import hashlib

import os

import json

# Configuration

HASHES\_FILE = "file\_hashes.json"

DIRECTORY\_TO\_MONITOR = "./monitor\_folder"

def calculate\_file\_hash(file\_path):

"""Calculate SHA-256 hash of a file."""

hasher = hashlib.sha256()

try:

with open(file\_path, 'rb') as f:

while chunk := f.read(4096):

hasher.update(chunk)

return hasher.hexdigest()

except Exception as e:

print(f"Error hashing file {file\_path}: {e}")

return None

def load\_hashes():

"""Load previously stored hashes."""

if os.path.exists(HASHES\_FILE):

with open(HASHES\_FILE, 'r') as f:

return json.load(f)

return {}

def save\_hashes(hashes):

"""Save the latest hashes."""

with open(HASHES\_FILE, 'w') as f:

json.dump(hashes, f, indent=4)

def check\_integrity():

"""Compare current file hashes with stored hashes."""

stored\_hashes = load\_hashes()

current\_hashes = {}

for root, \_, files in os.walk(DIRECTORY\_TO\_MONITOR):

for file in files:

file\_path = os.path.join(root, file)

file\_hash = calculate\_file\_hash(file\_path)

if file\_hash:

current\_hashes[file\_path] = file\_hash

if file\_path in stored\_hashes:

if stored\_hashes[file\_path] != file\_hash:

print(f"[MODIFIED] {file\_path}")

else:

print(f"[NEW FILE] {file\_path}")

for file\_path in stored\_hashes.keys():

if file\_path not in current\_hashes:

print(f"[DELETED] {file\_path}")

save\_hashes(current\_hashes)

def main():

print("Starting File Integrity Checker...")

check\_integrity()

print("Integrity check completed.")

if \_\_name\_\_ == "\_\_main\_\_":

main()