■ LungAI

Intelligent Lung Disease Detection System

Comprehensive Project Documentation

Version	1.0
Generated	25 February 2026, 06:29 PM
Project	Final Year AI Engineering Project
Tech Stack	Python · Flask · TensorFlow · MobileNetV2 · React
Platform	Web Application (localhost:5001)

■ For Educational & Engineering Demonstration Purposes Only

1. Project Overview

LungAI is a Final Year AI Engineering Project that provides automated lung disease detection from chest X-ray images using deep learning. The system classifies X-ray images into four categories in real time and provides clinical insights, Grad-CAM explainability heatmaps, IoT-integrated vital signs, patient management, and downloadable PDF reports — all through a modern web interface.

Key Capabilities:

✔ Automated 4-class chest X-ray classification (COVID-19, Pneumonia, Tuberculosis, Normal)

✔ Grad-CAM heatmap overlay for AI explainability (shows WHERE the model is looking)

✔ Severity scoring: Mild / Moderate / Severe based on confidence %

✔ Clinical risk badge: Low / Moderate / Severe / Critical Risk

✔ AI-generated medical insights, diet recommendations & medication guidance

✔ IoT-simulated real-time vitals: Heart Rate, SpO■, Temperature, Respiratory Rate

✔ PACS History Timeline — view all past analyses per patient

✔ Downloadable PDF clinical report per analysis

✔ Doctor Portal & Patient Portal with login/role-based access

✔ Google OAuth authentication support

2. Training Dataset

The model was trained on a curated multi-class chest X-ray dataset organised into four disease categories. The dataset follows a standard train / val / test split stored under **data/chest_xray_multi/**.

Dataset Structure:

Split	COVID-19	Normal	Pneumonia	Tuberculosis	Total (approx.)
Train	~1,800	~1,800	~3,900	~700	~8,200 images
Val	~200	~200	~400	~80	~880 images
Test	~200	~234	~390	~80	~900 images
TOTAL	**~2,200**	**~2,234**	**~4,690**	**~860**	**≈ 9,980 images**

Note: Exact counts depend on the dataset version used. Class imbalance (Pneumonia >> Tuberculosis) is handled automatically using computed class weights during training.

Data Augmentation Applied During Training:

- Rotation: ±10° — simulates patient positioning variation
- Width & Height Shift: ±10% — accounts for off-centre X-rays
- Shear Range: 0.1 — slight distortion tolerance
- Zoom Range: ±20% — scale invariance
- Brightness Range: [0.8 – 1.2] — exposure variation

- Horizontal Flip: True — bilateral symmetry of lungs
- Fill Mode: nearest — border fill after transformation

All images are resized to **224 × 224 pixels** and normalised using MobileNetV2's preprocessing function (scales pixel values to the range **[-1, 1]**).

3. Algorithms & Model Architecture

3.1 Primary Algorithm — Transfer Learning with MobileNetV2

The core classification algorithm is **Transfer Learning** using **MobileNetV2** (pre-trained on ImageNet). MobileNetV2 was chosen for its balance of accuracy and computational efficiency — making it deployable on standard hardware without a GPU.

Layer / Component	Details
Base Model	MobileNetV2 (ImageNet weights, include_top=False)
Input Shape	224 × 224 × 3 (RGB)
GlobalAveragePooling2D	Reduces spatial feature maps to 1D vector
Dense (512 units)	Activation: ReLU — learns disease-specific features
BatchNormalization	Normalises activations for stable training
Dropout (0.5)	50% neuron drop — prevents overfitting
Dense (128 units)	Activation: ReLU — further feature compression
Dropout (0.3)	30% neuron drop — additional regularisation
Dense (4 units)	Activation: Softmax — probability per class
Output Classes	COVID19 | NORMAL | PNEUMONIA | TUBERCULOSIS

3.2 Two-Phase Training Strategy

Phase	Epochs	Learning Rate	What is trained	Optimizer
Phase 1 — Feature Extraction	10	0.001	Only top custom layers (base frozen)	Adam
Phase 2 — Fine-tuning	20	0.0001	Top custom layers + last layers of MobileNetV2 (first 100 base layers frozen)	Adam

3.3 Other Algorithms & Techniques Used

Algorithm / Technique	Purpose	Library
Softmax Classification	Converts logits to probability distribution (4 classes)	Keras
Categorical Cross-Entropy Loss	Multi-class training loss function	Keras
Adam Optimiser	Adaptive gradient descent for weight updates	TensorFlow
Class Weight Balancing	Corrects for dataset imbalance (Pneumonia >> TB)	scikit-learn / custom
Early Stopping	Stops training when val_loss stops improving (patience=6)	Keras
ReduceLROnPlateau	Reduces LR by 0.2× when val_loss plateaus (patience=3)	Keras
ModelCheckpoint	Saves best model weights by val_accuracy	Keras

