```

Mql

//+------------------------------------------------------------------+

//| |

//| Expert Advisor Name: Jupiter |

//| Author: ……… |

//| |

//| |

//+------------------------------------------------------------------+

#property copyright “@Ashirafyiga!!!”

#property version “1.0.1”

#property strict

// Input parameters

Input double DailyTakeProfitTarget = 100;//Total daily profit in Account Currency;

Input double DailyProfitIncrease = 10; // Daily profit increase percentage;

Input bool DailyProfitTarget = false; // Disable daily profit target;

Input double GoalProfit = 2000; // Goal-based profit target in Account Currency;

Input bool UseGoalProfit = true; // Enable/Disable goal-based profit strategy;

Extern double TimeToReachGoal = 24; // Time to reach goal (hours);

Input double DailyStopLoss = 50;//Total daily loss in Account Currency;

Input double AutoLotRisk = 2;

Input int MagicNumber = 123;

Input bool MultiTrades = true; // Enable multiple trades at same time;

Input int MaxTrades = 5; // Maximum number of trades;

Input double TrailingStopAmount = 50.0; // Trailing stop amount in Account Currency;

Input double TrailingStepAmount = 10.0; // Trailing step amount in account currency;

Input double ThresholdFair = 1.5;

Input double ThresholdGood = 2.0;

Input double ThresholdExcellent = 2.1;

Input color ColorFair = Red;

Input color ColorGood = Orange;

Input color ColorExcellent = Green;

Input int StartHour = 9;

Input int StartMinute = 0;

Input int EndHour = 17;

Input int EndMinute = 0;

Input string NotificationEmail = “ramous777@gmail.com”;

Input string NotificationPhone = “+256709930629”;

Input string API\_KEY = “9999”; // Alpha Vantage API key;

Input string NEWS\_SYMBOLS = “EUR,JPY,GBP,USD”; // News symbols to filter;

Input enum Impact { Medium, High } NEWS\_IMPACT = High; // News impact filter;

Input int UPDATE\_INTERVAL = 10; // News update interval (minutes);

Input double MaxSpread = 5; // Maximum allowed spread;

Input double MaxSlippage = 5; // Maximum allowed slippage;

// Global variables

Double equity;

Double balance;

Double dailyProfit;

Datetime lastResetTime;

Double dailyProfitFactor;

Double[] historyProfitFactor;

String newsApiUrl = “(link unavailable)?”;

String newsParameters = “function=NEWS\_SENTIMENT&symbols=”;

String newsApiKey = “&apikey=” + API\_KEY;

String newsRequest;

// Initialize news request timer

Datetime startTime;

//+------------------------------------------------------------------+

// Check account login and expiry date

Input string AllowedLogins = “123456,789012,345678”;

Input datetime ExpiryDate = “2024.12.31 23:59”;

Int OnInit()

{

String currentLogin = AccountInfoString(ACCOUNT\_LOGIN);

If (!StringFind(AllowedLogins, currentLogin))

{

Alert(“Unauthorized account login!”);

Return INIT\_FAILED;

}

Datetime currentTime = iTime(Symbol(), PERIOD\_CURRENT, 0);

If (currentTime > ExpiryDate)

{

Alert(“EA has expired!”);

Return INIT\_FAILED;

}

Return INIT\_SUCCEEDED;

}

//| Expert initialization function |

//+------------------------------------------------------------------+

Int OnInit()

