import os

import subprocess

import logging

import sqlite3

from datetime import datetime

# Logger configuration

def create\_logger(log\_file):

if not os.path.exists(os.path.dirname(log\_file)):

os.makedirs(os.path.dirname(log\_file))

logger = logging.getLogger('PatchManagementLogger')

logger.setLevel(logging.INFO)

fh = logging.FileHandler(log\_file)

fh.setLevel(logging.INFO)

formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')

fh.setFormatter(formatter)

if not logger.handlers:

logger.addHandler(fh)

return logger

# Database setup

def initialize\_db(db\_path='patches.db'):

conn = sqlite3.connect(db\_path)

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS patches (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

description TEXT,

status TEXT DEFAULT 'pending',

applied\_on TEXT

)

''')

conn.commit()

conn.close()

# CRUD Operations

def create\_patch(name, description, db\_path='patches.db'):

conn = sqlite3.connect(db\_path)

cursor = conn.cursor()

cursor.execute('''

INSERT INTO patches (name, description)

VALUES (?, ?)

''', (name, description))

conn.commit()

conn.close()

print(f"Patch '{name}' created successfully.")

def read\_patches(db\_path='patches.db'):

conn = sqlite3.connect(db\_path)

cursor = conn.cursor()

cursor.execute('SELECT \* FROM patches')

rows = cursor.fetchall()

conn.close()

return rows

def update\_patch(patch\_id, name=None, description=None, status=None, db\_path='patches.db'):

conn = sqlite3.connect(db\_path)

cursor = conn.cursor()

if name:

cursor.execute('UPDATE patches SET name = ? WHERE id = ?', (name, patch\_id))

if description:

cursor.execute('UPDATE patches SET description = ? WHERE id = ?', (description, patch\_id))

if status:

cursor.execute('UPDATE patches SET status = ? WHERE id = ?', (status, patch\_id))

conn.commit()

conn.close()

print(f"Patch ID '{patch\_id}' updated successfully.")

def delete\_patch(patch\_id, db\_path='patches.db'):

conn = sqlite3.connect(db\_path)

cursor = conn.cursor()

cursor.execute('DELETE FROM patches WHERE id = ?', (patch\_id,))

conn.commit()

conn.close()

print(f"Patch ID '{patch\_id}' deleted successfully.")

# Check and install PSWindowsUpdate module

def install\_pswindowsupdate\_module(log):

try:

log.info("Checking if PSWindowsUpdate module is installed...")

result = subprocess.run(

['powershell', '-Command', 'Get-Module -ListAvailable -Name PSWindowsUpdate'],

stdout=subprocess.PIPE, text=True

)

if "PSWindowsUpdate" not in result.stdout:

log.info("PSWindowsUpdate module not found. Installing PSWindowsUpdate module...")

subprocess.run(

['powershell', '-Command', 'Install-Module -Name PSWindowsUpdate -Force -Scope CurrentUser'],

check=True

)

log.info("PSWindowsUpdate module installed successfully.")

else:

log.info("PSWindowsUpdate module already installed.")

except subprocess.CalledProcessError as e:

log.error(f"Failed to install PSWindowsUpdate module: {e}")

raise

# Apply system patches

def apply\_system\_patches(patch\_id, log, db\_path='patches.db'):

try:

log.info(f"Applying patch ID: {patch\_id}")

# Fetch patch details from DB

conn = sqlite3.connect(db\_path)

cursor = conn.cursor()

cursor.execute('SELECT name, description FROM patches WHERE id = ?', (patch\_id,))

patch = cursor.fetchone()

conn.close()

if not patch:

log.error(f"No patch found with ID: {patch\_id}")

print(f"No patch found with ID: {patch\_id}")

return

patch\_name, patch\_description = patch

log.info(f"Applying Patch: {patch\_name} - {patch\_description}")

# Import PSWindowsUpdate module

subprocess.run(['powershell', 'Import-Module PSWindowsUpdate'], check=True)

# Install the update (for demonstration, we use Install-WindowsUpdate)

log.info("Initiating Windows Update installation via PowerShell...")

subprocess.run('powershell Install-WindowsUpdate -AcceptAll -AutoReboot', shell=True, check=True)