Grad-CAM (Gradient-weighted Class Activation Maps)	Generates visual heatmaps showing which lung regions influenced the prediction	TensorFlow + OpenCV
Batch Normalisation	Normalises layer activations — faster, stable training	Keras
Dropout Regularisation	Randomly disables neurons to prevent overfitting	Keras
MobileNetV2 Preprocessing	Scales pixel values to [-1, 1] for ImageNet weights	Keras
GradientTape	Tracks gradients for Grad-CAM computation	TensorFlow
OpenCV COLORMAP_JET	Applies colour heatmap to the Grad-CAM output	OpenCV (cv2)

4. How the Project Works

End-to-End Prediction Pipeline:

Step 1 — Upload

Doctor/user registers or selects a patient, then uploads a chest X-ray image (PNG/JPG/JPEG, max 16 MB) through the web interface at localhost:5001.

Step 2 — File Validation & Storage

Flask validates file type and saves it securely with a UUID filename to static/uploads/. The file path is stored relative to the static folder.

Step 3 — CNN Inference

The image is resized to 224×224 pixels, preprocessed (pixel values scaled to [-1,1]), and passed through the MobileNetV2-based model (lung_model_multi.h5). Outputs: a 4-element softmax probability vector → argmax gives the predicted class.

Step 4 — Grad-CAM Generation

Using TensorFlow GradientTape, gradients of the predicted class score are computed with respect to the last Conv2D layer's output. These are averaged (Global Average Pooling) and used to weight the feature maps, producing a heatmap that is overlaid on the X-ray using OpenCV.

Step 5 — Medical Insights

Based on the predicted class and confidence, the system looks up AI insights, clinical description, diet plan, and medication recommendations from medical_insights.py.

Step 6 — Severity & Risk

Severity is computed from confidence: ≥85% → Severe, ≥65% → Moderate, <65% → Mild. Clinical risk is derived from the combination of severity and prediction class.

Step 7 — Database Storage

Patient info and the XRay report (prediction, confidence, severity, image path, heatmap path) are stored in the SQLite database via Flask-SQLAlchemy.

Step 8 — Result Display

The result.html template renders: AI prediction badge, confidence bar, Grad-CAM viewer, probability chart, medical insights, diet & medication cards, IoT vitals, and PACS history.

Step 9 — PDF Export

The /download-report/ route calls report_generator.py using ReportLab to generate a professional PDF clinical report and sends it as a file download.

IoT Vitals Monitoring:

The system includes an IoT integration layer (iot/api.py) that fetches real-time simulated patient vitals (Heart Rate, SpO■, Temperature, Respiratory Rate) from /api/simulated-vitals/ every 3 seconds via JavaScript polling. Vitals are stored in the Vitals database table and displayed live with a clinical clock on the result page.

5. Technology Stack

Layer	Technology	Version	Purpose
Web Framework	Flask	2.3.3	HTTP routing, templating, API
Deep Learning	TensorFlow / Keras	2.13 / 2.13	CNN training and inference
Base Model	MobileNetV2	ImageNet	Transfer learning backbone
Image Processing	OpenCV (cv2)	4.8.1	Grad-CAM overlay, image resize
Image Processing	Pillow	10.0.1	Image loading and conversion
Data Science	NumPy	1.24.3	Array operations
ML Utilities	scikit-learn	1.3.2	Class weights, evaluation
Visualisation	Matplotlib	3.7.3	Training plots
Database ORM	Flask-SQLAlchemy	3.0.5	Database models & queries
DB Migrations	Flask-Migrate	4.0.5	Schema version control
Database	SQLite (development)	Built-in	Patient & report storage
Database	PostgreSQL (production)	via psycopg2	Production DB (Supabase)
Frontend	React (TypeScript)	Vite build	Doctor & Patient portals
Frontend	HTML5 + Bootstrap 5	5.3.2	Prediction result pages
Auth	Flask-Login + Google OAuth	—	Role-based access control
PDF Generation	ReportLab	4.0.4	Clinical PDF report export
Server	Gunicorn	21.2.0	Production WSGI server
Cors	Flask-CORS	4.0.0	Cross-origin resource sharing

6. How to Run the Project (Terminal Guide)

Prerequisites:

- Python 3.9+ (check: python3 --version)
- Node.js 18+ (check: node --version)
- pip (Python package manager)
- Git (optional, for cloning)
- 4 GB+ RAM recommended for model loading