{

// Create chart panel

CreateChartPanel();

Return(INIT\_SUCCEEDED);

}

//+------------------------------------------------------------------+

//| Create chart panel |

//+------------------------------------------------------------------+

Void CreateChartPanel()

{

Static int panelHandle = 0;

// Define colors

Color COLOR\_BACKGROUND = White;

Color COLOR\_TEXT = Gold;

Color COLOR\_PROFIT = Cyan;

Color COLOR\_LOSS = Magenta;

// Define fonts

String FONT\_NAME = “Arial”;

Int FONT\_SIZE = 12;

// Create panel

If (panelHandle == 0)

{

panelHandle = ObjectCreate(“ChartPanel”, OBJ\_PANEL, 0, 0);

ObjectSet(“ChartPanel”, OBJPROP\_BACKCOLOR, COLOR\_BACKGROUND);

ObjectSet(“ChartPanel”, OBJPROP\_BORDERCOLOR, COLOR\_TEXT);

ObjectSet(“ChartPanel”, OBJPROP\_BORDERWIDTH, 1);

// Pair label

ObjectCreate(“PairLabel”, OBJ\_LABEL, 0, 10, “Pair: “ + Symbol());

ObjectSet(“PairLabel”, OBJPROP\_FONT, FONT\_NAME);

ObjectSet(“PairLabel”, OBJPROP\_FONTSIZE, FONT\_SIZE);

ObjectSet(“PairLabel”, OBJPROP\_COLOR, COLOR\_TEXT);

// Timeframe label

ObjectCreate(“TimeframeLabel”, OBJ\_LABEL, 0, 30, “Timeframe: “ + Period());

ObjectSet(“TimeframeLabel”, OBJPROP\_FONT, FONT\_NAME);

ObjectSet(“TimeframeLabel”, OBJPROP\_FONTSIZE, FONT\_SIZE);

ObjectSet(“TimeframeLabel”, OBJPROP\_COLOR, COLOR\_TEXT);

// Floating PL label

ObjectCreate(“FloatingPLLabel”, OBJ\_LABEL, 0, 50, “Floating PL: “);

ObjectSet(“FloatingPLLabel”, OBJPROP\_FONT, FONT\_NAME);

ObjectSet(“FloatingPLLabel”, OBJPROP\_FONTSIZE, FONT\_SIZE);

ObjectSet(“FloatingPLLabel”, OBJPROP\_COLOR, COLOR\_TEXT);

// Daily profit target label

ObjectCreate(“DailyProfitTargetLabel”, OBJ\_LABEL, 0, 70, “Daily Profit Target: “);

ObjectSet(“DailyProfitTargetLabel”, OBJPROP\_FONT, FONT\_NAME);

ObjectSet(“DailyProfitTargetLabel”, OBJPROP\_FONTSIZE, FONT\_SIZE);

ObjectSet(“DailyProfitTargetLabel”, OBJPROP\_COLOR, COLOR\_TEXT);

// Daily profit target input

ObjectCreate(“DailyProfitTargetInput”, OBJ\_EDIT, 150, 70, 50);

ObjectSet(“DailyProfitTargetInput”, OBJPROP\_BACKCOLOR, White);

ObjectSet(“DailyProfitTargetInput”, OBJPROP\_BORDERCOLOR, COLOR\_TEXT);

ObjectSet(“DailyProfitTargetInput”, OBJPROP\_BORDERWIDTH, 1);

ObjectSet(“DailyProfitTargetInput”, OBJPROP\_FONT, FONT\_NAME);

ObjectSet(“DailyProfitTargetInput”, OBJPROP\_FONTSIZE, FONT\_SIZE);

ObjectSetText(“DailyProfitTargetInput”, DoubleToString(DailyTakeProfitTarget, 2));

// Daily stop loss label

ObjectCreate(“DailyStopLossLabel”, OBJ\_LABEL, 0, 90, “Daily Stop Loss: “ + DoubleToString(DailyStopLoss, 2));

ObjectSet(“DailyStopLossLabel”, OBJPROP\_FONT, FONT\_NAME);

ObjectSet(“DailyStopLossLabel”, OBJPROP\_FONTSIZE, FONT\_SIZE);

ObjectSet(“DailyStopLossLabel”, OBJPROP\_COLOR, COLOR\_TEXT);

// Risk per trade label

ObjectCreate(“RiskPerTradeLabel”, OBJ\_LABEL, 0, 110, “Risk per Trade: “ + DoubleToString(AutoLotRisk, 2) + “%”);

ObjectSet(“RiskPerTradeLabel”, OBJPROP\_FONT, FONT\_NAME);

ObjectSet(“RiskPerTradeLabel”, OBJPROP\_FONTS

ObjectSet(“RiskPerTradeLabel”, OBJPROP\_FONTSIZE, FONT\_SIZE);

ObjectSet(“RiskPerTradeLabel”, OBJPROP\_COLOR, COLOR\_TEXT);

// Risk per trade input

