Technical Report: Explainable AI Platform

Deepfake Audio Detection and Lung Cancer Classification

Team Members:
Pierre Briand, Tom Delahaye, Alexandre Laroudie,
Kentin Guillemot, Gabriel Carlotti

January 7, 2026

Abstract

This project delivers a unified **Explainable Artificial Intelligence (XAI)** platform serving two distinct modalities: **Deepfake Audio Detection** (classifying audio as REAL or FAKE) and **Lung Cancer Classification** (diagnosing chest X-rays). The platform integrates three major explainability techniques (**Grad-CAM**, **LIME**, and **SHAP**) and provides a modern, responsive web interface for interactive analysis with side-by-side comparison of explanations.

Contents

1 Introduction and Context

1.1 Problem Statement

As AI systems become increasingly prevalent in high-stakes domains such as healthcare and media authenticity verification, the need for **transparent and interpretable** decision-making becomes critical. This platform addresses:

1. **Media Authenticity:** Detecting AI-generated audio deepfakes that pose threats to personal and organizational security.
2. **Medical Diagnosis:** Assisting healthcare professionals in lung cancer detection with explainable predictions.

1.2 Objectives

- Provide accurate classification using state-of-the-art deep learning models.
- Enable transparency through multiple XAI techniques.
- Deliver a user-friendly interface for non-technical users.
- Ensure modular, maintainable, and extensible architecture.

1.3 Technology Stack Rationale

Technology	Purpose	Rationale
FastAPI	REST API	Lightweight, asynchronous, automatic OpenAPI docs
TensorFlow/Keras	Audio Models	Compatibility with spectrogram-based transfer learning
PyTorch	Image Models	Native Grad-CAM hooks, torchvision integration
React + Vite	Frontend	Modern SPA with TanStack Router for routing
TailwindCSS v4	Styling	Utility-first CSS with responsive design

2 System Architecture

2.1 Overall Architecture

The system follows a modular client-server architecture, separating the frontend visualization from the backend inference engines.

