🧪 Medical Lab Bot

Medical Lab Bot is an AI-powered web application that interprets medical lab reports from uploaded images. It uses OCR, web scraping, and retrieval-augmented generation (RAG) to provide concise, layman-friendly explanations of medical test results. The web result is stored in Appwrite for catching purposes, so that if the data is already present there is no need to make redundant calls for the same data.

📌 Features

- **Image Upload & OCR**: Upload lab report images (PNG/JPEG) and extract structured tabular data.
- **AI Interpretation**: Uses LLMs to explain lab test results in user-friendly language.
- **Contextual Medical Insights**: Enhances interpretations by retrieving relevant info from medical websites and books using web scraping and vector embeddings.
- **Knowledge Storage**: Stores summarized medical information for repeated and fast access.
- **Modular Scrapers**: Easily extensible to support new medical sources or domains.

⚙️ How It Works

1. **Upload**: User uploads an image of a lab report via the Flask-based web UI.
2. **OCR**: FastAPI endpoint processes the image and extracts tabular data (as LaTeX).
3. **Interpretation**: Another FastAPI endpoint sends this table to LLMs (Groq API) along with the test name and suspected disease.
4. **Contextualization**: The system retrieves relevant context using:
    - ○ Web scraping (via Serper + BeautifulSoup)
    - ○ Vector DB (Pinecone) based on SentenceTransformers

5. **Response**: The final result is a 2-line medical interpretation presented on the web interface.

🧱 System Components

API Endpoints

- **`/extract-lab-report`** (POST)
    - Input: Image (`UploadFile`)
    - Output: LaTeX-formatted table (extracted via OCR)
- **`/chat`** (POST)
    - Input: Test name, suspected disease, and extracted table
    - Output: Layman-level explanation of the lab report using context from web and VDB

🗂️ Directory Structure (Overview)

graphql
CopyEdit

```
Medical-Lab-Bot/
│
├── API/
│   ├── app.py                 # Flask server for image upload +
results
│   ├── chatBot_final.py       # FastAPI for OCR + chatbot
interpretation
│   ├── functions.py           # Core logic for interpretation,
web/VDB search
│   ├── database.py            # Appwrite-based data store
│   ├── templates/             # HTML frontend
│   └── uploads/               # Uploaded lab images
│
├── Scrapper/
│   ├── scrapper.py
│   ├── testingLab_scrapper.py
│   └── crawl4ai_Scrapper.ipynb
│
├── Test_Files/
│   └── *.ipynb, *.py          # Test and experimentation scripts
```

🔌 Technologies Used

- **Backend**: Python, FastAPI, Flask
- **LLMs**: Groq API (LLaMA variants)
- **Vector DB**: Pinecone
- **Embeddings**: `msmarco-bert-base-dot-v5` (Sentence Transformers)
- **Web Scraping**: Serper API + BeautifulSoup
- **Prompt Chaining**: LangChain (ChatGroq, StrOutputParser)
- **Validation**: Pydantic
- **Database**: Appwrite (summarized test storage)

🚀 Getting Started

Prerequisites

- Python 3.9+
- API Keys for:
    - Groq
    - Serper
    - Pinecone
    - Appwrite

`.env` file with:

```
GROQ_API_KEY=your_key
GROQ_API_KEY_2=your_backup_key
```

```
SERPER_API_KEY=your_key
PINECONE_API_KEY=your_key

APPWRITE_API_KEY=your_key
PROJECT_ID=your_key
DATABASE_ID=your_key
END_POINT=your_key
```

Installation & Run

```
# Clone the repo
git clone https://github.com/your-repo/medical-lab-bot
cd medical-lab-bot

# Install dependencies
pip install -r requirements.txt

# Start Flask app (web UI)
python API/app.py

# Start FastAPI backend
uvicorn API.chatBot_final:app --reload --port 8001
```

Access the App

- Web Interface: `http://localhost:5000`
- FastAPI Docs: `http://localhost:8001/docs`