ObjectCreate(“RiskPerTradeInput”, OBJ\_EDIT, 150, 110, 50);

ObjectSet(“RiskPerTradeInput”, OBJPROP\_BACKCOLOR, White);

ObjectSet(“RiskPerTradeInput”, OBJPROP\_BORDERCOLOR, COLOR\_TEXT);

ObjectSet(“RiskPerTradeInput”, OBJPROP\_BORDERWIDTH, 1);

ObjectSet(“RiskPerTradeInput”, OBJPROP\_FONT, FONT\_NAME);

ObjectSet(“RiskPerTradeInput”, OBJPROP\_FONTSIZE, FONT\_SIZE);

ObjectSetText(“RiskPerTradeInput”, DoubleToString(AutoLotRisk, 2));

}

// Initialize news request timer

startTime = iTime(Symbol(), PERIOD\_M1, 0);

return(INIT\_SUCCEEDED);

}

// Strategy logic

Int buySignals = 0;

Int sellSignals = 0;

// NRTR Volatility

Double NRTR = iCustom(Symbol(), PERIOD\_CURRENT, “NRTR Volatility”,

NRTRHistory, NRTRATRPeriod, NRTRATRKoef,

NRTRPricePeriod, NRTRPriceMode, NRTRPricePrice,

NRTRPriceShift);

If (NRTR > 0) buySignals++; else if (NRTR < 0) sellSignals++;

// BrainTrend2

Double Brain = iCustom(Symbol(), PERIOD\_CURRENT, “BrainTrend2”,

BrainTrendNumBars, BrainTrendArtP, BrainTrendDartP,

BrainTrendCecf);

If (Brain > 0) buySignals++; else if (Brain < 0) sellSignals++;

// Close trade logic

If (sellSignals >= 2 && OrdersTotal() > 0) {

For (int I = OrdersTotal() – 1; I >= 0; i--) {

OrderSelect(I, SELECT\_BY\_POS);

If (OrderMagicNumber() == MagicNumber && OrderSymbol() == Symbol()) {

If (OrderType() == OP\_BUY) {

OrderClose(OrderTicket(), OrderLots(), Bid, 3, Violet);

Alert(“Buy trade closed successfully. Ticket: “, OrderTicket());

}

}

}

} else if (buySignals >= 2 && OrdersTotal() > 0) {

For (int I = OrdersTotal() – 1; I >= 0; i--) {

OrderSelect(I, SELECT\_BY\_POS);

If (OrderMagicNumber() == MagicNumber && OrderSymbol() == Symbol()) {

If (OrderType() == OP\_SELL) {

OrderClose(OrderTicket(), OrderLots(), Ask, 3, Violet);

Alert(“Sell trade closed successfully. Ticket: “, OrderTicket());

}

}

}

}

// Open trade logic

If (CanOpenNewTrade()) {

If (buySignals >= 2) {

// Open buy trade

Int ticket = OrderSend(Symbol(), OP\_BUY, 0.1, Ask, 3, Bid – 50 \* Point, Bid + 50 \* Point, “Jupiter EA Buy”, MagicNumber, 0, Green);

If (ticket > 0) {

Alert(“Buy trade opened successfully. Ticket: “, ticket);

} else {

Alert(“Error opening buy trade. Error code: “, GetLastError());

}

} else if (sellSignals >= 2) {

// Open sell trade

Int ticket = OrderSend(Symbol(), OP\_SELL, 0.1, Bid, 3, Ask + 50 \* Point, Ask – 50 \* Point, “Jupiter EA Sell”, MagicNumber, 0, Red);

If (ticket > 0) {

Alert(“Sell trade opened successfully. Ticket: “, ticket);

} else {

Alert(“Error opening sell trade. Error code: “, GetLastError());

}

}

}

// Check if trade is in profit

If (OrderClosePrice() – OrderOpenPrice() > 0) {

// Calculate trailing stop loss

Double stopLoss = OrderOpenPrice() – (TrailingStopAmount / SymbolInfoDouble(\_Symbol, SYMBOL\_BID));

// Trail stop loss

If (OrderClosePrice() – OrderOpenPrice() > TrailingStepAmount) {

stopLoss = MathMax(OrderOpenPrice() – (TrailingStopAmount / SymbolInfoDouble(\_Symbol, SYMBOL\_BID)), OrderStopLoss());

OrderModify(ticket, OrderOpenPrice(), stopLoss, OrderTakeProfit());

}

}

}

// Update labels

Try {

// Update labels

