Tracer

Tracer (stylized as t acer) is a multimodal deepfake intelligence platform. The Next.js web client guides journalists, brand teams, and security analysts through uploading suspicious media, while a FastAPI backend runs an ensemble of state-of-the-art computer vision and audio models to score authenticity in near real time.

Overview

- **Users** sign in through the web dashboard, upload an image, video, or audio clip, and track previous submissions.
- **Frontend** (Next.js/React + Tailwind) handles authentication, drag-and-drop uploads, live status cards, and report visualization.
- **Backend** (FastAPI + PyTorch) exposes `/detect/image`, `/detect/video`, and `/detect/audio` endpoints, orchestrating deepfake detection models and fallbacks.
- **Inference** leverages pretrained EfficientNet-based classifiers, attention-augmented DFDC models, and handcrafted audio forensics to return confidence scores and contextual analysis.

System Flow

1. **Upload & Routing** – The dashboard posts media files to the FastAPI gateway with secure multipart requests.
2. **Staging** – Backend stores the payload in a temporary location and routes to `DeepSecureInference`.
3. **Model Loading** – `DeepfakeModelLoader` hydrates checkpoints from `backend/DeepSecure-AI/checkpoints`, downloading demo weights on demand when EfficientNet is available.
4. **Ensemble Prediction** – For images (and sampled video frames) the loader fans out to:
    - `FaceForensicsModel` (`efficientnet-b4` backbone with custom binary head)
    - `DFDCModel` (`efficientnet-b7` backbone + attention map + global pooling)
    - `CelebDFModel` (`efficientnet-b4` backbone with deeper classifier) Individual logits are aggregated into an ensemble fake probability; per-model predictions are reported for transparency.
5. **Video Analysis** – `DeepSecureInference.detect_video` extracts evenly sampled frames via OpenCV, reuses the image ensemble, and calculates a consistency score across frames.
6. **Audio Analysis** – When deepfake checkpoints are unavailable, the engine falls back to spectral, temporal, MFCC, and harmonic heuristics built with Librosa to estimate synthetic speech likelihood.

7. **Response** – JSON payload contains `is_fake`, confidence, detection method, model metadata, and analysis artifacts for UI consumption.

If GPU-accelerated EfficientNet weights are missing, the system degrades gracefully to classical CV heuristics (noise patterns, compression artifacts, face alignment) so users still receive a best-effort verdict.

Backend Models at a Glance

- `backend/deepfake_models.py` defines the EfficientNet-based detectors and the `DeepfakeModelLoader`, which abstracts checkpoint management, preprocessing, and ensemble fusion.
- `backend/inference.py` hosts the `DeepSecureInference` orchestration class, bundling image/video/audio flows, temporary file hygiene, and error handling.
- `backend/DeepSecure-AI/models` contains legacy architectures (e.g., RawNet, TMC) and training scripts for experiments or retraining.
- `backend/inference.py` also implements CV fallbacks and audio forensics so the API remains robust in constrained environments.

Frontend Experience

The Next.js app (`frontend/app`) provides: - Marketing-style landing pages encouraging sign-up and demo trials. - Dashboard views (`frontend/app/dashboard`) for analytics, notification history, and upload management. - Animated sign-in/up flows with custom components (`frontend/components`). - UI primitives built atop a ShadCN-style component kit for consistent theming.

Running the Project

1. **Backend**

   ```
   cd backend
   pip install -r requirements.txt
   python run.py          # or: uvicorn main:app --reload
   ```

2. **Frontend**

   ```
   cd frontend
   npm install
   npm run dev
   ```

3. Visit `http://localhost:3000` for the UI and `http://localhost:8000/docs` for API exploration.

AI Acceleration Report

AI in Our Process

- Cursor: Assisted daily with boilerplate generation, rapid refactors, and repo-wide navigation.
- ChatGPT (GPT-5 Codex): Helped design the ensemble architecture, debug device-placement bugs, and summarize research for documentation.

Impact on Workflow

- **Faster prototyping** – Cursor + Copilot produced initial model wrappers (`FaceForensicsModel`, `DFDCModel`, `CelebDFModel`) and inference scaffolding so we could focus on training IDs and evaluation.
- **Debugging support** – ChatGPT traced tricky dependency issues (EfficientNet imports, gdown fallbacks) and suggested defensive coding patterns that shortened troubleshooting loops.
- **Documentation & planning** – Gemini and ChatGPT drafted README sections, dependency notes, and sprint outlines, letting the team iterate instead of starting from zero.
- **Consistency boosts** – Copilot kept coding style uniform across frontend components and backend services, reducing review cycles and merge conflicts.

AI in Our Product

- **What it does** – The platform ingests images, videos, and audio clips, runs sophisticated deepfake checks, and returns human-readable authenticity reports with confidence metrics.
- **Models & APIs** – PyTorch EfficientNet backbones (B4/B7), custom attention modules, pretrained checkpoints distributed via Google Drive/gdown, and Librosa-powered audio forensics. Optional fallbacks rely on OpenCV, NumPy, and handcrafted signal-processing heuristics to maintain coverage when deep networks are unavailable.

Questions or ideas? Open an issue or reach out—we're actively evolving t acer to stay ahead of the deepfake threat curve.