```
"""
بـوت شذا لـلـتـداول 💙
ICT/SMC - DBOS + IDM + OB
نـسخة شخصية - بـالـلـهجة الـسعـوديـة
"""

import asyncio
import os
import logging
import requests
import random
from datetime import datetime, timedelta
import pytz
import yfinance as yf
import pandas as pd
from telegram import Bot
from telegram.ext import Application, CommandHandler

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

TELEGRAM_TOKEN = os.environ.get("TELEGRAM_TOKEN", "YOUR_TOKEN_HERE")
CHAT_ID = os.environ.get("CHAT_ID", "YOUR_CHAT_ID_HERE")
RIYADH_TZ = pytz.timezone('Asia/Riyadh')

# ==================== إعدادات الحساب ====================
ACCOUNT = {
    "balance": 5000.0,
    "max_drawdown": 10.0,
    "daily_drawdown": 5.0,
    "drawdown_used": 0.0,
    "daily_used": 0.0,
    "trades_week": 0,
    "pnl_percent": 0.0,
}

SYMBOLS = {
    "XAUUSD": "GC=F",
    "XAGUSD": "SI=F",
    "EURUSD": "EURUSD=X",
    "GBPUSD": "GBPUSD=X",
    "BTCUSD": "BTC-USD",
    "USDCHF": "USDCHF=X",
    "USDJPY": "USDJPY=X",
```

```
    "AUDUSD": "AUDUSD=X",
}

HIGH_IMPACT_KEYWORDS = [
    "Fed", "Federal Reserve", "FOMC", "Interest Rate",
    "CPI", "NFP", "Non-Farm", "GDP", "Powell", "ECB", "BOE", "BOJ"
]

# ==================== رسائل الانتظار ====================
WAITING_MSGS = [
    "🔍 جالس أفحص الأسواق لك.. لحظة صبر يا بطلة",
    "👀 عيني على الشارت، لحظة وأخبرك",
    "⏳ البحث مستمر، السوق مو دايم يعطي فرص، بس أنا صاحي 💪",
    "🧐 فاحص كل زوج بعين.. لا شي يفوتني",
]

NO_SETUP_MSGS = [
    "😁 ما لقيت سيتاب يستاهل الحين. دبر عمرك بشغلة ثانية وأنا أراقب لك",
    "👀 السوق هادي الحين، ما في فرصة تستاهل. روحي اتقهوي وأنا هنا 😴",
    "💙 فحصت كل شي، ما في سيتاب بشروطنا الحين. الصبر مفتاح، والفرص تجي",
    "⏸️ السوق مو متحرك على شروطنا الحين. ما تدخلين بدون سيتاب صح، هذا اللي علمناه",
    "💙 هدوء في الأسواق الحين. استغلي الوقت تحللين أو تستريحين، وأنا أراقب 🌙",
]

STATUS_MSGS = [
    "🔍 جالس أبحث لك عن سيتاب.. عيني على الشارت",
    "📊 أفحص الأزواج واحد واحد، لو في شي أنبهك فوراً",
    "💙 صاحي ومراقب، لا تقلقين 👁️",
    "⚡ شغّال بكامل طاقتي، ما شي يفوتني إن شاء الله",
]

DAILY_TIPS = [
    "👑 ما في صفقة تستاهل تخلك تكسري خطتك. الخطة هي الملك",
    "🎯 السوينق يحتاج صبر. الصفقة الصح تجيك، ما تروحين إليها",
    "🧘 الخسارة جزء من التداول. المهم إدارة المخاطرة مو الربح السريع",
    "⛔ أي ضغط داخل الصفقة؟ هذا إشارة توقفين مو تكملين",
    "🏆 الفرق بين المحترف والمبتدئ مو في الصفقات، في الانضباط",
    "📝 اكتبي كل صفقة في الجورنال. اللي ما يوثق، ما يتعلم",
    "💙 لو حسيتِ بالثقل من السوق، خذي استراحة. الحساب أهم من الصفقة",
]

# ==================== الأخبار ====================

def check_news():
    try:
        r = requests.get("https://nfs.faireconomy.media/ff_calendar_thisweek.json
        if r.status_code != 200:
```

```
            return {"has_news": False, "events": []}
        now = datetime.utcnow()
        upcoming = []
        for ev in r.json():
            try:
                if ev.get("impact") != "High":
                    continue
                t = datetime.fromisoformat(ev.get("date","").replace("Z",""))
                diff = t - now
                if timedelta(hours=-1) <= diff <= timedelta(hours=24):
                    title = ev.get("title","")
                    if any(k.lower() in title.lower() for k in HIGH_IMPACT_KEYWOR
                        upcoming.append({
                            "title": title,
                            "currency": ev.get("country",""),
                            "hours": round(diff.total_seconds()/3600, 1)
                        })
            except:
                continue
        return {"has_news": len(upcoming)>0, "events": upcoming[:3]}
    except:
        return {"has_news": False, "events": []}