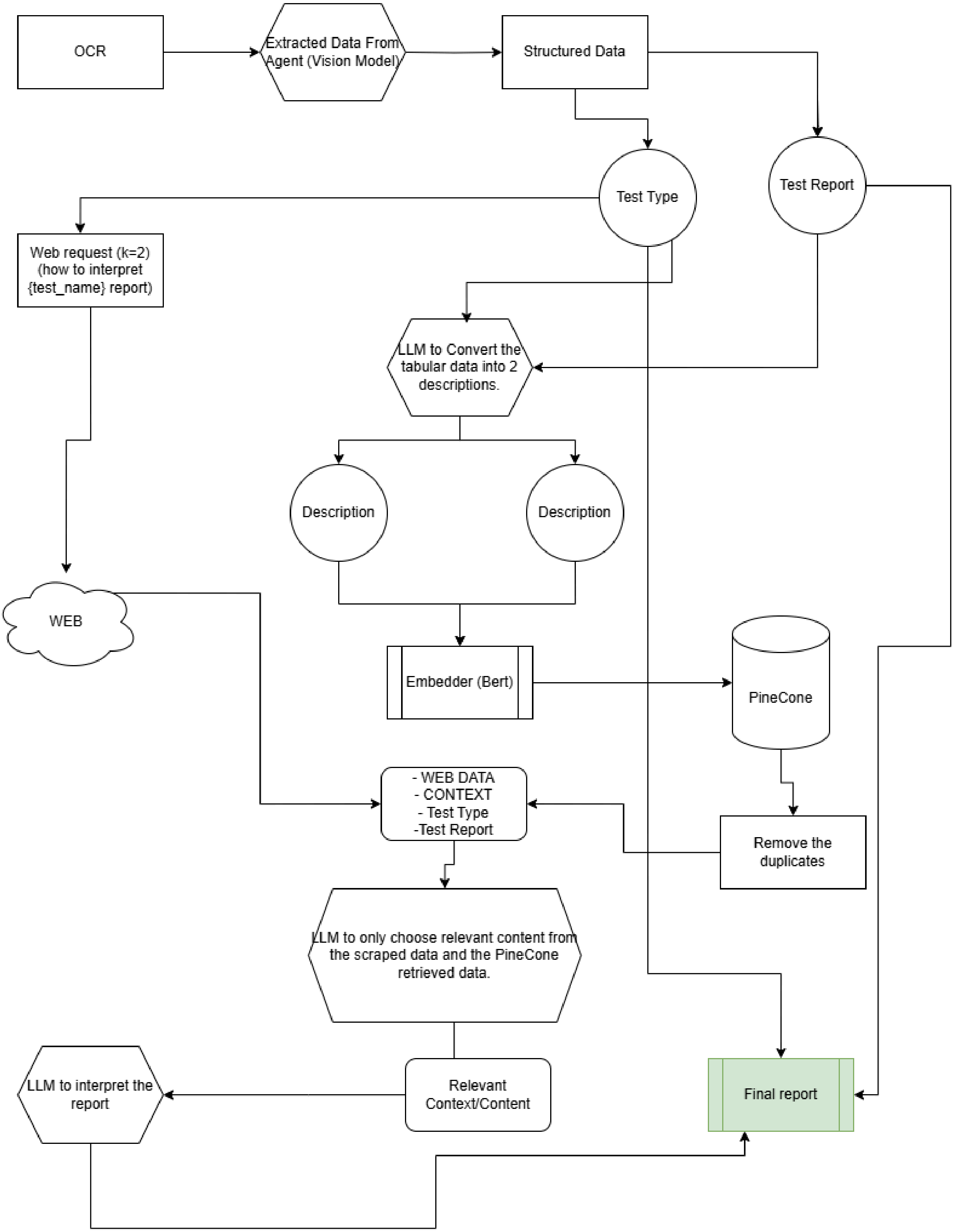
📈 Example Flow

1. Upload a CBC lab report image.
2. Image is converted into LaTeX tabular format.
3. System recognizes “Complete Blood Count” as test type.

4. Groq LLM generates two-line interpretation using:
    - Summarized content from Testing.com, MedlinePlus
    - Retrieved book/web context via Pinecone + web scraped data in Real-time (Serper api) + LangChain
5. User receives human-readable insight like:

    "Your white blood cell count is elevated, which could indicate a bacterial infection. Further consultation is recommended."

Flow-Diagram:


OCR
Extracted Data From Agent (Vision Model)
Structured Data
Test Type
Test Report
Web request (k=2) (how to interpret {test_name} report)
LLM to Convert the tabular data into 2 descriptions.
Description
Description
WEB
Embedder (Bert)
PineCone
- WEB DATA
- CONTEXT
- Test Type
-Test Report
Remove the duplicates
LLM to only choose relevant content from the scraped data and the PineCone retrieved data.
LLM to interpret the report
Relevant Context/Content
Final report

⚠️ Disclaimer

This application is intended for informational and educational purposes only. It does **not replace professional medical advice**. Always consult a licensed healthcare provider.