Product Requirements Document (PRD)

Product Name: TraceFinder
Category: Digital Forensics / Image Source Identification
Prepared For: Academic & Research Implementation
Primary Technologies: Python, Streamlit, Google Colab, OpenCV, TensorFlow/PyTorch

1. Overview

TraceFinder is an AI-powered forensic analysis tool designed to identify the source scanner device used to digitize a document or image. Each scanner leaves unique digital artifacts — such as noise patterns, color inconsistencies, or compression traces — that can be detected using computer vision and deep learning. This tool helps forensic investigators, legal professionals, and document authentication systems determine whether a document was scanned from an authorized or fraudulent source.

2. Goals and Objectives

Goal: Identify the scanner brand or model from a scanned image.

Objectives:

- Analyze intrinsic noise patterns in scanned images.
- Build and train a classification model for scanner identification.
- Provide a user-friendly Streamlit interface for image upload and prediction.
- Offer explainable AI (Grad-CAM/SHAP) insights to validate model predictions.

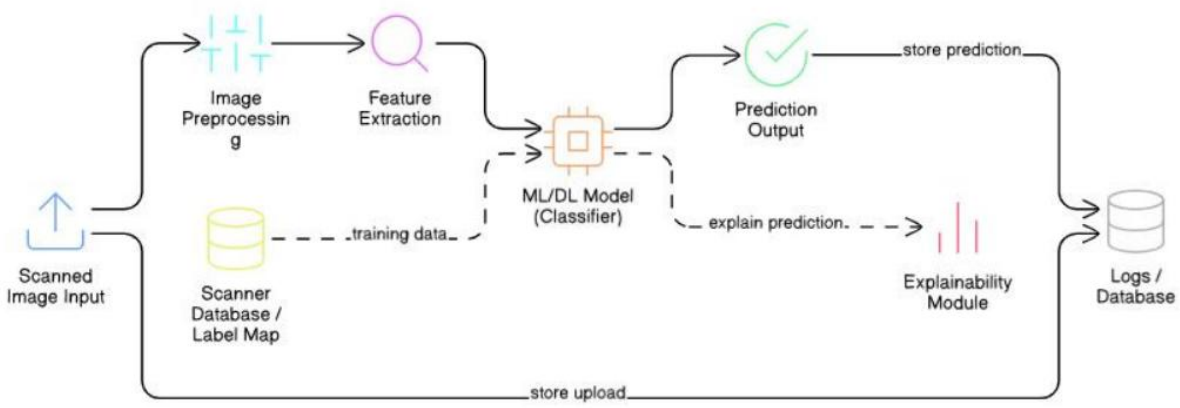
3. Use Cases

Use Case	Description	Example
Digital Forensics	Detect forged or duplicate legal/scanned documents.	Check if a fake certificate originated from a specific scanner.
Document Authentication	Identify the source scanner to verify authenticity.	Differentiate between scans from authorized vs unauthorized departments.
Legal Evidence Verification	Validate the origin of scanned copies in legal or corporate contexts.	Ensure documents came from a company-approved scanner.

4. Technical Architecture

System Flow:

1. Input: User uploads an image (PNG, JPG, TIFF, etc.)
2. Preprocessing: Resize, grayscale conversion, normalization, optional denoising.
3. Feature Extraction: Compute texture, PRNU, FFT noise maps, and scanner artifacts.
4. Model Prediction: CNN or hybrid model predicts the scanner source.
5. Output: Display predicted scanner brand/model with confidence score + visual explanation (Grad-CAM).



5. Tech Stack & Tools

Category	Tools/Frameworks	Purpose
Language	Python (≥3.10)	Core logic, data processing
Libraries	NumPy, Pandas, OpenCV, Scikit-learn	Preprocessing & feature extraction
Deep Learning	TensorFlow / PyTorch	CNN training and inference
Visualization	Matplotlib, Seaborn	Training analysis, Grad-CAM
Explainability	SHAP, LIME, Grad-CAM	Model interpretability
Frontend	Streamlit	Upload interface and live inference
Environment	Google Colab / JupyterLab	Model experimentation
Version Control	Git + GitHub	Code management

Dataset Handling	Flatfield dataset	Training and testing

6. Knowledge & Concepts Required

- Image Processing: Noise filtering, FFT, PRNU extraction, and texture mapping.
- Machine Learning & Deep Learning: Feature extraction, CNNs, and classifier training.
- Explainable AI (XAI): Grad-CAM / SHAP to visualize model focus regions.
- Web App Development: Streamlit app for deployment and inference.
- Evaluation Metrics: Accuracy, Precision, Recall, F1-Score, Confusion Matrix.