Verifiable AI Decisions

Artificial intelligence is increasingly used in healthcare for clinical decision support, yet the accountability of AI-generated decisions remains a critical challenge. AI models evolve over time, inference can be non-deterministic, and decisions may be questioned long after they are made. This project presents a system for creating a **tamper-proof and privacy-preserving proof of an AI decision** by storing only a cryptographic fingerprint of the decision on a blockchain. The fingerprint uniquely binds the input, output, and model configuration used at inference time, without storing any medical data or AI models on-chain. During audits or disputes, the decision can be independently verified for integrity without re-running the AI or exposing sensitive information. The system ensures that while AI models may change, the accountability of past decisions remains immutable.

Problem Statement

The use of Artificial Intelligence in healthcare is rapidly increasing, particularly for tasks such as diagnosis support, risk prediction, and clinical decision assistance. While these systems can improve efficiency and outcomes, they introduce a critical challenge: long-term accountability of AI decisions.

AI-generated outputs may change due to:

- updates or replacement of models,
- non-deterministic inference behavior,
- modification or loss of historical logs,
- or post-hoc alteration of stored results.

When an AI-assisted decision is questioned weeks, months, or years later, healthcare providers currently lack a reliable way to prove what the AI decided at the exact moment the decision was made.

Existing solutions rely on centralized databases and application logs, which are:

- fully controlled by the institution,
- mutable by administrators,
- and not independently verifiable by regulators or third parties.

Although blockchain offers immutability and third-party trust, directly storing medical data or AI outputs on-chain violates privacy regulations, introduces scalability issues, and exposes sensitive information.

Therefore, there is a need for a system that can:

- provide immutable proof of an AI decision,
- preserve patient privacy,
- avoid re-running AI models,
- and allow independent verification of decision integrity.

This project addresses this gap by enabling verifiable, tamper-proof AI decision accountability without storing medical data or models on the blockchain.

Core Logic

The core logic of this system is to create a **tamper-proof cryptographic proof of an AI decision** at the moment it is made, without storing sensitive data or AI models on the blockchain.

Instead of verifying *how* the AI works, we verify **that the exact decision used at that time has not been altered later**.

Key Principle

Same input + same model + same parameters must always produce the same cryptographic fingerprint.
If any part changes, the fingerprint changes.

System Methodology (Core Logic)

Step 1: AI Decision Generation

- Input: Demo medical report (text / PDF)
- Output: Risk prediction
  (e.g., "Pneumonia risk: 70%")

This output is assumed to be **used in real decision-making**

Step 2: Model & Inference Fingerprinting

We capture **exact metadata** used during inference:

- Model ID (e.g., radiology-v1.0)
- Model checksum (hash of model file / weights)
- Inference parameters (thresholds, configs)

This guarantees:

"This output came from this exact model under these exact conditions."

Step 3: Cryptographic Fingerprint Creation

We generate a **single deterministic hash**:

```
AI_Decision_Fingerprint = hash(
   input_hash +
   output_value +
   model_id +
   model_checksum +
   inference_parameters
)
```

This fingerprint uniquely represents **one AI decision at one moment in time**.

Step 4: Blockchain Anchoring

Only the following is stored on-chain:

- AI Decision Fingerprint
- Timestamp

Never stored on-chain:

- Patient data
- Medical reports
- AI models
- Prediction values

This ensures **privacy + immutability**.

Step 5: Verification (Audit / Dispute)

