MediScan

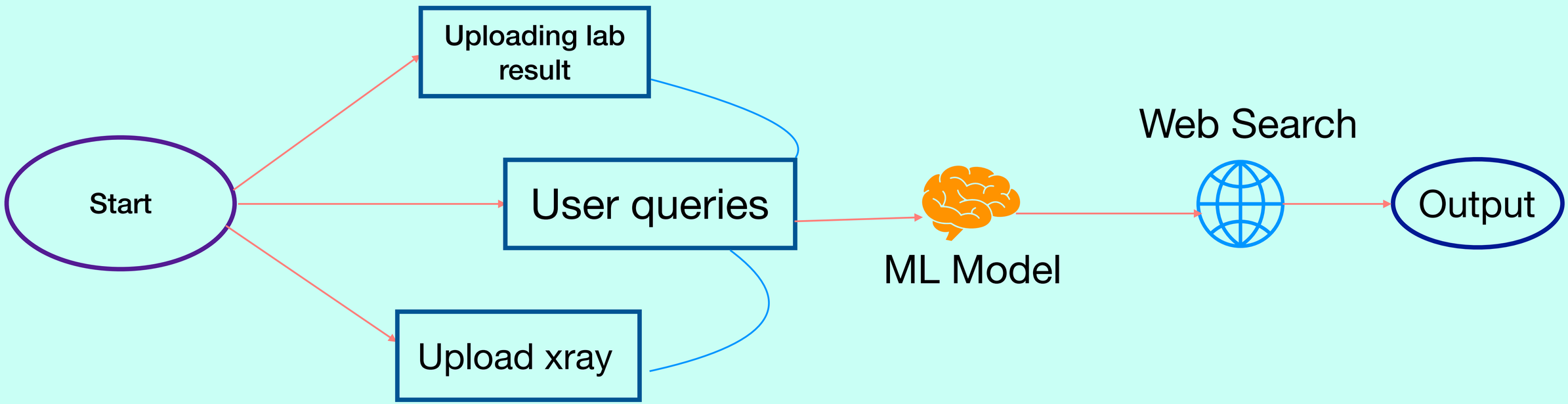
AI Diagnostic Chatbot

Decaf Dictators

Problem Statement

A significant challenge in healthcare today is the reliability and timeliness of test reports from radiologists. Many reports may lack accuracy due to the commercialisation of the medical industry, and even accurate reports can be delayed. Patients often have no trustworthy way to verify their results or receive guidance on potential illnesses. To address this, propose a CHATBOT that analyses test samples (like blood tests or X-rays) and generates a comprehensive report. This chatbot will not only verify the findings but also provide actionable suggestions for further steps regarding the patient's health.

Approach


Uploading lab result
Start
User queries
Upload xray
ML Model
Web Search
Output

Proposed Solution

1. Key Components

Input Handling:

- Blood Test: Upload and extract text from PDFs.
- X-rays: Upload and preprocess image files.

AI Models:

- **Text Analysis**: Use GPT/bioBert models to analyse blood test reports and provide diagnostic insights.
- **Image Analysis**: Use pre-trained models (e.g., DenseNet121) for X-ray classification and visualisation.

Chatbot Interface:

- Enable users to upload files, ask health-related queries, and receive actionable insight

2. Implementation Steps

Blood Test Analysis:

- Extract text using PyPDF2.
- Use GPT to analyse and provide results with recommendations.

X-ray Analysis:

- Preprocess images using OpenCV/Pillow.
- Use a pre-trained deep learning model for diagnosis.

Chatbot Integration:

- Build a conversational flow using Flask and integrate analysis pipelines.
- Offer clear, actionable health suggestions in user-friendly language.

Proposed Solution

3. Output

- Diagnostic reports with confidence scores.
- Actionable steps, such as lifestyle changes, further tests, or specialist consultations.

4. User Flow

1. User selects analysis type (Blood Test/X-ray).
2. Uploads the file.
3. Chatbot analyzes data and presents findings.
4. User receives a diagnostic report with recommendations.

Technical Approach

Programming Languages

- **Python**: For building the AI models, data processing, and backend integration.
- **HTML/CSS/JS**: For creating the frontend of the web application.

Frameworks and Libraries

- **PyTorch/TensorFlow**: For building and fine-tuning AI models for both text and image analysis.
- **Hugging Face Transformers**: For text-based diagnostic analysis (e.g., GPT models).
- **Flask**: For creating the backend and REST API endpoints.
- **OpenCV/Pillow**: For handling and preprocessing X-ray images.
- **PyPDF2**: For extracting text from PDF blood test reports.