCR or if image as scanned PDF
- Applies Translation if enabled
- Stores result if enabled
Stores result in DB / File Stg



Processing Modules

OCR Module

- Tesseract /
Google
Vision API

Summarizer

HuggingFace
Transformers
(BERT, Ts GPT)

NER Module

- spACy
models
- Google
Translate
DeepL APIs

Output Delivery

Final summarized
report translated

4: TECHNOLOGY STACK

To build a robust, scalable, and efficient AI-powered summarization system for medical reports, a modern technology stack is employed across the frontend, backend, AI processing, storage, and deployment layers. Each component has been carefully chosen to balance performance, developer productivity, and ease of integration.

4.1 Frontend Technologies

■ **React.js / Next.js – UI Development**

- **React.js** is a powerful JavaScript library used for building responsive and component-based user interfaces.
- **Next.js** is a React-based framework that enables server-side rendering, static site generation, and better performance out of the box.
- These technologies help in building a **highly interactive, SEO-friendly** frontend for uploading files and visualizing output summaries.

■ **Tailwind CSS – Styling**

- A utility-first CSS framework used for rapid UI development.
 - Allows consistent styling with responsive design features, dark mode support, and mobile compatibility.
 - Speeds up the design process and ensures a clean, professional look across devices.
-

4.2 Backend Technologies

■ **FastAPI / Flask – API Layer**

- **FastAPI** is a modern, high-performance Python web framework built for RESTful APIs, known for its speed and automatic interactive documentation (Swagger).
- **Flask** is a lightweight alternative used when simplicity is preferred.

■ The backend handles routes for:

- ◆ File upload
- ◆ Triggering the summarization pipeline
- ◆ Returning summaries and metadata
- ◆ Optional translation requests

■ **Celery + Redis – *Asynchronous Task Queue***

- **Celery** is used to manage long-running or resource-intensive tasks like OCR, summarization, and translation.
 - **Redis** serves as a fast in-memory message broker to queue and manage tasks between services.
 - Enables non-blocking performance and scalability for handling large files or concurrent requests.
-

4.3 AI/ML Tools

■ **Hugging Face Transformers – *NLP Summarization Models***

Provides access to pre-trained transformer models such as:

- **T5** – Abstractive summarization
- **BERT** – Extractive summarization
- **GPT** – Natural, fluent text generation

These models are used to convert lengthy medical content into short, coherent summaries.

■ **spaCy – *Named Entity Recognition (NER)***

- A fast, industrial-strength NLP library that supports custom and medical-specific entity recognition.
 - Used to extract and highlight terms like diseases, medications, dosages, and lab results from summaries.
-

4.4 OCR Tools

■ **Tesseract OCR – *Open-source Text Extraction***

A widely-used open-source engine that converts scanned image documents into machine-readable text.

Ideal for running in local or on-premise environments where privacy is critical.

■ **Google Vision API – *Cloud-based OCR***

- Offers superior accuracy, especially on handwritten or low-resolution documents.
 - Scales well with enterprise-grade document processing needs.
-

4.5 Storage Technologies

■ **PostgreSQL / MongoDB – *Metadata and Logs***

- **PostgreSQL** (relational DB) for structured data like user logs, file metadata, summary logs, and access records.
- **MongoDB** (NoSQL) is preferred when document-oriented storage is required, especially for nested JSON-like outputs.

■ **Firebase / AWS S3 – *File Storage***

- **Firebase** provides easy-to-integrate cloud storage and real-time updates for frontends.
 - **AWS S3** is used for secure and scalable storage of:
 - Uploaded medical reports
 - Generated summary files (PDFs or text)
 - OCR outputs and metadata
-

4.6 Deployment Infrastructure

● **Docker – *Containerization***

Used to package the application along with all dependencies into lightweight, portable containers.

Ensures consistent environments across development, staging, and production.

- **Kubernetes – *Orchestration***

Manages containerized workloads and services.

Supports auto-scaling, rolling updates, and zero-downtime deployments—ideal for large-scale, real-time applications.

- **Cloud Hosting Platforms:**

AWS EC2 / GCP Compute Engine – For deploying the backend and processing workers on scalable virtual machines.

