# DOCTORY AI

A Multi-Modal AI System for Preliminary Disease Diagnosis

**Digital Egypt Pioneers Initiative (DEPI)**

**Graduation Project**

**Documentation**

# Submitted to DEPI Program

December 8, 2025

# Contents

# 1 Project Planning

## 1.1 Project Motivation

Access to fast, reliable, and understandable medical guidance is limited for a large number of people, especially in remote areas or regions with weak healthcare infrastructure.

Most AI-based diagnostic tools currently available provide raw technical outputs (such as probability scores or heatmaps) that are difficult for the average patient to interpret.

This project aims to bridge that gap by combining Computer Vision, Machine Learning, and Large Language Models (LLMs) to create an intelligent, human-centered medical assistant that acts as a bridge between complex medical data and the patient.

## 1.2 Project Scope

**DOCTORY AI** is a unified AI-driven medical diagnostic platform designed to assist users in identifying potential diseases through various input methods, including medical images, clinical measurements, and textual symptom descriptions.

The scope specifically covers four critical medical domains:

1. **Pneumonia Detection:** Using chest X-ray images to identify

lung opacity.

2. **Malaria Detection:** Analyzing blood smear microscopy images for parasitic infection.

3. **Diabetes Risk Prediction:** Using tabular diagnostic values (glucose, BMI, insulin levels).

4. **Heart Disease Risk Analysis:** Evaluating lifestyle and medical indicators (age, cholesterol, blood pressure).

It is important to note that the system output is always a **preliminary** result and is never intended to replace professional medical advice or doctors.

## 1.3 Project Objectives

The primary objectives of this project are:

- Build accurate AI diagnostic models for the four specified diseases.

- Provide an intuitive, accessible, and simple user interface for non-technical users.

- Integrate Large Language Models to translate technical predictions into empathetic, human-friendly medical explanations.

- Ensure data privacy by following a strictly stateless system design.

- Deploy the system on a cloud platform (Streamlit Cloud) for easy accessibility.

## 1.4 Development Methodology

The project follows the **Agile Development** methodology due to its flexibility and the need for iterative validation of AI models:

- **Sprint 1:** Requirement analysis, dataset acquisition, and feasibility study.
- **Sprint 2:** Model development, training, validation, and hyperparameter optimization.
- **Sprint 3:** Backend logic development and integration of ONNX Runtime for efficient inference.
- **Sprint 4:** Frontend implementation using Streamlit, UI design, and LLM API connection.
- **Sprint 5:** System testing, deployment on Streamlit Cloud, and final documentation.

## 1.5 Tools & Technologies

The following technologies were selected to ensure performance and scalability:

| **Category** | **Technologies Used** |
|---|---|
| Programming Language | Python |
| Web Framework | Streamlit |
| ML Frameworks | TensorFlow, PyTorch, Scikit-learn |
| Computer Vision | Ultralytics YOLOv11, VGG16 |
| LLM Integration | API-based LLM Service |
| Deployment | Streamlit Cloud |

Table 1: Technology Stack

## 1.6 Team Roles & Responsibilities

The development of DOCTORY AI is executed by a dedicated team. Each member holds specific responsibilities to ensure the successful delivery of the project components.

| Team Member | Key Responsibilities |
|---|---|
| **Jasmine** | Data acquisition, Developing Diabetes Model & System Integration |
| **Basmala** | UI/UX Implementation, Documentation, & Presentation |
| **Arwa** | Team Leadership, LLM Integration & Cloud Deployment |
| **Aya** | Developing Pneumonia Detection Model & Documentation |
| **Esraa** | Developing Malaria Detection Model & Documentation |
| **Haneen** | Developing Heart Disease Risk Model & Documentation |

Table 2: Team Members and Assigned Roles

# 2 Stakeholder Analysis

Stakeholders are individuals, groups, or entities that can affect or are affected by the system's objectives. Understanding their specific interests ensures better system alignment and user satisfaction.

| **Stakeholder** | **Role & Interest** | **Influence** |
|---|---|---|
| End Users | Individuals seeking fast, confidential, and reliable preliminary medical information. | High |
| Development Team | The group responsible for system design, model training, and integration. | High |
| Academic Evaluators | Assessors judging the technical quality, documentation, and innovation level. | High |
| Healthcare Professionals | Doctors providing indirect validation and future clinical input or oversight. | Medium |

Table 3: Stakeholder Matrix

# 3 System Architecture and Data Handling

## 3.1 Overall Architecture

The system utilizes **Streamlit**, which unifies the frontend and backend into a single, cohesive application structure. This simplifies data flow and deployment while maintaining high performance. The interaction between the User, the Application Logic, the AI Inference Engine, and the External LLM Service is illustrated in Figure 1.

