Vault Utilities Module

import sqlite3  
import os  
import base64  
from azure.identity import DefaultAzureCredential  
from azure.keyvault.secrets import SecretClient  
from Crypto.Cipher import AES  
from Crypto.Util.Padding import unpad  
  
DB\_PATH = "legacy\_vault.db"  
KEY\_VAULT\_NAME = "your-keyvault-name"  
SECRET\_NAME = "encryption-key"  
vault\_url = f"https://{KEY\_VAULT\_NAME}.vault.azure.net"  
credential = DefaultAzureCredential()  
secret\_client = SecretClient(vault\_url, credential)  
  
def delete\_vault\_record(vault\_id: str) -> str:  
 conn = sqlite3.connect(DB\_PATH)  
 cursor = conn.cursor()  
 cursor.execute("SELECT vault\_id FROM legacy\_vault WHERE vault\_id = ?", (vault\_id,))  
 if not cursor.fetchone():  
 conn.close()  
 return f"Error: No record found for Vault ID {vault\_id}"  
 cursor.execute("DELETE FROM legacy\_vault WHERE vault\_id = ?", (vault\_id,))  
 conn.commit()  
 conn.close()  
 return f"Vault record {vault\_id} permanently deleted."  
  
def decrypt\_data(encrypted\_data: str, key: str) -> str:  
 encrypted\_bytes = base64.b64decode(encrypted\_data)  
 cipher = AES.new(key.encode("utf-8"), AES.MODE\_ECB)  
 decrypted\_bytes = unpad(cipher.decrypt(encrypted\_bytes), AES.block\_size)  
 return decrypted\_bytes.decode("utf-8")  
  
def export\_vault\_record(vault\_id: str):  
 conn = sqlite3.connect(DB\_PATH)  
 cursor = conn.cursor()  
 cursor.execute("SELECT vault\_id, title, description FROM legacy\_vault WHERE vault\_id = ?", (vault\_id,))  
 record = cursor.fetchone()  
 conn.close()  
 if not record:  
 return f"Error: No record found for Vault ID {vault\_id}"  
 encryption\_key = secret\_client.get\_secret(SECRET\_NAME).value  
 decrypted\_description = decrypt\_data(record[2], encryption\_key)  
 os.makedirs("exports", exist\_ok=True)  
 filename = f"exports/{record[1].replace(' ', '\_')}.txt"  
 with open(filename, "w", encoding="utf-8") as file:  
 file.write(f"Vault ID: {record[0]}\n")  
 file.write(f"Title: {record[1]}\n")  
 file.write(f"Description:\n{decrypted\_description}\n")  
 return f"Vault record {vault\_id} exported successfully as {filename}"  
  
def fetch\_active\_vault\_records():  
 conn = sqlite3.connect(DB\_PATH)  
 cursor = conn.cursor()  
 cursor.execute("PRAGMA table\_info(legacy\_vault)")  
 columns = [col[1] for col in cursor.fetchall()]  
 if "deleted" in columns:  
 cursor.execute("SELECT vault\_id, title, created\_at FROM legacy\_vault WHERE deleted IS NULL OR deleted = 0 ORDER BY created\_at DESC")  
 else:  
 cursor.execute("SELECT vault\_id, title, created\_at FROM legacy\_vault ORDER BY created\_at DESC")  
 records = cursor.fetchall()  
 conn.close()  
 return [{"vault\_id": r[0], "title": r[1], "created\_at": r[2]} for r in records]  
  
def restore\_vault\_record(vault\_id: str) -> str:  
 conn = sqlite3.connect(DB\_PATH)  
 cursor = conn.cursor()  
 cursor.execute("SELECT vault\_id FROM legacy\_vault WHERE vault\_id = ? AND deleted = 1", (vault\_id,))  
 if not cursor.fetchone():  
 conn.close()  
 return f"Error: No soft-deleted record found for Vault ID {vault\_id}"  
 cursor.execute("UPDATE legacy\_vault SET deleted = 0, deleted\_at = NULL WHERE vault\_id = ?", (vault\_id,))  
 conn.commit()  
 conn.close()  
 return f"Vault record {vault\_id} restored successfully."  
  
def lock\_vault\_record(vault\_id: str) -> str:  
 conn = sqlite3.connect(DB\_PATH)  
 cursor = conn.cursor()  
 cursor.execute("PRAGMA table\_info(legacy\_vault)")  
 columns = [col[1] for col in cursor.fetchall()]  
 if "locked" not in columns:  
 cursor.execute("ALTER TABLE legacy\_vault ADD COLUMN locked INTEGER DEFAULT 0")  
 conn.commit()  
 cursor.execute("SELECT vault\_id FROM legacy\_vault WHERE vault\_id = ?", (vault\_id,))  
 if not cursor.fetchone():  
 conn.close()  
 return f"Error: No record found for Vault ID {vault\_id}"  
 cursor.execute("UPDATE legacy\_vault SET locked = 1 WHERE vault\_id = ?", (vault\_id,))  
 conn.commit()  
 conn.close()  
 return f"Vault record {vault\_id} locked successfully."  
  
def unlock\_vault\_record(vault\_id: str) -> str:  
 conn = sqlite3.connect(DB\_PATH)  
 cursor = conn.cursor()  
 cursor.execute("PRAGMA table\_info(legacy\_vault)")  
 columns = [col[1] for col in cursor.fetchall()]  
 if "locked" not in columns:  
 cursor.execute("ALTER TABLE legacy\_vault ADD COLUMN locked INTEGER DEFAULT 0")  
 conn.commit()  
 cursor.execute("SELECT vault\_id FROM legacy\_vault WHERE vault\_id = ?", (vault\_id,))  
 if not cursor.fetchone():  
 conn.close()  
 return f"Error: No record found for Vault ID {vault\_id}"  
 cursor.execute("UPDATE legacy\_vault SET locked = 0 WHERE vault\_id = ?", (vault\_id,))  
 conn.commit()  
 conn.close()  
 return f"Vault record {vault\_id} unlocked successfully."