Vercel / Netlify – For deploying the frontend (especially if built with Next.js), offering global CDN support and CI/CD integration.

5: SYSTEM REQUIREMENTS

To ensure smooth development, deployment, and usage of the AI-powered medical report summarization system, certain system requirements are necessary. These requirements are categorized into **hardware**, **software**, and **environmental dependencies**, based on the role of each component (development, server, and user-end systems).

5.1 Hardware Requirements

For Development / Local Testing

Component	Minimum Requirement	Recommended
CPU	Intel i5 (6th Gen) or AMD equivalent	Intel i7/i9 or AMD Ryzen 7+
RAM	8 GB	16 GB or more
Storage	256 GB SSD	512 GB SSD or more
GPU (Optional)	NVIDIA GTX 1050 (for small model training)	NVIDIA RTX 3060 or higher
Display	1080p Monitor	1080p Dual Monitor Setup

For Production / Cloud Deployment

Component	Minimum Requirement	Scalable Option
CPU	2 vCPUs	Auto-scaled via Kubernetes
RAM	4–8 GB per container	16–32 GB per instance
GPU	Not required (if using API models)	NVIDIA T4/V100 (for custom model inference)
Storage	50 GB block storage (SSD preferred)	AWS EBS / GCP Persistent Disk
Network	100 Mbps	Load-balanced VPC

5.2 Software Requirements

Software	Version / Tooling
Operating System	Windows 10/11, Ubuntu 20.04+, macOS
Python	3.9 or later
Node.js + npm	18.x or later
Docker & Docker Compose	Latest stable
Git	Version control (2.30+)
VS Code / PyCharm	IDE

Backend Dependencies (Python)

- fastapi / flask
- uvicorn (for ASGI server)
- celery, redis
- transformers, torch, spacy, scispacy
- pytesseract, pdf2image
- google-cloud-vision, requests, httpx
- sqlalchemy, psycopg2, pymongo

Frontend Dependencies (JavaScript)

- react or next
- tailwindcss, axios
- formik / react-hook-form
- i18next (for multilingual frontend)

5.3 Environmental Requirements

Service	Usage Purpose
Redis	Broker for Celery async task queue
PostgreSQL	Structured storage (report metadata)
MongoDB	Optional for unstructured summaries
Firebase	File storage and frontend auth
AWS S3	Scalable cloud storage
Google Cloud Vision API	OCR service (alternative to Tesseract)
Google Translate / DeepL	Summary translation

5.4 User-End Requirements

To ensure accessibility and usability for end-users such as patients or doctors:

Requirement	Description
Web Browser	Chrome / Edge / Firefox (latest versions)
Internet Speed	5 Mbps or more recommended
File Types Supported	.pdf, .jpg, .png, .txt
Mobile Compatibility	Fully responsive frontend
Accessibility	Alt-text, ARIA roles, font zoom compatibility

5.5 Optional Tools for CI/CD and Monitoring

Tool	Purpose
GitHub Actions	Continuous Integration
Docker Hub / GHCR	Container registry
Prometheus + Grafana	Monitoring and metrics
Sentry / LogRocket	Frontend/backend error tracking
Nginx / Traefik	Reverse proxy/load balancer

This chapter presents the output generated by the system after successfully processing and summarizing a medical report. It includes example summaries, entity highlights, translations, and downloadable formats. The aim is to demonstrate how effectively the system converts complex input into clear, readable, and informative summaries for patients and healthcare professionals.

6.1 Output Overview

The output of the system consists of:

A **concise medical summary** extracted from long-form reports

Highlighted medical terms (like diseases, medicines, vital signs) using Named Entity Recognition (NER)

Optional multilingual translation

A **downloadable report** in text or PDF format

JSON-formatted response (for integration into other systems)

6.2 Sample Input and Output

Input File:

Type: Scanned PDF

Content (sample):

“Mr. Anil Sharma, age 62, was admitted with high-grade fever and chills. Diagnosed with urinary tract infection (UTI). Started on IV Ceftriaxone. BP was 140/90. Discharged on oral Ciprofloxacin and advised follow-up after 7 days.”

