
🌱 COMPLETE PROJECT — REMAINING WORK (STEP-BY-STEP)

Think of it like 4 big phases:

```
1️⃣ Prepare Data
2️⃣ Train Models
3️⃣ Connect Model to Backend
4️⃣ Final Polish & Testing
```

I'll explain each deeply but simply.

🧩 STEP 1 — Prepare Dataset (VERY FIRST THING)

Your system is multimodal, meaning it uses:

```
Image + Text + Vitals
```

So you must prepare inputs for training.

🩻 1.1 Download Image Dataset

Use:

👉 Chest X-ray Pneumonia Dataset (Kaggle)

Folder should look like:

```
dataset/
   train/
       NORMAL/
       PNEUMONIA/
   test/
       NORMAL/
       PNEUMONIA/
```

Why this dataset?

- Already labeled

- Easy for beginners
- Good for demo AI

📝 1.2 Create Text Data

Your dataset doesn't contain clinical notes.

So you create synthetic examples like:

```
"persistent cough and fever"
"shortness of breath"
"mild chest pain"
```

Save in CSV:

`text_data.csv`

Columns:

`image_name, clinical_notes`

📊 1.3 Prepare Vitals Data

Vitals are just numbers.

Example format:

`image_name, spo2, temperature, heart_rate`

You can simulate values.

Reason:
Vitals improve "multimodal" explanation during demo.

🤖 STEP 2 — Train Image Model

Now you teach AI to understand X-rays.

Create file:

`train_image_model.py`

Use PyTorch.

2.1 Load Pretrained Model

Use:

`ResNet18`

Why?

- Lightweight
- Easy to train
- Enough for project

2.2 Modify Last Layer

Change output to:

`2 classes → Normal / Pneumonia`

2.3 Train Model

Basic training loop:

```
for epoch:
   forward pass
   calculate loss
   backpropagation
```

Train few epochs (3–5 is enough).

2.4 Save Model

```
torch.save(model.state_dict(), "image_model.pth")
```

Now you have image intelligence.

🧠 STEP 3 — Process Text Data

Create file:

`text_encoder.py`

Use:

`sentence-transformers`

Model:

`all-MiniLM-L6-v2`

What happens:

```
clinical notes → vector embedding
```

Example:

```
"fever and cough" → [0.12, -0.33, 0.91 ...]
```

This makes text usable for AI.

🔢 STEP 4 — Prepare Vitals Features

Vitals must be numbers.

Convert:

```
SpO2 → divide by 100
Temp → normalize
Heart rate → normalize
```

Now vitals become small numeric vector.

🔗 STEP 5 — Build Multimodal Model (MOST IMPORTANT AI PART)

Create file:

`multimodal_model.py`

Structure:

```
image_features = CNN(xray)
text_features  = SBERT(notes)
vitals_features = numeric values
```

```
combined = concat(image_features, text_features, vitals_features)

prediction = LinearLayer(combined)
```

This is called **feature fusion**.

You are not building a giant transformer — just a smart combined model.

🏋️ STEP 6 — Train Multimodal Model

Training flow:

```
Load image
Load text embedding
Load vitals
Combine
Predict label
Update weights
```

Train few epochs.

Save model:

```
multimodal_model.pth
```

⚙️ STEP 7 — Connect Model to Backend (VERY EASY NOW)

Open:

```
backend/main.py
```

At startup:

```
load multimodal_model.pth
```

Inside `/upload` endpoint:

Replace placeholder AI with:

```
prediction = model(xray, clinical_notes, vitals)
```

Return:

```
{
```

```
    label: "Pneumonia",
    confidence: "82%"
}
```

Doctor dashboard will automatically update because it already reads backend data.

You DO NOT need to change frontend again.

👨‍⚕️ STEP 8 — Improve Doctor Dashboard (Small Changes)

Add:

- Severity color (high/medium/low)
- Timestamp
- Better patient name display

No heavy coding — just UI polish.

🧪 STEP 9 — Test Full Flow

Test like real user:

```
Patient uploads image
Backend runs model
Doctor dashboard shows prediction
```

If this works — your project is technically COMPLETE.

📄 STEP 10 — Documentation & Submission Work

Finish:

- SRS update
- Architecture diagram
- Flow diagram
- README
- Demo explanation

⭐ FINAL SYSTEM YOU WILL HAVE

```
React Frontend
       ↓
FastAPI Backend
       ↓
Multimodal AI Model
       ↓
Doctor Dashboard Results
```

That is a FULL AI medical platform.

🤍 Honest Truth (So You Don’t Feel Overwhelmed)

You already completed

```
Frontend + Backend + Integration
```

Now you only need to:

```
Add intelligence (model training)
```

The rest is polishing.