ObjectSetText(“PairLabel”, “Pair: “ + Symbol());

ObjectSetText(“TimeframeLabel”, “Timeframe: “ + Period());

ObjectSetText(“FloatingPLLabel”, “Floating PL: “ + DoubleToString(AccountInfoDouble(ACCOUNT\_PROFIT), 2));

ObjectSetText(“DailyProfitTargetLabel”, “Daily Profit Target: “ + DoubleToString(DailyTakeProfitTarget, 2));

ObjectSetText(“DailyStopLossLabel”, “Daily Stop Loss: “ + DoubleToString(DailyStopLoss, 2));

ObjectSetText(“RiskPerTradeLabel”, “Risk per Trade: “ + DoubleToString(AutoLotRisk, 2) + “%”);

} catch (int err) {

ConsoleWrite(“Error updating chart panel: “, err);

}

}

ObjectSetText(“PairLabel”, “Pair: “ + Symbol(), 12, FONT\_NAME, COLOR\_TEXT);

ObjectSetText(“TimeframeLabel”, “Timeframe: “ + Period(), 12, FONT\_NAME, COLOR\_TEXT);

ObjectSetText(“FloatingPLLabel”, “Floating PL: “ + DoubleToString(AccountInfoDouble(ACCOUNT\_PROFIT), 2), 12, FONT\_NAME, (AccountInfoDouble(ACCOUNT\_PROFIT) >= 0) ? COLOR\_PROFIT : COLOR\_LOSS);

ObjectSetText(“DailyProfitTargetLabel”, “Daily Profit Target: “, 12, FONT\_NAME, COLOR\_TEXT);

ObjectSetText(“DailyStopLossLabel”, “Daily Stop Loss: “ + DoubleToString(DailyStopLoss, 2), 12, FONT\_NAME, COLOR\_TEXT);

ObjectSetText(“RiskPerTradeLabel”, “Risk per Trade: “ + DoubleToString(AutoLotRisk, 2) + “%”, 12, FONT\_NAME, COLOR\_TEXT);

}

//+------------------------------------------------------------------+

//| Expert tick function |

//+------------------------------------------------------------------+

Void OnTick()

{

Static datetime newsTime = 0;

Static bool newsDetected = false;

// Check if news update interval has passed

If (iTime(Symbol(), PERIOD\_M1, 0) – startTime >= UPDATE\_INTERVAL \* 60)

{

// Reset news request timer

startTime = iTime(Symbol(), PERIOD\_M1, 0);

// Make news request to Alpha Vantage API

newsRequest = newsApiUrl + newsParameters + NEWS\_SYMBOLS + newsApiKey;

int handle = WebRequest(Get, newsRequest, NULL, NULL, NULL);

// Check if news request was successful

If (handle > 0)

{

// Parse news response

String newsResponse = WebRequestResult(handle);

JSONObj json = JSONParse(newsResponse);

// Filter news by impact and symbol

For (int I = 0; I < JSONObjGetArraySize(json, “news”); i++)

{

String newsSymbol = JSONObjGetString(json, “news[“ + I + “].symbol”);

String newsImpact = JSONObjGetString(json, “news[“ + I + “].impact”);

Datetime newsTimestamp = JSONObjGetInteger(json, “news[“ + I + “].timestamp”);

Check if news symbol and impact match filters

If (StringFind(NEWS\_SYMBOLS, newsSymbol) != -1 && newsImpact == NEWS\_IMPACT)

{

newsTime = newsTimestamp;

newsDetected = true;

}

}

}

}

// Only open new trades if time to news is greater than 20 minutes

If (newsDetected && (newsTime – iTime(Symbol(), PERIOD\_M1, 0)) / 60 < 20)

{

// Prevent new trades 20 minutes before news event

Return;

}

Else if (!newsDetected)

{

// Open trades logic…

}

// Reset news detection flag after news event passes

If (newsDetected && (newsTime – iTime(Symbol(), PERIOD\_M1, 0)) / 60 > 20)

{

newsDetected = false;

}

}

Void OnTick()

{

Int OnInit()

{

// Check spread and slippage

CheckSpreadAndSlippage();

} catch (err) {