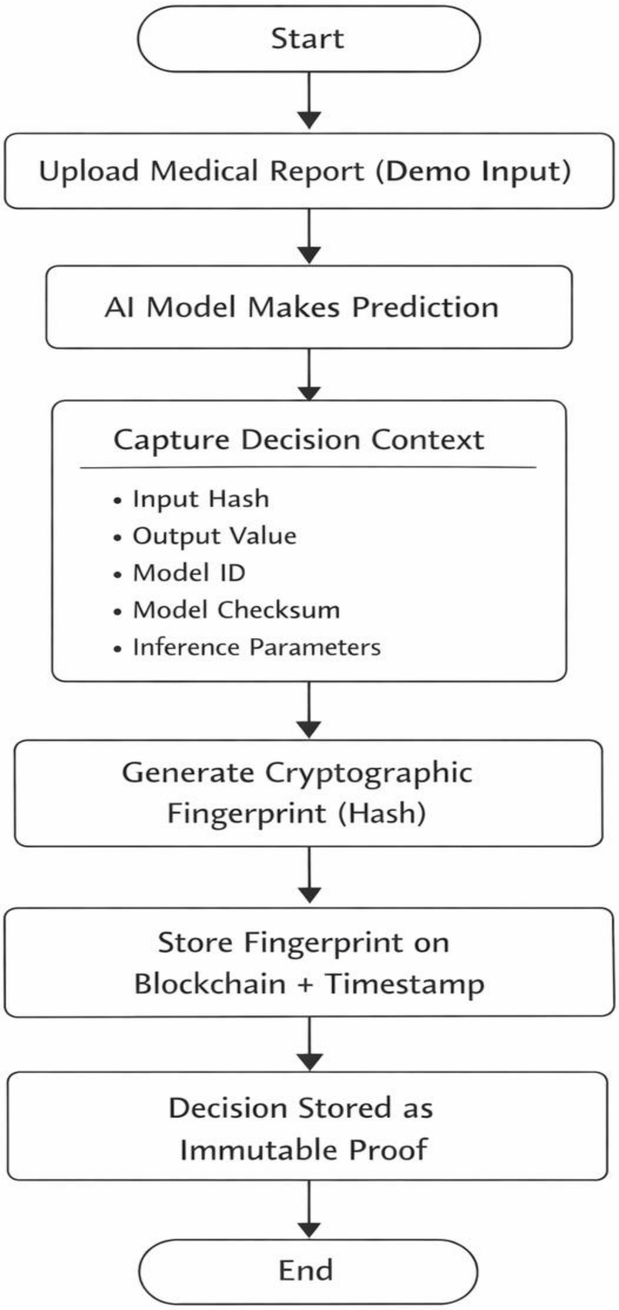
During verification:

1. Hospital provides stored input & output
2. Fingerprint is regenerated
3. Compared with blockchain record

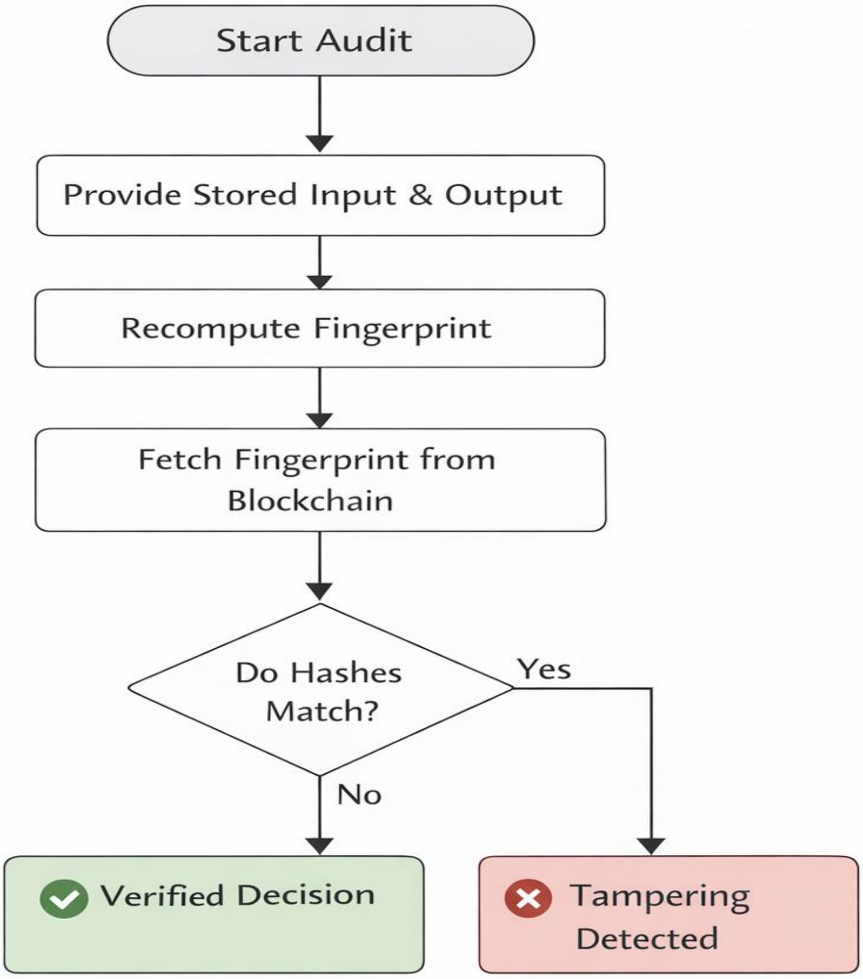
Results:

- Match → Decision authentic
- Mismatch → Decision altered

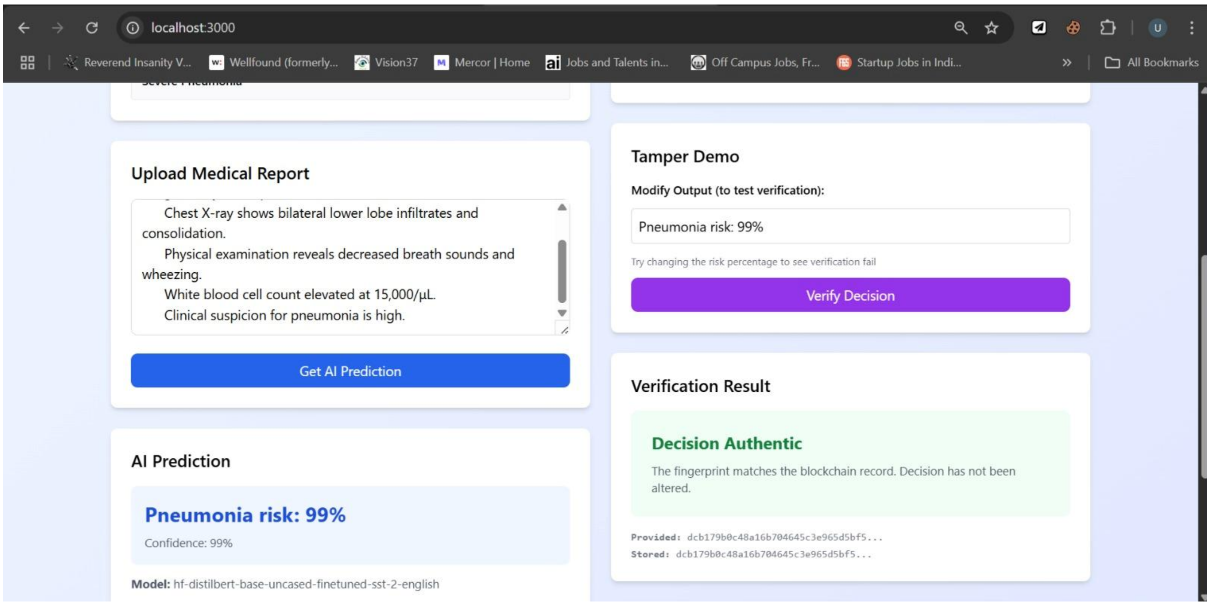
MAIN FLOW


Start
Upload Medical Report (Demo Input)
AI Model Makes Prediction
Capture Decision Context
• Input Hash
• Output Value
• Model ID
• Model Checksum
• Inference Parameters
Generate Cryptographic Fingerprint (Hash)
Store Fingerprint on Blockchain + Timestamp
Decision Stored as Immutable Proof
End

VERIFICATION FLOWCHART



OUTPUT OF ROUND 1



Upload Medical Report

Chest X-ray shows bilateral lower lobe infiltrates and consolidation.
Physical examination reveals decreased breath sounds and wheezing.
White blood cell count elevated at 15,000/μL.
Clinical suspicion for pneumonia is high.

AI Prediction

Pneumonia risk: 99%

Confidence: 99%

Model: hf-distilbert-base-uncased-finetuned-sst-2-english

Tamper Demo

Modify Output (to test verification):