# ==================== البيانات والتحليل ====================

def get_candles(yf_sym, tf, limit=100):
    try:
        period = {"1h":"7d","4h":"60d","1d":"180d","1wk":"2y"}.get(tf,"60d")
        df = yf.Ticker(yf_sym).history(period=period, interval=tf)
        df = df.rename(columns={'Open':'open','High':'high','Low':'low','Close':'
        return df.tail(limit)
    except:
        return pd.DataFrame()

def detect_trend(df):
    if len(df) < 20:
        return "neutral"
    r = df.tail(20)
    if r['high'].iloc[-1] > r['high'].iloc[0] and r['low'].iloc[-1] > r['low'].il
        return "bullish"
    if r['high'].iloc[-1] < r['high'].iloc[0] and r['low'].iloc[-1] < r['low'].il
        return "bearish"
    return "neutral"

def find_swings(df, lb=3):
    highs, lows = [], []
    for i in range(lb, len(df)-lb):
```

```
        if df['high'].iloc[i] == df['high'].iloc[i-lb:i+lb+1].max():
            highs.append((i, df['high'].iloc[i]))
        if df['low'].iloc[i] == df['low'].iloc[i-lb:i+lb+1].min():
            lows.append((i, df['low'].iloc[i]))
    return highs, lows

def detect_dbos(df, highs, lows, direction):
    if direction == "bullish" and len(highs) >= 2:
        for i in range(len(highs)-1, 0, -1):
            if highs[i][1] > highs[i-1][1]:
                for j in range(highs[i-1][0], len(df)):
                    if df['close'].iloc[j] > highs[i-1][1]:
                        return {'index': j, 'price': highs[i-1][1]}
    elif direction == "bearish" and len(lows) >= 2:
        for i in range(len(lows)-1, 0, -1):
            if lows[i][1] < lows[i-1][1]:
                for j in range(lows[i-1][0], len(df)):
                    if df['close'].iloc[j] < lows[i-1][1]:
                        return {'index': j, 'price': lows[i-1][1]}
    return None

def find_idm(df, idx, direction):
    for i in range(idx+1, min(idx+25, len(df))):
        if direction == "bullish":
            if df['close'].iloc[i] < df['open'].iloc[i] and df['low'].iloc[i] < d
                return {'index': i, 'price': df['low'].iloc[i]}
        else:
            if df['close'].iloc[i] > df['open'].iloc[i] and df['high'].iloc[i] >
                return {'index': i, 'price': df['high'].iloc[i]}
    return None

def find_ob(df, idx, direction):
    if not idx or idx < 2:
        return None
    for i in range(idx, max(idx-15,0), -1):
        c = df.iloc[i]
        if direction == "bullish" and c['close'] < c['open']:
            return {'top': c['open'], 'bottom': c['close']}
        elif direction == "bearish" and c['close'] > c['open']:
            return {'top': c['close'], 'bottom': c['open']}
    return None

def check_sweep(df, direction):
    if len(df) < 15:
        return False
    rh = df['high'].tail(15).iloc[:-2].max()
    rl = df['low'].tail(15).iloc[:-2].min()
```

```
    last = df.iloc[-2]
    if direction == "bullish":
        return last['low'] < rl and df['close'].iloc[-1] > rl
    return last['high'] > rh and df['close'].iloc[-1] < rh

def calc_quality(dbos, idm, ob, sweep, daily_match, has_news):
    score = 0
    if dbos: score += 25
    if idm:  score += 25
    if ob:   score += 25
    if sweep: score += 15
    if daily_match: score += 10
    if has_news: score -= 15
    return max(0, min(100, score))

def get_risk_advice(quality, account):
    dd = account['drawdown_used']
    daily = account['daily_used']
    remaining_dd = account['max_drawdown'] - dd
    remaining_daily = account['daily_drawdown'] - daily
    pnl = account['pnl_percent']

    # وضع الحساب خسارة
    if pnl <= -7:
        return 0, "⛔ لا تدخلين أي صفقة، استريحي وراجعي استراتيجيتك في وضع صعب."