Alert(“Error on tick: “, err);

}

}

//+------------------------------------------------------------------+

//| Check spread and slippage |

//+------------------------------------------------------------------+

Void CheckSpreadAndSlippage() {

Double spread = SymbolInfoDouble(\_Symbol, SYMBOL\_SPREAD);

Double ask = SymbolInfoDouble(\_Symbol, SYMBOL\_ASK);

Double bid = SymbolInfoDouble(\_Symbol, SYMBOL\_BID);

// Check maximum spread

If (spread > MaxSpread) {

Alert(“Spread exceeds maximum allowed: “, spread);

Return;

}

// Check maximum slippage for buy orders

If (OrderType() == OP\_BUY) {

Double slippage = ask – OrderOpenPrice();

If (slippage > MaxSlippage) {

Alert(“Slippage exceeds maximum allowed for buy order: “, slippage);

Return;

}

}

// Check maximum slippage for sell orders

If (OrderType() == OP\_SELL) {

Double slippage = OrderOpenPrice() – bid;

If (slippage > MaxSlippage) {

Alert(“Slippage exceeds maximum allowed for sell order: “, slippage);

Return;

}

}

}

lastResetTime = iTime(Symbol(), PERIOD\_DAILY, 0);

ArrayResize(historyProfitFactor, 30);

Return(INIT\_SUCCEEDED);

}

//+------------------------------------------------------------------+

//| Tick function |

//+------------------------------------------------------------------+

Void OnTick()

{

// Check if new day

If (iTime(Symbol(), PERIOD\_DAILY, 0) > lastResetTime)

{

lastResetTime = iTime(Symbol(), PERIOD\_DAILY, 0);

dailyProfitFactor = 0;

}

// Calculate daily profit factor

Int totalProfitTrades = 0;

Int totalLossTrades = 0;

For (int I = 0; I < OrdersTotal(); i++)

{

If (OrderSelect(I, SELECT\_BY\_POS) && OrderMagicNumber() == MagicNumber && OrderSymbol() == Symbol())

{

If (OrderClosePrice() > OrderOpenPrice())

{

totalProfitTrades++;

}

Else if (OrderClosePrice() < OrderOpenPrice())

{

totalLossTrades++;

}

}

}

If (totalLossTrades > 0)

{

dailyProfitFactor = (double)totalProfitTrades / totalLossTrades;

}

Else

{

dailyProfitFactor = 0;

}

// Update history

ArrayShift(historyProfitFactor);

historyProfitFactor[0] = dailyProfitFactor;

// Visualization

String label = “Daily Profit Factor: “ + DoubleToString(dailyProfitFactor, 2);

ObjectCreate(“ProfitFactorLabel”, OBJ\_LABEL, 0, 0, 0);

ObjectSetText(“ProfitFactorLabel”, label, 12, “Arial”, (dailyProfitFactor >= ThresholdExcellent) ? ColorExcellent : ((dailyProfitFactor >= ThresholdGood) ? ColorGood : ColorFair));

// Line graph

ObjectCreate(“ProfitFactorGraph”, OBJ\_LINE, 0, 0, 0);

ObjectSet(“ProfitFactorGraph”, OBJPROP\_DATA, historyProfitFactor);

ObjectSet(“ProfitFactorGraph”, OBJPROP\_COLOR, (dailyProfitFactor >= ThresholdExcellent) ? ColorExcellent : ((dailyProfitFactor >= ThresholdGood) ? ColorGood : ColorFair));

// Save data when EA stops trading

If (!IsTesting() && !IsOptimization())

{

GlobalVariableSet(“DailyProfitFactorHistory”, historyProfitFactor);

}

}

// Check if trading time

If (TimeHour() >= StartHour && TimeHour() <= EndHour && TimeMinute() >= StartMinute && TimeMinute() <= EndMinute)

{

// Update equity and balance

Equity = AccountEquity();

Balance = AccountBalance();