Pneumonia risk: 99%

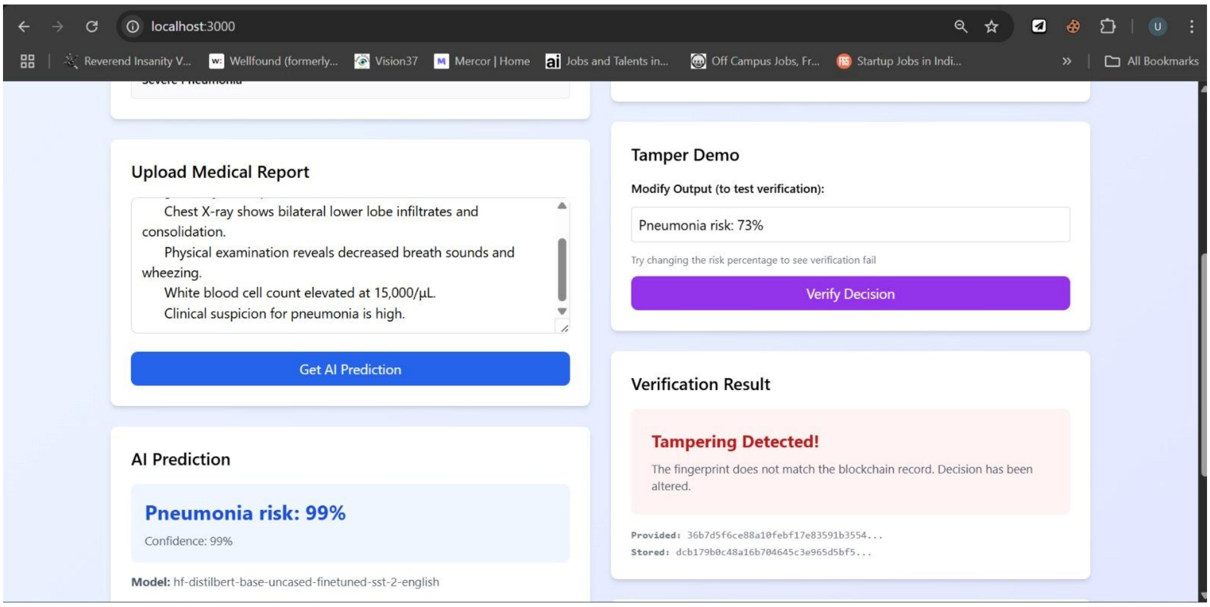
Try changing the risk percentage to see verification fail

Verification Result

Decision Authentic

The fingerprint matches the blockchain record. Decision has not been altered.

Provided: dcb179b0c48a16b704645c3e965d5bf5...
Stored: dcb179b0c48a16b704645c3e965d5bf5...



Upload Medical Report

Chest X-ray shows bilateral lower lobe infiltrates and consolidation.
Physical examination reveals decreased breath sounds and wheezing.
White blood cell count elevated at 15,000/μL.
Clinical suspicion for pneumonia is high.

AI Prediction

Pneumonia risk: 99%

Confidence: 99%

Model: hf-distilbert-base-uncased-finetuned-sst-2-english

Tamper Demo

Modify Output (to test verification):

Pneumonia risk: 73%

Try changing the risk percentage to see verification fail

Verification Result

Tampering Detected!

The fingerprint does not match the blockchain record. Decision has been altered.

Provided: 36b7d5f6ce88a10febf17e83591b3554...
Stored: dcb179b0c48a16b704645c3e965d5bf5...

Round 2: System Enhancements & Upgrades

1. **Merkle Tree–Based Batch Verification**

- Introduces batch processing of AI decision fingerprints.
- Individual decision hashes are aggregated into a **Merkle Tree**.
- Only the **Merkle Root** is stored on the blockchain.

Technical Advantages:

- Significantly reduces on-chain storage requirements.
- Enables verification of individual decisions using Merkle proofs.
- Preserves privacy by avoiding exposure of unrelated decision data.
- Improves performance and reduces gas costs.

2. **Commitment–Reveal Protocol**

- Implements a two-phase cryptographic commitment mechanism.

Commit Phase (Decision Time):

- A cryptographic commitment (hash) of the AI decision is recorded on-chain.
- No sensitive data or prediction values are disclosed.

Reveal Phase (Audit Time):

- Original decision data is disclosed only during verification.
- Commitment is recomputed and matched against the blockchain record.

Security Benefits:

- Prevents post-decision tampering or back-dated modifications.
- Provides verifiable proof of decision existence at a specific time.
- Aligns with standard cryptographic audit practices.

3. **Enhanced Verification Workflow**

- Supports batch verification using Merkle proofs.
- Separates decision creation from audit verification.
- Enables efficient third-party auditing without re-executing AI models.
- Maintains full privacy by keeping patient data off-chain.

## Setup

1. Install dependencies:
```bash
npm install
cd frontend && npm install
cd ../backend && npm install
cd ../contracts && npm install
cd ../ai-service && pip install -r requirements.txt
```

2. Start blockchain:
```bash
cd contracts
npx hardhat node
```

3. Deploy contract:
```bash
cd contracts
npx hardhat run scripts/deploy.js --network localhost
```

4. Configure backend - create `backend/.env`:
```
PORT=3001
AI_SERVICE_URL=http://localhost:5000
RPC_URL=http://localhost:8545
CONTRACT_ADDRESS=<contract address from step 3>
PRIVATE_KEY=<private key from hardhat node>
```

5. Start services:
```bash
# Terminal 1: AI Service
cd ai-service && python app.py

# Terminal 2: Backend
cd backend && npm run dev

# Terminal 3: Frontend
cd frontend && npm run dev
```

Visit http://localhost:3000