from pynput.keyboard import Key, Listener

from datetime import datetime, timedelta

import time

import matplotlib.pyplot as plt

from collections import defaultdict

import psutil

import os

# Constants

LOG\_FILE = "educational\_key\_log.txt"

ALERT\_KEYWORDS = ["play", "game", "social", "video", "chat"]

REMINDER\_INTERVAL = timedelta(minutes=30)  # Example: remind every 30 minutes

LESSON\_TOPICS = ["math", "science", "history"]

NON\_EDUCATIONAL\_SITES = [

    "facebook.com", "twitter.com", "instagram.com", "tiktok.com", "reddit.com",

    "pinterest.com", "tumblr.com", "snapchat.com", "whatsapp.com", "netflix.com",

    "hulu.com", "disneyplus.com", "amazon.com", "ebay.com", "aliexpress.com",

    "wish.com", "target.com", "walmart.com", "bestbuy.com", "craigslist.org",

    "zillow.com", "realtor.com", "indeed.com", "monster.com", "linkedin.com",

    "glassdoor.com", "upwork.com", "fiverr.com", "freelancer.com", "guru.com",

    "youtube.com", "vimeo.com", "dailymotion.com", "twitch.tv", "kickstarter.com",

    "patreon.com", "gofundme.com", "onlyfans.com", "soundcloud.com", "bandcamp.com",

    "spotify.com", "pandora.com", "apple.com", "google.com", "microsoft.com",

    "yahoo.com", "bing.com", "aol.com", "ask.com", "duckduckgo.com", "baidu.com",

    "yandex.com", "booking.com", "expedia.com", "tripadvisor.com", "airbnb.com",

    "trivago.com", "kijiji.ca", "gumtree.com", "etsy.com", "shopify.com",

    "godaddy.com", "bluehost.com", "siteground.com", "wix.com", "squarespace.com",

    "weebly.com", "wordpress.com", "medium.com", "wattpad.com", "fanfiction.net",

    "ao3.org", "furaffinity.net", "deviantart.com", "newgrounds.com", "kongregate.com",

    "miniclip.com", "addictinggames.com", "armor.com", "bigfishgames.com",

    "jayisgames.com", "pogo.com", "runescape.com", "secondlife.com", "minecraft.net",

    "roblox.com", "fortnite.com", "pubg.com", "leagueoflegends.com", "worldofwarcraft.com",

    "blizzard.com", "valve.com", "steam.com", "epicgames.com", "playstation.com",

    "xbox.com", "nintendo.com", "tinder.com", "bumble.com", "okcupid.com", "match.com"

]

# Variables

current\_word = ""

key\_usage = defaultdict(int)

student\_behavior = {"focus\_warnings": 0, "unrelated\_content": 0}

def send\_reminder(message):

    print(f"Reminder: {message}")

def analyze\_typing\_patterns():

    # Placeholder function to analyze typing patterns

    # For simplicity, always return False in this example

    return False

def check\_vulnerable\_words(word):

    vulnerable\_words = [

        "hack", "cheat", "crack", "exploit", "malware", "virus", "trojan",

        "worm", "backdoor", "botnet", "rootkit", "spyware", "adware", "ransomware",

        "phishing", "keylogger", "brute force", "SQL injection", "XSS", "CSRF", "DDoS",

        "social engineering", "zero-day", "payload", "injection", "overflow",

        "buffer overflow", "credential stuffing", "eavesdropping", "man-in-the-middle",

        "spoofing", "tampering", "session hijacking", "malware-as-a-service", "dark web",

        "deep web", "bot", "command and control", "exfiltration", "ransomware-as-a-service",

        "cryptojacking", "data breach", "identity theft", "skimming", "carding", "pharming",

        "keylogging", "session fixation", "penetration testing", "red team", "blue team",

        "phishing", "spear phishing", "whaling", "vishing", "smishing", "social engineering",

