***CAN BLOCKCHAIN IMPROVE INTERNAL AUDIT INTEGRITY***

A Python-Based Simulation and Real-World Audit Perspective

By Mayank Agarwal | Ex-EY | MSc FinTech – University of Liverpool

# Executive Summary

This project explores how blockchain technology can be applied to internal auditing to reduce fraud, increase transparency, and improve trust in financial systems. Drawing from my audit experience at Ernst & Young (EY), I simulate a Python-based blockchain ledger to demonstrate how immutable, timestamped entries can create fraud-resistant audit trails.

# Background

In many audit engagements, internal control failures such as undocumented journal entries, backdated adjustments, and approval overrides increase the risk of material misstatements or fraud. Traditional ERPs allow overwriting or deletion of entries, something auditors constantly test through vouching, walkthroughs, and control checks.  
This project asks: Can blockchain be used to create tamper-proof audit trails that self-validate and eliminate audit override risk?

# Blockchain for Audit: The Concept

A blockchain is a linked list of blocks, each containing data, a timestamp, and a hash of the previous block. It’s:  
- Immutable: Once written, data can’t be changed.  
- Transparent: Each new entry is publicly visible to nodes.  
- Secure: Tampering with one block changes all subsequent hashes.  
This makes blockchain perfect for tracking financial transactions or journal entries in a controlled environment.

# The Python Simulation

Using Python, I built a basic blockchain prototype that simulates how audit records can be stored across blocks. Each block contains:  
- Entry ID  
- Timestamp  
- Data (e.g., journal entry or approval)  
- Hash of the previous block

**Sample Python Code:**

import hashlib  
import datetime  
  
class Block:  
 def \_\_init\_\_(self, index, timestamp, data, previous\_hash):  
 self.index = index  
 self.timestamp = timestamp  
 self.data = data  
 self.previous\_hash = previous\_hash  
 self.hash = self.calculate\_hash()  
  
 def calculate\_hash(self):  
 content = str(self.index) + str(self.timestamp) + str(self.data) + self.previous\_hash  
 return hashlib.sha256(content.encode()).hexdigest()

# Key Takeaways

- Tamper Detection: Any change in audit data breaks the hash chain, flagging it instantly.  
- Timestamped Records: Entries can’t be backdated or retro-edited.  
- Transparency: Auditors get a verifiable trail without needing ERP log exports.  
- Trust Layer: External auditors can access hash records to verify internal processes.

# Reflection

This project allowed me to merge my auditing background at EY with FinTech skills learned at the University of Liverpool. I believe blockchain has strong potential in internal audit applications, especially for:  
- High-risk industries (insurance, real estate, banking)  
- AI/ML-based decision engines (e.g., auto-lending)