    if remaining_dd <= 1 or remaining_daily <= 0.5:
        return 0, "⛔ توقفي اليوم، الحساب أهم من أي صفقة! الدرودون ضيق جداً"
    if remaining_dd <= 3:
        max_risk = 0.5
        note = f"⚠️ باقي {remaining_dd:.1f}% درودون، تعاملي بحذر شديد"
    elif quality >= 90:
        max_risk = 2.0
        note = "🔥 سيتاب ممتاز، تستاهل المخاطرة الكاملة"
    elif quality >= 75:
        max_risk = 1.5
        note = "💪 سيتاب قوي، المخاطرة كويسة"
    elif quality >= 60:
        max_risk = 1.0
        note = "✅ سيتاب معقول، مخاطرة محافظة"
    else:
        return 0, "❌ الجودة ضعيفة، ما ندخل"

    # تعديل لو الحساب رابح
    if pnl >= 5:
        note += f"\n💰 ماشي كويس +{pnl}%! حافظي على مكاسبك"

    return max_risk, note
```

```
def analyze(sym_name, yf_sym, tf, news):
    df = get_candles(yf_sym, tf)
    if df.empty or len(df) < 30:
        return None
    trend = detect_trend(df)
    if trend == "neutral":
        return None
    highs, lows = find_swings(df)
    dbos = detect_dbos(df, highs, lows, trend)
    if not dbos: return None
    idm = find_idm(df, dbos['index'], trend)
    if not idm: return None
    ob = find_ob(df, idm['index'], trend)
    if not ob: return None

    current = df['close'].iloc[-1]
    ob_range = ob['top'] - ob['bottom']
    in_ob = (ob['bottom'] - ob_range*0.3) <= current <= (ob['top'] + ob_range*0.3
    sweep = check_sweep(df, trend)

    df_d = get_candles(yf_sym, "1d", 30)
    daily_match = detect_trend(df_d) == trend if not df_d.empty else False

    quality = calc_quality(dbos, idm, ob, sweep, daily_match, news['has_news'])
    if quality < 60:
        return None

    return {
        'symbol': sym_name, 'tf': tf, 'trend': trend,
        'current': current, 'ob_top': ob['top'], 'ob_bottom': ob['bottom'],
        'in_ob': in_ob, 'sweep': sweep, 'daily_match': daily_match,
        'quality': quality, 'news': news,
    }

# ==================== الرسائل ====================

def setup_msg(a):
    arrow = "📈" if a['trend'] == "bullish" else "📉"
    direction = "شراء" if a['trend'] == "bullish" else "بيع"
    risk, risk_note = get_risk_advice(a['quality'], ACCOUNT)

    news_txt = ""
    if a['news']['has_news']:
        news_txt = "\n⚠️ تنبيه: في أخبار مهمة قريبة!\n"
        for ev in a['news']['events']:
            news_txt += f"   • {ev['title']} ({ev['currency']}) بعد {ev['hours']}
```

```
        news_txt += "خذي بالك وخففي المخاطرة 🙏\n"

    extras = []
    if a['sweep']: extras.append("✅ سحب سيولة قبل الحركة 🎯")
    if a['daily_match']: extras.append("✅ اليومي يدعم النظرة 💪")
    extras_txt = "\n".join(extras)

    zone_txt = "🎯 السعر داخل الـ OB لا تفوتينها الحين! 👀" if a['in_ob'] else \
               f"⏳ انتظري السعر يوصل للمنطقة:\n   {a['ob_bottom']:.4f} ← {a['ob_

    quality_bar = "🟢" * (a['quality']//20) + "⚪" * (5 - a['quality']//20)

    if risk == 0:
        risk_txt = f"⛔ {risk_note}"
    else:
        risk_txt = f"💰 المخاطرة المقترحة: {risk}%\n{risk_note}"

    return f"""
{'🔥'*4} سيتاب {direction} | {a['symbol']} {'🔥'*4}
{arrow} فريم: {a['tf']}
{'—'*32}
✅ DBOS - كسر هيكل مزدوج
✅ IDM - أول بول باك
✅ OB - أوردر بلوك جاهز
{extras_txt}
{news_txt}
💰 السعر الحالي: {a['current']:.4f}
{zone_txt}

⭐ جودة السيتاب: {a['quality']}/100
{quality_bar}

{risk_txt}

ذكريكِ شذا 💙
القرار النهائي إلك، هذا تنبيه مو توصية
{'—'*32}"""

def daily_advice_msg():
    dd = ACCOUNT['drawdown_used']
    daily = ACCOUNT['daily_used']
    trades = ACCOUNT['trades_week']
    pnl = ACCOUNT['pnl_percent']
    remaining = ACCOUNT['max_drawdown'] - dd

