Early-Stage Disease Diagnosis System

# 1. Project Overview

A Flask-based web application that classifies human nail images using a pre-trained CNN (VGG‑16). Users upload a nail image, and the app displays a predicted disease label from a trained model.

# 2. Repository Structure

/ (root)  
├─ dataset/ # Image data for training and testing  
├─ static/ # CSS, JS, images for frontend  
├─ templates/ # HTML templates (e.g., upload.html, result.html)  
├─ uploads/ # Runtime folder: stores user-uploaded files  
├─ Model.ipynb # Training notebook using Keras and VGG‑16  
├─ notebook.ipynb # Another notebook with model exploration  
├─ vgg16\_deep\_learning.py# Python script training/fine-tuning the model  
├─ app.py # Flask application entry point  
├─ \*.h5 # Trained Keras models (e.g., VGG16-nail-disease.h5, best\_model.h5)  
├─ class\_labels.json # Mapping from model output indices to disease names  
├─ code.txt # Additional model/code documentation  
└─ README.md # Repository overview

# 3. Prerequisites

- Python ≥ 3.7  
- TensorFlow / Keras  
- Flask  
- OpenCV, Pillow, NumPy  
- Jupyter (optional, for .ipynb notebooks)

# 4. Installation

1. Clone the repo:

git clone https://github.com/shubham0328/early-stage-disease-diagnosis-system-project

cd early-stage-disease-diagnosis-system-project

2. Install dependencies:

pip install -r requirements.txt

(If requirements.txt is missing, manually install: flask, tensorflow, keras, opencv-python, pillow, numpy.)

# 5. Model Training

- Open Model.ipynb or notebook.ipynb in Jupyter to retrain or fine-tune the VGG‑16 model.  
- Alternatively, run:  
 python vgg16\_deep\_learning.py  
- Saved model files (.h5) can be used for inference.

# 6. Class Labels

The class\_labels.json file contains a mapping like:

{"0": "Healthy", "1": "Anemia", "2": "Diabetes", ...}

# 7. Flask Application

In app.py:  
- Loads the .h5 model and class\_labels.json.  
- Exposes / route with an upload form.  
- Processes uploaded images (resize, normalize).  
- Returns prediction and renders a results page.

Example Run:

python app.py

Then visit: http://127.0.0.1:5000

# 8. Usage Workflow

1. Launch app (python app.py).  
2. Upload a nail image via the web form.  
3. View the disease prediction or “Healthy” result.

# 9. Dataset

Include your own dataset of nail images categorized by disease condition. Ensure structure like:  
dataset/  
 ├─ train/  
 │ ├─ Healthy/  
 │ ├─ Anemia/  
 │ └─ ...  
 └─ test/  
 ├─ Healthy/  
 ├─ Anemia/  
 └─ ...

# 10. Future Enhancements

- Improved data augmentation and regularization.  
- Support for multiple model architectures.  
- Grad‑CAM explainability visualization.  
- User history logging and admin interface.  
- Docker deployment & cloud hosting.

# 11. Troubleshooting

- Model load errors: Check file paths in app.py.  
- Image size issues: Ensure image resizing matches training config.  
- Dependency issues: Verify installed library versions, or recreate virtual environment.

# 12. References

This project builds upon standard CNN transfer‑learning practices using VGG‑16, TensorFlow/Keras image classification, and Flask for deployment.