# Update patch status in DB

applied\_on = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

conn = sqlite3.connect(db\_path)

cursor = conn.cursor()

cursor.execute('UPDATE patches SET status = ?, applied\_on = ? WHERE id = ?', ('applied', applied\_on, patch\_id))

conn.commit()

conn.close()

log.info(f"Patch ID {patch\_id} applied successfully.")

print(f"Patch ID {patch\_id} applied successfully.")

except subprocess.CalledProcessError as e:

log.error(f"Failed to apply patch ID {patch\_id}: {e}")

print(f"Failed to apply patch ID {patch\_id}: {e}")

# Monitor patch effects

def monitor\_patch\_effects(patch\_id, log, db\_path='patches.db'):

try:

log.info(f"Monitoring effects for patch ID: {patch\_id}")

# Fetch patch details from DB

conn = sqlite3.connect(db\_path)

cursor = conn.cursor()

cursor.execute('SELECT name, applied\_on, status FROM patches WHERE id = ?', (patch\_id,))

patch = cursor.fetchone()

conn.close()

if not patch:

log.error(f"No patch found with ID: {patch\_id}")

print(f"No patch found with ID: {patch\_id}")

return

patch\_name, applied\_on, status = patch

log.info(f"Monitoring Patch: {patch\_name}, Applied On: {applied\_on}, Status: {status}")

# Placeholder for monitoring logic

if status.lower() == 'applied':

log.info(f"Patch ID {patch\_id} has been applied successfully. No issues detected.")

print(f"Patch ID {patch\_id} has been applied successfully. No issues detected.")

else:

log.warning(f"Patch ID {patch\_id} status: {status}. Further investigation required.")

print(f"Patch ID {patch\_id} status: {status}. Further investigation required.")

except Exception as e:

log.error(f"Error monitoring patch effects for patch ID {patch\_id}: {e}")

print(f"Error monitoring patch effects for patch ID {patch\_id}: {e}")

# Main function to interact with the system

def run\_patch\_management\_system():

# Initialize logger and database

log = create\_logger('logs/patch\_management.log')

initialize\_db()

# Ensure PSWindowsUpdate module is installed

try:

install\_pswindowsupdate\_module(log)

except Exception as e:

print("Failed to install PSWindowsUpdate module. Check logs for details.")

return

while True:

print("\nPatch Management System")

print("-----------------------")

print("1. Create Patch Record")

print("2. View Patch Records")

print("3. Update Patch Record")

print("4. Delete Patch Record")

print("5. Apply System Patch")

print("6. Monitor Patch Effects")

print("7. Exit")

choice = input("Enter your choice (1-7): ").strip()

if choice == '1':

name = input("Enter patch name: ").strip()

description = input("Enter patch description: ").strip()

create\_patch(name, description)

log.info(f"Created patch record: {name} - {description}")

elif choice == '2':

patches = read\_patches()

if patches:

print("\nPatch Records:")

print("{:<5} {:<30} {:<50} {:<15} {:<20}".format("ID", "Name", "Description", "Status", "Applied On"))

print("-" \* 120)

for patch in patches:

print("{:<5} {:<30} {:<50} {:<15} {:<20}".format(

patch[0], patch[1], patch[2], patch[3], patch[4] if patch[4] else "N/A"))

else:

print("No patch records found.")

elif choice == '3':

patch\_id = input("Enter patch ID to update: ").strip()

print("Leave fields blank if no change is needed.")

name = input("Enter new patch name: ").strip()

description = input("Enter new patch description: ").strip()

status = input("Enter new status (pending/applied/failed): ").strip()

update\_patch(patch\_id, name=name or None, description=description or None, status=status or None)

log.info(f"Updated patch ID {patch\_id}")

elif choice == '4':

patch\_id = input("Enter patch ID to delete: ").strip()

confirm = input(f"Are you sure you want to delete patch ID {patch\_id}? (y/n): ").strip().lower()

if confirm == 'y':

delete\_patch(patch\_id)

log.info(f"Deleted patch ID {patch\_id}")

else:

print("Deletion cancelled.")

elif choice == '5':

patch\_id = input("Enter patch ID to apply: ").strip()

apply\_system\_patches(patch\_id, log)

elif choice == '6':

patch\_id = input("Enter patch ID to monitor: ").strip()

monitor\_patch\_effects(patch\_id, log)

elif choice == '7':

print("Exiting Patch Management System.")

break

else:

print("Invalid choice. Please try again.")

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

run\_patch\_management\_system()