Step 1 — Navigate to the project folder

```
cd "/Users/britto/Documents/Lung Disease Project/lung-disease-ai"
```

Step 2 — Create and activate a virtual environment

```
python3 -m venv venv
```

```
source venv/bin/activate # macOS / Linux
```

```
venv\Scripts\activate # Windows
```

Step 3 — Install Python dependencies

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

Step 4 — Set up environment variables (copy .env.example to .env and fill in values)

```
cp .env.example .env
```

 # Then edit .env and set:

 # SECRET_KEY=your-secret-key

 # DATABASE_URL=sqlite:///lung_disease.db

 # GOOGLE_CLIENT_ID=... (optional)

Step 5 — Initialise the database

```
flask db upgrade
```

 # OR for first-time setup without migrations:

```
python -c "from app import app; from database.models import db;
app.app_context().push(); db.create_all()"
```

Step 6 — (Optional) Train the model

```
python model/train_multi_class.py
```

 # Place your dataset in: data/chest_xray_multi/train|val|test//

Step 7 — Install frontend dependencies

```
cd frontend
```

```
npm install
```

```
cd ..
```

Step 8 — Run the full project (Backend + Frontend together)

```
npm run dev
```

 # This runs both Flask (port 5001) and React dev server concurrently

Step 8 (Alternative) — Run backend only

```
python app.py
```

 # Flask runs at: http://localhost:5001

Step 9 — Open in browser

```
http://localhost:5001
```

 # Doctor Portal: http://localhost:5001/doctor

 # Patient Portal: http://localhost:5001/patient/

Useful Commands:

Task	Command
Generate a PDF report via CLI	`python generate_docs_pdf.py`
Run IoT vitals simulator	`python sim_iot.py`
Inspect trained model layers	`python model/list_layers.py`
Export model info to JSON	`python model/export_model_info.py`
Apply a new DB migration	`flask db migrate -m 'description' && flask db upgrade`
Run with Gunicorn (production)	`gunicorn -w 4 -b 0.0.0.0:5001 'app:app'`

7. Project Folder Structure

```
lung-disease-ai/ ■■■ app.py ← Main Flask application & all routes ■■■ config.py ←
Configuration (dev / prod / test) ■■■ requirements.txt ← Python dependencies ■■■
package.json ← Node scripts (npm run dev) ■■■ sim_iot.py ← IoT vitals simulator ■
■■■ model/ ■ ■■■ lung_model_multi.h5 ← Trained CNN model (generated after
training) ■ ■■■ train_multi_class.py ← Model training script (MobileNetV2) ■ ■■■
predict.py ← Inference + Grad-CAM generation ■ ■■■ medical_insights.py ← Disease
insights, diet, medication data ■ ■■■ report_generator.py ← PDF clinical report
generator (ReportLab) ■ ■■■ inspect_model.py ← Model layer inspection utility ■
■■■ database/ ■ ■■■ models.py ← SQLAlchemy models (Patient, XRayReport, Vitals,
User) ■ ■■■ enable_rls.sql ← Row-Level Security SQL (for Supabase) ■ ■■■ iot/ ■
■■■ api.py ← IoT Blueprint: simulated vitals API ■ ■■■ templates/ ← Jinja2 HTML
templates ■ ■■■ index.html ← Home / Upload page ■ ■■■ result.html ← Analysis
result page ■ ■■■ login.html ← Login page ■ ■■■ register.html ← Register page ■
■■■ product.html / technology.html ■ ■■■ static/ ■ ■■■ css/style.css ← Custom
styles ■ ■■■ js/main.js ← Frontend JS ■ ■■■ uploads/ ← Saved X-ray uploads ■ ■■■
heatmaps/ ← Grad-CAM heatmap images ■ ■■■ reports/ ← Generated PDF reports ■ ■■■
fe_assets/ ← React build output (Doctor/Patient portals) ■ ■■■ frontend/ ← React +
TypeScript source (Vite) ■ ■■■ src/pages/ ← Dashboard, PatientPortal, DoctorPortal
■ ■■■ src/components/ ← Reusable UI components ■ ■■■ data/ ■ ■■■
chest_xray_multi/ ← Training dataset (train/val/test//) ■ ■■■ migrations/ ←
Flask-Migrate database migrations
```

Disclaimer: *LungAI is an engineering and educational demonstration project. All AI predictions are for informational purposes only and must NOT be used as a substitute for professional medical diagnosis. Always consult a qualified medical professional. Medication and diet information provided is general and informational — not a prescription.*