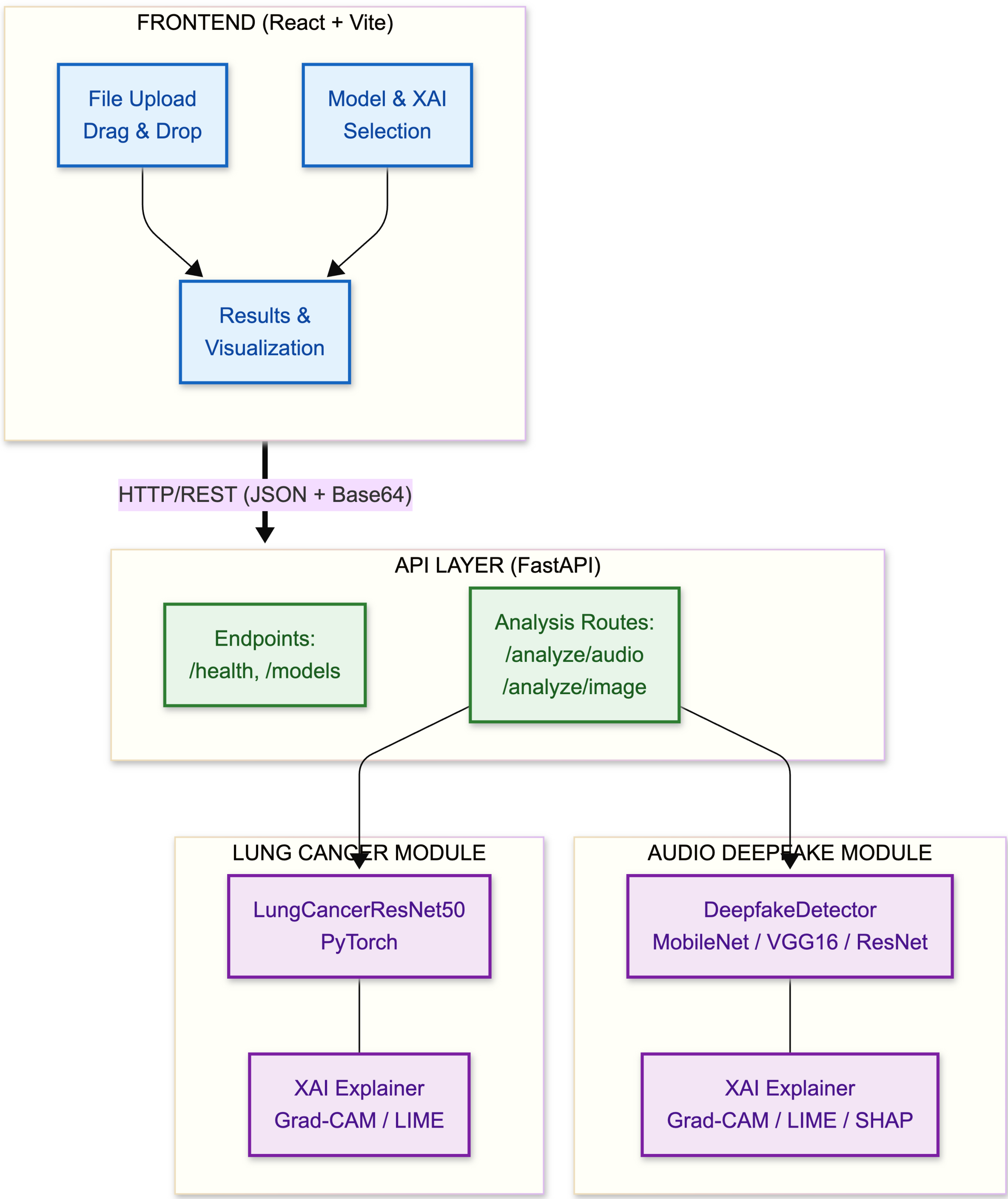

Figure 1: System Architecture Overview

2.2 Backend API Service

The central API service (`api.py`) provides a unified interface for both modalities.

- **Lazy Loading:** Modules are loaded on-demand.
- **CORS Support:** Full cross-origin resource sharing.
- **Base64 Encoding:** XAI visualizations are returned as encoded PNG images.

3 Technical Implementation Details

3.1 Audio Processing Pipeline

Audio files are converted to Mel-spectrograms before being fed into CNN models originally designed for image recognition.

