Jaundice Detection System – Full Documentation

📌 1. Project Overview

This project is a web-based jaundice detection system that uses an ESP32-CAM module to stream live video. The user captures a face image, and the backend performs image processing to analyze jaundice levels using color metrics (Hue, Saturation, Value - HSV). The results are visually displayed in the UI.

---

2. System Architecture

➤ Frontend (Flask Web Interface)

- Live video feed
- Capture photo with a given filename
- Show processed face (ROI), yellow mask, and HSV results

➤ Backend (Flask Server)

- Streams live feed from ESP32-CAM
- Captures and saves frames
- Applies computer vision (OpenCV) to:
    - Detect face
    - Extract ROI (under the eyes)
    - Create a yellow mask
    - Calculate mean HSV values

➤ Hardware

- ESP32-CAM (serving video over HTTP stream)

---

3. Features

- 📸 Live video feed from ESP32-CAM
- 📝 Enter filename for image capture
- 🧠 Automatic face + ROI detection
- 🎨 Yellow region mask visualization
- 📡 HSV color analysis (mean values)

- Hue bar with a cursor indicating value

---

4. Technology Stack

Component	Technology
Frontend	HTML, Bootstrap 5
Backend	Python, Flask
Image Processing	OpenCV
Hardware	ESP32-CAM

---

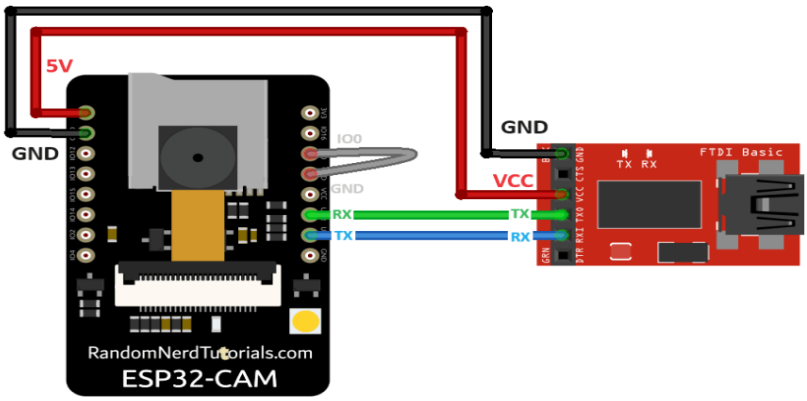
5. Setup Instructions

Hardware Setup

Components:

- ESP32-CAM
- FTDI USB-to-Serial Adapter

Connection:



- Connect **GND → GND**
- **IO0 → GND** (for flashing)
- **TX → RX**
- **RX → TX**
- **5V → VCC**

Flashing:

1. Upload the code (`esp32-cam.ino`) using Arduino IDE.
2. After uploading, **remove the wire between GND and IO0**.
3. Reconnect ESP32-CAM to power to boot into normal mode.

Software Setup

From python ide

1. Run the following command in your virtual environment:

   ```
   pip install -r requirements.txt
   ```

2. Connect your device (laptop or phone) to ESP32's Access Point:
   - SSID: `ESP32-CAM`
   - Password: `12345678`
3. Run the server:

   ```
   python server.py
   ```

4. Open your browser and navigate to:

   http://192.168.4.2:5000/

Or From app.exe:

- `Just run the app.exe`

Now you can see the video stream and interact with the app.

---

6. API Endpoints

Endpoint	Method	Description
`/`	GET	Homepage with video feed
`/video_feed`	GET	Streams MJPEG video from ESP32-CAM
`/trigger-capture`	POST	Captures image from stream
`/check-result`	GET	Polls for processed result JSON

Endpoint	Method	Description
`/get-image/<path>`	GET	Serves processed images (ROI/mask)

---

7. File Structure

```
project/
│
├── static/
│   ├── css/
│   │   └── bootstrap.min.css
│   └── js/
│       └── bootstrap.bundle.min.js
│
├── templates/
│   └── index.html
│
├── server.py             # Main Flask app
├── requirements.txt
```

---

8. Example Workflow

1. User sees live ESP32-CAM stream.
2. Enters a filename and clicks **Take Photo**.
3. Flask captures a frame and processes it:
    - Face detection
    - ROI extraction
    - Yellow mask creation
    - HSV value computation
4. Results are shown:
    - Image with face + ROI
    - Yellow mask image
    - Hue bar with a pointer
    - Textual HSV values