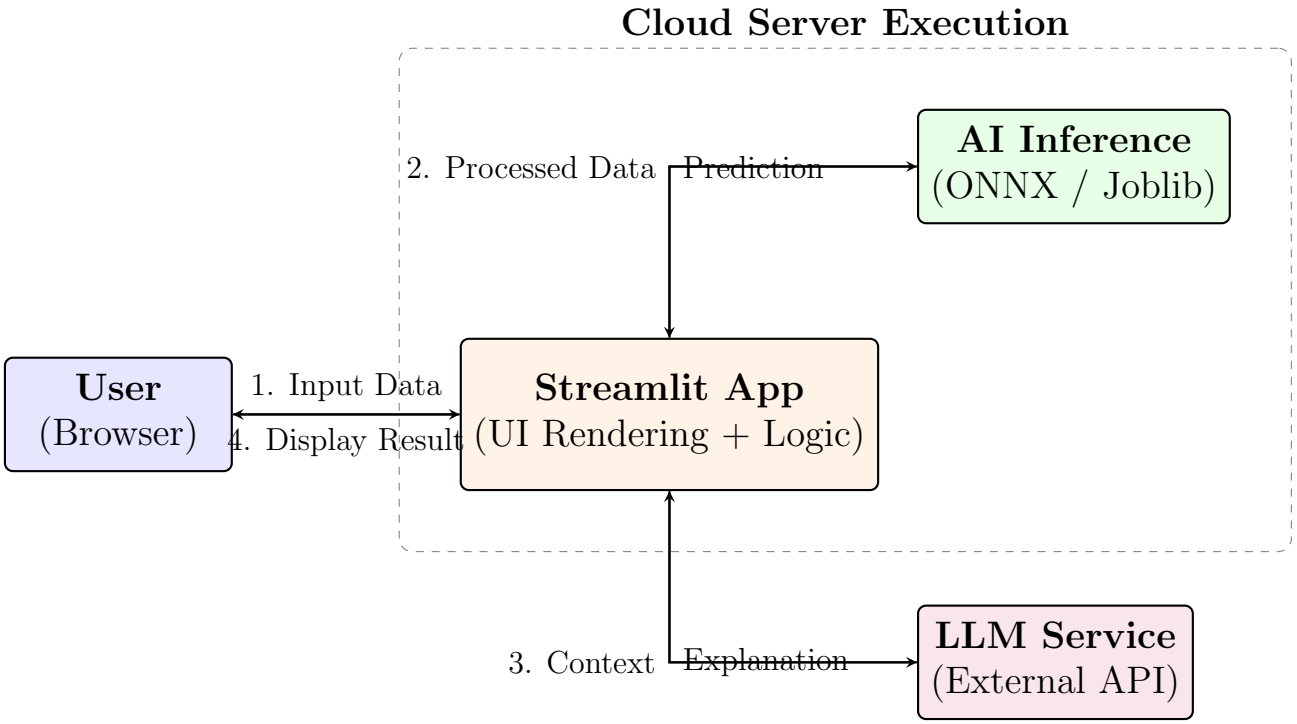


Figure 1: System Architecture showing the centralized role of the Streamlit Application.

The architecture consists of three main components:

- **Streamlit Application:** Acts as both the presentation layer (UI) and the controller. It manages user sessions, processes inputs, and calls the appropriate AI models.

- **Inference Engine:** Optimized model artifacts (ONNX/Joblib) that run directly within the application environment for low-latency predictions.

- **LLM Integration:** An external service called via API to generate natural language explanations based on the model's numerical output.

## 3.2 Stateless Design

DOCTORY AI is implemented using a **no-persistence approach**. The system does not store any sensitive user data, medical images, or medical history in a database. Data exists only in the volatile memory (RAM) during the user's active session and is discarded immediately after the result is displayed.

## 3.3 Data Flow

The data lifecycle in the application follows these steps:

1. The user accesses the web app and selects a diagnostic tool via the sidebar.

2. Input data is uploaded (image) or entered (form) and sent to the server script.

3. The Python logic preprocesses the data and runs inference using the loaded model.

4. The prediction is sent to the LLM to generate a user-friendly explanation.

5. The final result and advice are rendered immediately on the user's screen.

# 4 System Requirements

This section outlines the specific behaviors and functions of the system (Functional Requirements) as well as the criteria that can be used to judge the operation of the system (Non-Functional Requirements).