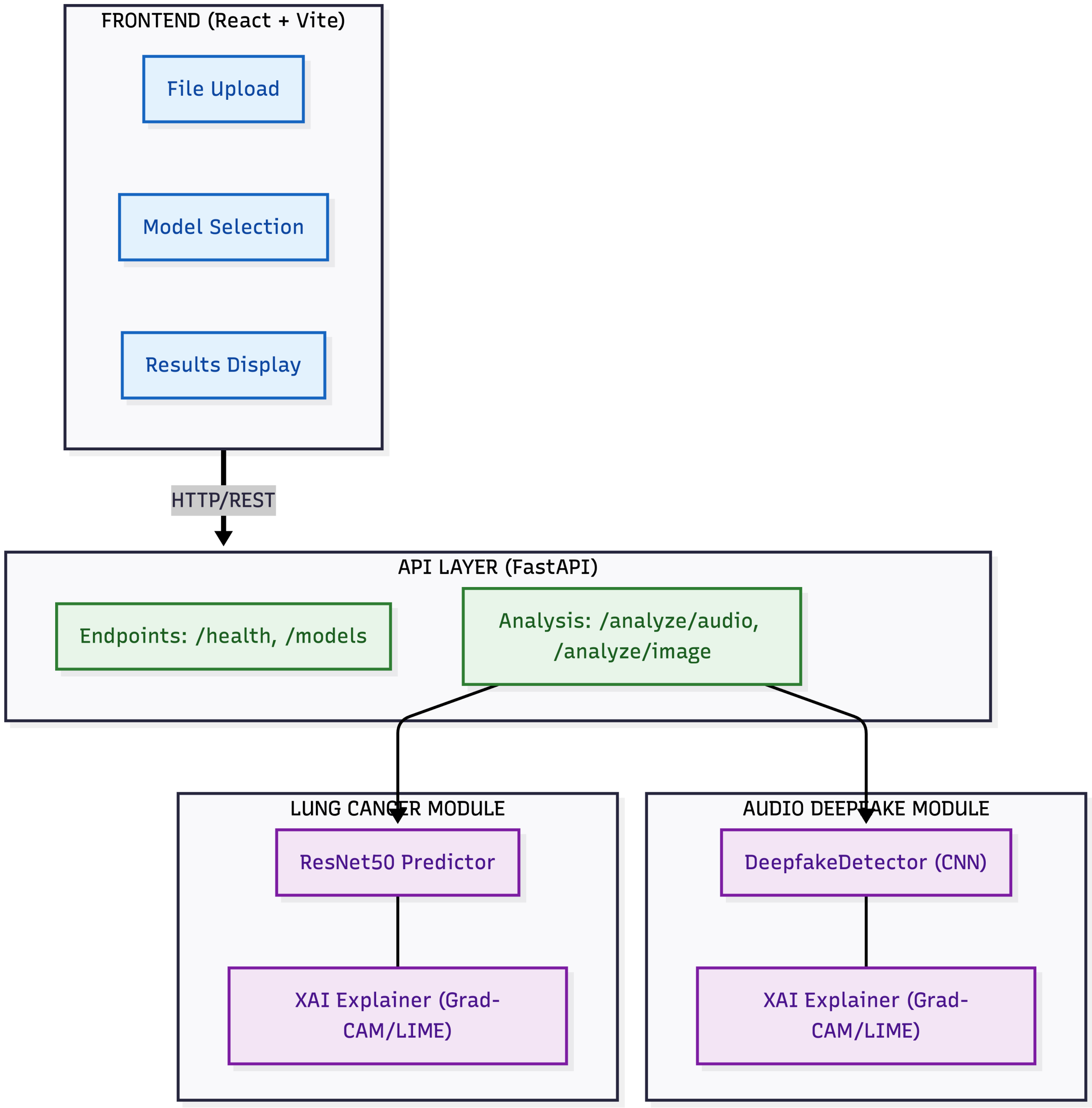

Figure 2: Audio Processing Pipeline

Spectrogram Parameters: Sample rate: Native; Mel bands: 128; Max frequency: 8000 Hz; Conversion: Power to dB.

3.2 XAI Methods Implementation

3.2.1 Grad-CAM

- **Audio (TF):** Computed on the last convolutional layer. Weights feature maps by mean gradients, applied ReLU, and overlaid on the spectrogram.
- **Image (PyTorch):** Uses hooks on `layer4` to compute weighted sum: $\sum(mean\_grad_i \times feature_i)$.

3.2.2 LIME SHAP

Both modules use `lime.lime_image.LimeImageExplainer` (generating perturbations on superpixels) and gradient-based SHAP to compute feature attribution values.

4 Models and Algorithms

4.1 Audio Models (Spectrogram-based)

All models utilize ImageNet pretrained weights and output 2 classes (REAL, FAKE).

Model	Base Arch.	Head Architecture
MobileNet	MobileNet V1	GAP → Dense(128) → Dropout → Dense(2)
VGG16	VGG16	Flatten → Dense(256) → Dropout → Dense(128) → Dense(2)
ResNet50	ResNet50	GAP → Dense(256) → Dropout → Dense(128) → Dense(2)

4.2 Image Model

LungCancerResNet50: Based on ResNet50 (ImageNet). Custom head: Linear(2048→128) → ReLU → Dropout → Linear(128→3). The three classes are Adenocarcinoma, Benign, and Squamous Cell Carcinoma.

5 API Reference

5.1 Example Endpoint: Analyze Audio

`POST /api/analyze/audio`


```
{
    "model": "deepfake-mobilenet",
    "prediction": "FAKE",
    "confidence": 0.847,
    "probabilities": {"real": 0.153, "fake": 0.847},
    "xai_results": {
        "lime": {
            "type": "lime",
            "image": "data:image/png;base64,iVBORw0KGgo...",
            "num_samples": 1000
        }
    }
}
```


6 Key Improvements and Innovations

- **Unified Platform:** Single API surface for multiple modalities.

- **Robust Error Handling:** Graceful fallbacks for XAI failures and detailed status codes.
- **Performance:** Lazy loading of heavy ML libraries and GPU support detection.
- **Developer Experience:** Auto-generated Swagger docs and TypeScript frontend.

7 Getting Started

7.1 Backend Setup

```
cd XAI_Final_Project
pip install -r requirements.txt
python api.py
# API available at http://localhost:5000
```

7.2 Frontend Setup

```
cd frontend
pnpm install
pnpm dev
# App available at http://localhost:3000
```

8 Future Enhancements

- **Short-term:** File size enforcement and result caching.
- **Medium-term:** Batch processing and PDF report generation.
- **Long-term:** Additional modalities (Video/Text) and in-platform model training.

Document Version: 2.0 | Last Updated: January 7, 2026