    if pnl > 0:
        pnl_txt = f"✅ الحساب رابح {pnl:.1f}% 🎉 واصلي بنفس المنهج"
```

```
    elif pnl == 0:
        pnl_txt = "➡️ الحساب عند نقطة البداية، ركزي على الجودة"
    elif pnl >= -5:
        pnl_txt = f"⚠️ الحساب خاسر {abs(pnl):.1f}%، خففي المخاطرة وما تتسرعين"
    else:
        pnl_txt = f"🔴 الحساب خاسر {abs(pnl):.1f}%! اية الحساب مو استرداد الخسارة

    if dd == 0:
        dd_txt = "🌱 الحساب طازج ما استخدمتِ شي"
    elif remaining >= 7:
        dd_txt = f"💪 كثير الحمدلله {remaining:.1f}% باقي، استخدمتِ {dd:.1f}%"
    elif remaining >= 4:
        dd_txt = f"⚠️ باقي {remaining:.1f}% دروداون، تعاملي بحذر"
    else:
        dd_txt = f"🔴 باقي {remaining:.1f}% قصوى عناية يحتاج الحساب بس!"

    if trades == 0:
        trades_txt = "🥇 انتظري السيتاب الصح الصبر ذهب ما دخلتِ صفقات،"
    elif trades <= 2:
        trades_txt = f"👍 صفقة، ممتاز! السوينق مو يحتاج كثير دخلتِ {trades}"
    else:
        trades_txt = f"⚠️ {trades} خففي للسوينق، شوي كثير صفقات الأسبوع،"

    return f"""
☀️ نصايح اليوم من بوتك 💙
{'—'*32}

1️⃣ وضع الحساب:
{pnl_txt}

2️⃣ الدروداون:
{dd_txt}
الديلي المستخدم: {daily:.1f}%

3️⃣ الصفقات هالأسبوع:
{trades_txt}

4️⃣ نصيحة اليوم:
{random.choice(DAILY_TIPS)}

🌟 وفقك الله شذا
{'—'*32}"""

def status_msg():
    now = datetime.now(RIYADH_TZ)
    pnl = ACCOUNT['pnl_percent']
    pnl_emoji = "📈" if pnl >= 0 else "📉"
```

```
    return f"""
{random.choice(STATUS_MSGS)}

⏰ بتوقيت الرياض {now.strftime('%H:%M')}
{pnl_emoji} الحساب: {'+' if pnl >= 0 else ''}{pnl:.1f}%
📉 دروداون مستخدم: {ACCOUNT['drawdown_used']:.1f}%
📊 صفقات الأسبوع: {ACCOUNT['trades_week']}
"""

# ==================== الحلقة الرئيسية ====================

async def scan_markets(bot):
    news = check_news()
    found = []
    for name, yf_sym in SYMBOLS.items():
        for tf in ["4h", "1h"]:
            try:
                r = analyze(name, yf_sym, tf, news)
                if r:
                    found.append(r)
            except Exception as e:
                logger.error(f"خطأ {name} {tf}: {e}")

    if found:
        found.sort(key=lambda x: x['quality'], reverse=True)
        for s in found:
            await bot.send_message(chat_id=CHAT_ID, text=setup_msg(s))
            await asyncio.sleep(2)
        return True
    return False

async def trading_loop(bot):
    await bot.send_message(chat_id=CHAT_ID,
        text="🚀 بوتك شغّال يا شذا!\n"
             "يفحص كل ساعة وينبهك بأي سيتاب 💙\n"
             "النصايح اليومية الساعة 8 صباحاً\n"
             "وكل 4 ساعات أخبرك وش أسوي 😁\n\n"
             "الأوامر:\n"
             "/scan - فحص فوري\n"
             "/advice - نصايح اليوم\n"
             "/update - تحديث الحساب\n"
             "/status - وش أسوي الحين")

    last_advice_day = None
    last_status_hour = -1
    scan_count = 0
```

```
    while True:
        try:
            now = datetime.now(RIYADH_TZ)
            today = now.date()

            # نصايح يومية الساعة 8 صباحاً
            if now.hour == 8 and now.minute < 5 and last_advice_day != today:
                await bot.send_message(chat_id=CHAT_ID, text=daily_advice_msg())
                last_advice_day = today

            # رسالة الحالة كل 4 ساعات
            if now.hour % 4 == 0 and now.hour != last_status_hour and now.minute
                found = await scan_markets(bot)
                if not found:
                    await bot.send_message(chat_id=CHAT_ID,
                        text=random.choice(NO_SETUP_MSGS))
                last_status_hour = now.hour