☐ Output Summary:

Summary:
Patient Anil Sharma (62 years old) presented with fever and chills. Diagnosed with Urinary Tract Infection (UTI) and treated with intravenous Ceftriaxone. Discharged with oral Ciprofloxacin and instructed for follow-up after 7 days.

NER Highlighted Terms:

Entity Type	Extracted Terms
Disease	Urinary Tract Infection (UTI)
Medication	Ceftriaxone, Ciprofloxacin
Vital Sign	Blood Pressure: 140/90
Procedure/Action	Intravenous, Follow-up

☐

6.3 Output Format Details

Field	Description
summary	AI-generated short version of the report
entities	Dictionary containing named medical terms
translated_summary	Summary translated into the selected language (optional)
pdf_url	Download link for generated PDF summary
processing_time	Time taken to process the request (for performance tracking)
status	success or error, with optional message

6.4 Output Interpretation and Use

For Patients: Easily understand diagnosis, medication, and next steps without decoding medical jargon.

For Doctors: Save time by reviewing key points quickly instead of reading lengthy reports.

For Admins/Systems: Can integrate structured summaries into dashboards, health apps, or EHRs.

Source code

```
import os
import re
import nltk
import pytesseract
import slate3k as slate
from PIL import Image
from pdf2image import convert_from_path
from nltk.tokenize import sent_tokenize

nltk.download("punkt")

# Optional: specify Tesseract path on Windows
# pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"
```

```

def extractText(file):
    try:
        with open(file, "rb") as pdfFileObj:
            pdfPages = slate.PDF(pdfFileObj)

            return "\n".join(pdfPages)
    except:
        return ""

def extractOCR(file):
    pages = convert_from_path(file, 300)
    text = ""
    for i, page in enumerate(pages):
        filename = f"page_{i+1}.jpg"
        page.save(filename, "JPEG")
        text += pytesseract.image_to_string(Image.open(filename)).replace("-\n", "")
        os.remove(filename)
    return text

def summarize_medical(text, fileName):
    summary_parts = []

    # Normalize text
    text = text.replace('$', '5').replace('na', 'no')

    # --- Patient Name ---
    name_match = re.search(r"(?:Name:|Patient Name)\s*([A-Z][a-z]+(?:\s[A-Z][a-z]+)*)", text)
    if name_match:
        name = name_match.group(1)
        summary_parts.append(f"Patient {name}")

```

```

# --- Admission and Discharge Dates ---
admission_match = re.search(r"Date of Admission\s*\n?([A-Za-z]+\s\d{1,2},\s*\d{4})", text)
discharge_match = re.search(r"Date of Discharge\s*\n?([A-Za-z]+\s\d{1,2},\s*\d{4})", text)
if admission_match and discharge_match:
    summary_parts.append(f"was admitted on {admission_match.group(1)} and discharged on {discharge_match.group(1)}.")

# --- Reason for Admission ---
reason_match = re.search(r"Reason for Admission\s*\n([\s\S]+?)\n\n", text)
if reason_match:
    summary_parts.append("Reasonforadmission:"+reason_match.group(1).strip().replace("\n", " "))

# --- Procedures ---
procedure_match = re.search(r"Procedures Performed\s*\n([\s\S]+?)\n\n", text)
if procedure_match:
    summary_parts.append("Proceduresperformed:"+procedure_match.group(1).strip().replace("\n", " "))

# --- Medications ---
meds_match = re.search(r"Medications Prescribed\s*\n([\s\S]+?)\n\n", text)
if meds_match:
    summary_parts.append("Medicationsprescribed:"+meds_match.group(1).strip().replace("\n", " "))

# --- Condition at Discharge ---
condition_match = re.search(r"Condition at Discharge\s*\n([\s\S]+?)\n\n", text)
if condition_match:
    summary_parts.append("Conditionatdischarge:"+condition_match.group(1).strip().replace("\n", " "))

# --- Follow-up Instructions ---
followup_match = re.search(r"Follow-up Instructions\s*\n([\s\S]+)", text)
if followup_match:
    summary_parts.append("Followupinstructions:"+followup_match.group(1).strip().replace("\n", " "))