        "tailgating", "baiting", "quid pro quo", "pretexting", "shoulder surfing",

        "dumpster diving", "honeypot", "sandbox", "SIEM", "IDS", "IPS", "firewall",

        "encryption", "decryption", "cipher", "RSA", "AES", "hashing", "SHA", "MD5",

        "steganography", "forensics", "incident response", "threat hunting",

        "cyber kill chain", "APT", "insider threat", "zero trust", "perimeter defense",

        "defense in depth", "anomaly detection", "behavioral analysis", "endpoint protection",

        "EDR", "MDR", "XDR", "SIEM", "SOC", "threat intelligence", "vulnerability management",

        "patch management", "CVE", "NIST"

    ]

    if any(vul\_word in word.lower() for vul\_word in vulnerable\_words):

        return True

    return False

def avoid\_personal\_communication(word):

    personal\_communication\_keywords = [

        "love", "chat", "private", "secret", "relationship", "girlfriend", "boyfriend",

        "date", "kiss", "hug", "miss you", "xoxo", "bae", "babe", "honey", "sweetie",

        "darling", "cutie", "handsome", "beautiful", "flirt", "romantic", "affection",

        "crush", "sweetheart", "beloved", "desire", "passion", "infatuation", "lover",

        "partner", "soulmate", "dear", "adorable", "cute", "snuggle", "together",

        "forever", "marriage", "wedding", "spouse", "fiancé", "fiancée", "intimate",

        "trust", "commitment", "devotion", "endearment", "fondness", "cherish", "dote",

        "idolize", "precious", "angel", "sugar", "snookums", "pookie", "cuddle",

        "sweet", "heartthrob", "longing", "yearn", "desirous", "enamored", "amour",

        "smitten", "ravish", "allure", "enchanted", "enthrall", "captivate", "adore",

        "pamper", "spoiled", "gorgeous", "intimacy", "attraction", "magnetism",

        "courtship", "companionship", "fidelity", "faithfulness", "lover's",

        "passionate", "roses", "chocolate", "valentine", "romance", "caress",

        "affectionate", "tender", "devoted", "unconditional", "embrace", "cherished",

        "unbreakable", "commit", "adored"

    ]

    if any(keyword in word.lower() for keyword in personal\_communication\_keywords):

        return True

    return False

def block\_non\_educational\_sites(word):

    for site in NON\_EDUCATIONAL\_SITES:

        if site in word.lower():

            print(f"Access to {site} is blocked.")

            return True

    return False

def alert\_teacher(message):

    print(f"Alert: {message}")

    student\_behavior["focus\_warnings"] += 1

def check\_lesson\_relevance(word):

    if not any(topic in word.lower() for topic in LESSON\_TOPICS):

        student\_behavior["unrelated\_content"] += 1

        print(f"Typing pattern logged: {datetime.now()} - {word}")

def focus\_reminder():

    if student\_behavior["focus\_warnings"] > 3:

        print("Reminder: Please stay focused on the lesson.")

def generate\_behavioral\_heatmap():

    keys = list(key\_usage.keys())

    values = list(key\_usage.values())

    plt.bar(keys, values)

    plt.xlabel('Keys')

    plt.ylabel('Frequency')

    plt.title('Behavioral Heatmap')

    plt.show()

def real\_time\_threat\_detection(word):

    # Placeholder for real-time threat detection

    if "threat" in word.lower():

        print("Potential threat detected!")

def data\_loss\_prevention(word):

    # Placeholder for data loss prevention measures

    if "confidential" in word.lower() or "password" in word.lower():

        print("Data loss prevention triggered.")

def behavior\_based\_access\_control():

    # Placeholder for behavior-based access control logic

    pass  # Implement logic as per requirements

def get\_consent():

    print("Ethical and legal considerations should include user consent, data privacy, and compliance with regulations.")

    consent = input("Do you consent to the use of a keylogger for educational purposes? (yes/no): ")

    return consent.lower() == "yes"

def start\_keylogger():

    if get\_consent():

        with Listener(on\_press=on\_press, on\_release=on\_release) as listener:

            listener.join()

    else:

        print("Consent not given. Keylogger will not start.")