                scan_count += 1
            else:
                # فحص عادي كل ساعة بدون رسالة انتظار
                await scan_markets(bot)

            await asyncio.sleep(3600)

        except Exception as e:
            logger.error(f"خطأ: {e}")
            await asyncio.sleep(60)

# ==================== أوامر التيليغرام ====================

async def start_cmd(update, context):
    await update.message.reply_text(
        "🚀 أهلاً شذا!\n"
        "أنا بوتك للتداول، أراقب الأسواق ٢٤/٧ 💙\n\n"
        "الأوامر:\n"
        "/scan - فحص فوري للأسواق\n"
        "/advice - نصايح اليوم\n"
        "/status - وش أسوي الحين\n"
        "/update - تحديث وضع حسابك\n\n"
        "ابدئي بـ /update عشان أعرف وضع حسابك 💪"
    )

async def scan_cmd(update, context):
    await update.message.reply_text(random.choice(WAITING_MSGS))
    found = await scan_markets(context.bot)
    if not found:
        await update.message.reply_text(random.choice(NO_SETUP_MSGS))
```

```
async def advice_cmd(update, context):
    await update.message.reply_text(daily_advice_msg())

async def status_cmd(update, context):
    await update.message.reply_text(status_msg())

async def update_cmd(update, context):
    """
    تحديث الحساب
    مثال: /update pnl=+3.5 dd=2.5 daily=1.0 trades=2
    """
    try:
        args = " ".join(context.args)
        updated = []

        if "pnl=" in args:
            val = float(args.split("pnl=")[1].split()[0].replace("+",""))
            ACCOUNT['pnl_percent'] = val
            updated.append(f"PnL: {'+' if val>=0 else ''}{val}%")

        if "dd=" in args:
            val = float(args.split("dd=")[1].split()[0])
            ACCOUNT['drawdown_used'] = val
            updated.append(f"دروداون: {val}%")

        if "daily=" in args:
            val = float(args.split("daily=")[1].split()[0])
            ACCOUNT['daily_used'] = val
            updated.append(f"ديلي: {val}%")

        if "trades=" in args:
            val = int(args.split("trades=")[1].split()[0])
            ACCOUNT['trades_week'] = val
            updated.append(f"صفقات: {val}")

        if updated:
            await update.message.reply_text(
                f"✅ تم التحديث!\n" + "\n".join(updated) +
                "\n\nحسابك محفوظ عندي 💙"
            )
        else:
            await update.message.reply_text(
                "الاستخدام:\n"
                "/update pnl=+3.5 dd=2.5 daily=1.0 trades=2\n\n"
                "pnl = نسبة الربح أو الخسارة\n"
                "dd = الدروداون المستخدم\n"
```

```
                "daily = اليومي المستخدم\n"
                "trades = عدد الصفقات هالأسبوع\n\n"
                "مثال: لو رابح 3.5% وعندك 2.5% دروداون\n"
                "/update pnl=+3.5 dd=2.5 daily=0.5 trades=1"
            )
    except Exception as e:
        await update.message.reply_text(
            "❌ في خطأ في البيانات\n"
            "مثال صح: /update pnl=+3.5 dd=2.5 daily=1.0 trades=2"
        )

async def main():
    app = Application.builder().token(TELEGRAM_TOKEN).build()
    app.add_handler(CommandHandler("start", start_cmd))
    app.add_handler(CommandHandler("scan", scan_cmd))
    app.add_handler(CommandHandler("advice", advice_cmd))
    app.add_handler(CommandHandler("status", status_cmd))
    app.add_handler(CommandHandler("update", update_cmd))

    bot = Bot(token=TELEGRAM_TOKEN)
    async with app:
        await app.start()
        await app.updater.start_polling()
        await trading_loop(bot)

if __name__ == "__main__":
    asyncio.run(main())
```