## 4.1 Functional Requirements

Functional requirements define what the system must do to satisfy user needs.

1. **Diagnostic Inputs:**
    - The system must allow users to upload image files (JPG, PNG) for Pneumonia and Malaria detection.
    - The system must provide input fields for numerical clinical data (e.g., Glucose level, BMI, Age) for Diabetes and Heart Disease prediction.
    - The system must validate user inputs to ensure they fall within realistic medical ranges before processing.
2. **AI Inference Processing:**
    - The system must process the input data using the appropriate pre-trained model (YOLOv11, VGG16, or Scikit-learn models).

- The system must generate a classification result (e.g., "Normal" vs "Pneumonia") along with a confidence score.

3. **LLM Explanation Generation:**

   - The system must send the classification result and context to the LLM API.

   - The system must display a natural language explanation of the diagnosis in English (or the user's preferred language if implemented).

4. **User Interface Operations:**

   - The system must allow users to navigate between different disease modules via a sidebar.

   - The system must display a loading indicator while inference is being performed.

   - The system must allow users to reset the form to perform a new diagnosis.

## 4.2 Non-Functional Requirements

Non-functional requirements specify criteria that act as constraints on the design and operation of the system.

1. **Performance Latency:**

- The system should generate a diagnostic result within 5 seconds for tabular data and within 10 seconds for image analysis.

- The application interface should load within 3 seconds on standard broadband connections.

2. **Scalability:**

   - The system architecture (Streamlit + Cloud) must support multiple concurrent users without crashing.

   - The backend must be capable of handling simultaneous API requests to the LLM service.

3. **Usability:**

   - The user interface must be responsive and accessible on both desktop and tablet devices.

   - The design must follow accessibility standards (e.g., high contrast, clear fonts) to accommodate elderly or visually impaired users.

4. **Reliability Availability:**

   - The system should be available 99% of the time during operational hours.

   - In case of AI model failure or API timeout, the system must display a user-friendly error message instead of crashing.

5. **Security Privacy:**

    - **Statelessness:** The system must NOT store any user-uploaded images or personal health data after the session ends.

    - All data transmission between the client and the server must be encrypted using HTTPS.

# 5 UI/UX Design

## 5.1 Design Philosophy

The interface follows a "Patient-First" philosophy, utilizing Streamlit's clean and modern component library. It prioritizes simplicity, high accessibility, and emotional comfort.

## 5.2 User Interaction

The typical user flow is designed to be linear and friction-free:

1. **Navigation:** Users use a clear sidebar to switch between different disease modules.

2. **Input:** Simple widgets (File Uploaders, Sliders, Number Inputs) guide the data entry process.

3. **Feedback:** Loading spinners indicate processing status to keep the user informed.

4. **Result:** Results are displayed using clear metrics and expandable text boxes for detailed explanations.

## 5.3 Conceptual Wireframes

# Application Layout

**Structure Description:**

- **Sidebar (Left Panel):**
  - Application Logo.
  - Navigation Menu (Pneumonia, Malaria, Diabetes, Heart).
  - "About Us" section.
- **Main Area (Center):**
  - Title and Brief Instruction.
  - Input Zone (Image Upload or Data Form).
  - "Analyze" Action Button.
- **Output Zone (Bottom):**
  - Prediction Result (e.g., "Healthy").
  - Confidence Score Bar.
  - AI Assistant Message (LLM Explanation).

*The layout leverages the standard Streamlit single-page application structure for consistency and ease of use.*

Figure 2: Streamlit Application Layout Concept

# 6 Risk Analysis

Identifying potential risks is crucial for deployment:

- **Model Bias:** Potential for bias if the training dataset is unbalanced regarding age, gender, or ethnicity.
- **Over-reliance:** Users might treat the AI suggestion as a final medical verdict, ignoring professional advice.
- **Service Interruptions:** Dependency on third-party APIs (like LLM providers) could lead to downtime.
- **False Results:** The risk of false negatives (telling a sick patient they are healthy) is a critical concern addressed by setting high sensitivity thresholds.

# 7 Ethical Considerations

- **Transparency:** The system clearly states it is a preliminary diagnostic tool on every screen.
- **Privacy:** No personal data is stored, preserving user anonymity.
- **Explainability:** We prioritize providing reasons for the AI's decision rather than a "black box" output.

- **Medical Disclaimer:** Strong encouragement is provided to consult professional doctors for confirmation.