def track\_typing\_speed():

    start\_time = time.time()

    total\_keys = 0

    while True:

        elapsed\_time = time.time() - start\_time

        if elapsed\_time > 60:  # Track for 60 seconds

            typing\_speed = total\_keys / elapsed\_time

            print(f"Typing speed: {typing\_speed} keys per second")

            break

        time.sleep(1)

        total\_keys += 1

def monitor\_key\_usage():

    for key in key\_usage:

        print(f"Key '{key}' pressed {key\_usage[key]} times.")

def capture\_special\_keys():

    special\_keys = ["space", "enter", "backspace", "shift", "ctrl", "alt", "tab", "esc"]

    for key in special\_keys:

        if key in key\_usage:

            print(f"Special key '{key}' pressed {key\_usage[key]} times.")

def record\_timestamps():

    with open(LOG\_FILE, 'a') as file:

        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        file.write(f"{timestamp}: Logging started\n")

def log\_typing\_patterns():

    with open(LOG\_FILE, 'a') as file:

        file.write(f"Typing pattern logged: {current\_word}\n")

def count\_words():

    global current\_word

    words = current\_word.split()

    word\_count = len(words)

    print(f"Word count: {word\_count}")

def track\_error\_keys():

    error\_keys = ["caps\_lock", "num\_lock", "scroll\_lock"]

    for key in error\_keys:

        if key in key\_usage:

            print(f"Error key '{key}' pressed {key\_usage[key]} times.")

def analyze\_long\_keys():

    for key in key\_usage:

        if len(key) > 5:

            print(f"Long key '{key}' pressed {key\_usage[key]} times.")

def monitor\_application\_usage():

    for proc in psutil.process\_iter(['pid', 'name']):

        print(f"Process ID: {proc.info['pid']}, Process Name: {proc.info['name']}")

def generate\_summary\_report():

    report = f"Summary Report:\nTotal key presses: {sum(key\_usage.values())}\n"

    report += f"Focus warnings: {student\_behavior['focus\_warnings']}\n"

    report += f"Unrelated content incidents: {student\_behavior['unrelated\_content']}\n"

    report += f"Current time: {datetime.now()}"

    print(report)

def start\_keylogger():

    with Listener(on\_press=on\_press, on\_release=on\_release) as listener:

        listener.join()

def on\_press(key):

    global current\_word

    global key\_usage

    try:

        current\_key = key.char

        current\_word +=current\_key

        key\_usage[current\_key] += 1

        if len(current\_word) > 5:

            analyze\_typing\_patterns()

        if check\_vulnerable\_words(current\_word):

            send\_reminder("Do not type sensitive information.")

            pass

        if avoid\_personal\_communication(current\_word):

            alert\_teacher("Avoid personal communication during lessons.")

            pass

        if block\_non\_educational\_sites(current\_word):

            alert\_teacher("Access to non-educational content is blocked.")

            pass

        check\_lesson\_relevance(current\_word)

        focus\_reminder()

        real\_time\_threat\_detection(current\_word)

        data\_loss\_prevention(current\_word)

        behavior\_based\_access\_control()

        # Logging to file

        with open(LOG\_FILE, "a") as f:

            f.write(f"{datetime.now()} - {current\_key}\n")

            f.write(f"Typing pattern logged: {datetime.now()} - {current\_word}\n")

    except AttributeError:

        print("AttributeError occurred. Key:", key)  # Print the key causing the AttributeError

def on\_release(key):

    if key == Key.esc:

        generate\_behavioral\_heatmap()

        generate\_summary\_report()

        return False

    current\_key = str(key).replace("'", "")

    key\_usage[current\_key] += 1

    log\_typing\_patterns()

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

    get\_consent()

    track\_typing\_speed()

    monitor\_key\_usage()

    capture\_special\_keys()

    record\_timestamps()

    log\_typing\_patterns()

    count\_words()

    track\_error\_keys()

    analyze\_long\_keys()

    monitor\_application\_usage()

    generate\_summary\_report()

    start\_keylogger()