```

```

# --- Final Summary ---
summary = " ".join(summary_parts).strip()

if summary:
    with open(fileName + "_MedicalSummary.txt", "w") as summaryText:
        summaryText.write(summary)
    print("\n✓ Summary:\n", summary)
else:
    print("❑ No medical summary could be generated. Check input format or OCR output.")

# ----- Main Program -----
print("Enter the file name (without extension):")
fileName = input().strip()
filePathPDF = fileName + ".pdf"
filePathIMG = fileName + ".jpg"

if not os.path.exists(filePathPDF) and not os.path.exists(filePathIMG):
    print("✗ File not found.")
    exit()

filePath = filePathPDF if os.path.exists(filePathPDF) else filePathIMG
# Step 1: Try direct text extraction
text = extractText(filePath) if filePath.endswith(".pdf") else ""

# Step 2: Fallback to OCR if weak content
needs_ocr = (
    len(text.strip()) < 100 or
    not any(k in text.lower() for k in ["admission", "discharge", "diagnosis", "patient name", "reason for
admission"])
)
if needs_ocr:
    print("❑ Switching to OCR...")
    text = extractOCR(filePath)

summarize_medical(text, fileName)

```

INPUT PDF OR IMAGE

Discharge Summary

Summary	Details
Patient Identification	Name: John Rivers Date of Birth: January 1, 1980 Medical Record Number: 123456
Date of Admission	April 1, 2023
Date of Discharge	May 25, 2023
Reason for Admission	Patient presented to the emergency department with severe chest pain radiating to the left arm, accompanied by shortness of breath and sweating. Initial EKG findings and elevated cardiac enzymes confirmed the diagnosis of Acute Myocardial Infarction (heart attack).
Procedures Performed	Coronary Angiogram revealed a 90% stenosis in the left anterior descending (LAD) artery. Coronary Angioplasty and placement of a drug-eluting stent were successfully performed to reestablish blood flow.
Medications Prescribed	- Aspirin, 81 mg daily (antiplatelet agent for heart health) - Atorvastatin, 40 mg daily (to manage cholesterol levels) - Metoprolol, 50 mg twice daily (beta-blocker to control heart rate and blood pressure) - Nitroglycerin, as needed (for chest pain relief)
Patient's Condition at Discharge	At discharge, the patient's condition was stable. Mild, intermittent chest discomfort was reported but significantly improved compared to admission. The patient was ambulatory and independent in performing activities of daily living. Vital signs were within normal range. Cardiac rhythm was sinus rhythm with no significant ectopy.
Follow-up Care Instructions	Medications: Continue all prescribed medications without interruption unless advised by a doctor. Diet: Follow a heart-healthy diet low in sodium, saturated fats, and simple carbohydrates. Exercise: Start with light walking, gradually increasing to moderate-intensity exercise for at least 30 minutes a day as tolerated. Lifestyle: Smoking cessation advised (if applicable), and stress management techniques recommended. Follow-up: Schedule an appointment with a cardiologist in 2 weeks and with the primary care physician within one week.

OUTPUT

```
[nltk_data] Downloading package punkt to
[nltk_data]   C:\Users\Ifzaan\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Enter the PDF or image file name (without extension):
discharge-summary-example
📄 Switching to OCR...
```

```
[nltk_data] Downloading package punkt to
[nltk_data]   C:\Users\Ifzaan\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Enter the PDF or image file name (without extension):
discharge-summary-example
📄 Switching to OCR...
```

```
✅ Summary:
Patient John Rivers
Date Reason for admission: Procedures Performed Coronary Angiogram revealed a 90% stenosis in the left anterior descending (LAD) artery. Coronary
Angioplasty and placement of a drug-eluting stent were successfully performed to reestablish blood flow. Medications prescribed: At discharge, the
patient's condition was stable. Mild, intermittent chest discomfort was reported but significantly improved compared to admission. The patient was
ambulatory and independent in performing activities of daily living. Vital signs were within normal range. Cardiac rhythm was sinus rhythm with no
significant ectopy. Condition at discharge: : Medications: Continue all prescribed medications without interruption

Process finished with exit code 0
```