# AI-Powered Medical Report Simplifier (Enhanced Version)

This document provides a complete guide to the enhanced version of the Medical Report Simplifier. This version has been refactored for clarity, robustness, and professionalism, directly addressing advanced evaluation criteria.

## Summary of Enhancements

* **Professional Code Structure:** The project is no longer a single file. Logic is separated into services (ocr\_service, normalization\_service, summary\_service), making the code clean, reusable, and easy to maintain. app.py is now a lean API router.
* **Improved OCR Accuracy:** The system now uses **OpenCV** to automatically pre-process images (grayscale, contrast enhancement) before sending them to Tesseract, increasing the accuracy of text extraction from real-world scans.
* **Advanced Guardrails:** A **physiological sanity check** has been added. The system now validates extracted values against known biological limits to catch and discard major OCR errors (e.g., a Hemoglobin value of "102").
* **Standardized Error Responses:** All API errors now return a consistent JSON schema ({"status": "error", "message": "..."}), which is a best practice for building reliable APIs.

## New Project Structure

Plum-assignment/  
├── services/ # NEW: All business logic is here  
│ ├── \_\_init\_\_.py # Makes 'services' a Python package  
│ ├── ocr\_service.py # Handles image processing and Tesseract OCR  
│ ├── normalization\_service.py # Handles test normalization and validation  
│ └── summary\_service.py # Handles generating patient-friendly text  
├── app.py # The main Flask application (now much smaller)  
├── requirements.txt # Updated list of dependencies  
└── README\_ENHANCED.md # This guide

## Setup and Running

The setup process is very similar to before, with one new dependency.

### 1. Prerequisites

* **Python 3.8+**
* **Tesseract OCR Engine**: See original guide for installation.

### 2. Installation

1. Organize your files according to the new project structure above. Create a services folder and place the service files inside it. Create an empty \_\_init\_\_.py file inside the services folder as well.
2. Open a terminal in the Plum-assignment root folder.
3. Create and activate a Python virtual environment:  
   python -m venv venv  
   source venv/Scripts/activate
4. Install the updated dependencies:  
   pip install -r requirements.txt

### 3. Running the Application

With your virtual environment active, run the main app:

python app.py

The server will start on http://127.0.0.1:5000.

## Testing

You can use Postman to test the /simplify endpoint exactly as before. The API's contract (how you call it and the successful response format) has not changed, but its internal logic and robustness are now significantly improved.