// Initialize daily profit target

dailyProfitTarget = DailyTakeProfitTarget;

// Load previous day’s profit from file

LoadPreviousDayProfit();

}

//+------------------------------------------------------------------+

//| Load previous day’s profit function |

//+------------------------------------------------------------------+

Void LoadPreviousDayProfit()

{

// File path

String filePath = “previous\_day\_profit.txt”;

// Check if file exists

If (FileIsExist(filePath))

{

// Load previous day’s profit

previousDayProfit = FileReadDouble(filePath);

// Calculate new daily profit target

dailyProfitTarget = previousDayProfit \* (1 + DailyProfitIncrease/100);

}

}

//+------------------------------------------------------------------+

//| Expert tick function |

//+------------------------------------------------------------------+

Void OnTick()

{

// Check daily profit target

If (AccountInfoDouble(ACCOUNT\_PROFIT) >= dailyProfitTarget)

{

// Check daily profit target

If (dailyProfit >= DailyTakeProfitTarget)

{

// Send notification

SendNotification(NotificationEmail, NotificationPhone, “Daily profit target reached”);

dailyProfit = 0;

}

// Check daily stop loss

If (equity – balance <= -DailyStopLoss)

{

// Close all trades

CloseTrades();

dailyProfit = 0;

}

Goal-based profit strategy.

// Exponential factor calculation

Double ExpFactor = 1 + (TimeToReachGoal / 24) ^ (1 / (1 + (GoalProfit / 100)));

// Lot size calculation

Double LotSizeModified = (GoalProfit / (TimeToReachGoal \* (1 – (MaxSpread / 100)))) \* (1 + (MaxSlippage / 100)) \* ExpFactor;

If (UseGoalProfit && AccountProfit() >= GoalProfit) {

// Close all positions and stop trading

For (int I = OrdersTotal() – 1; I >= 0; i--) {

OrderSelect(I, SELECT\_BY\_POS);

If (OrderSymbol() == Symbol()) {

OrderClose(OrderTicket(), OrderLots());

}

}

// Stop trading

Return;

}

//+------------------------------------------------------------------+

Int totalTrades = 0; // Counter for total trades

//+------------------------------------------------------------------+

//| Check if multiple trades are allowed |

//+------------------------------------------------------------------+

Bool AllowMultipleTrades()

{

Return MultiTrades && totalTrades < MaxTrades;

}

//+------------------------------------------------------------------+

//| Count total trades |

//+------------------------------------------------------------------+

Void CountTrades()

{

totalTrades = 0;

for (int I = 0; I < OrdersTotal(); i++)

{

If (OrderSelect(I, SELECT\_BY\_POS) && OrderMagicNumber() == MagicNumber && OrderSymbol() == Symbol())

{

totalTrades++;

}

}

}

//+------------------------------------------------------------------+

//| Check if new trade can be opened |

//+------------------------------------------------------------------+

Bool CanOpenNewTrade()

{

CountTrades();

Return AllowMultipleTrades();

}

//+------------------------------------------------------------------+

Open trade function |

//+------------------------------------------------------------------+

Void OpenTrade(int type, double price)

{

If (CanOpenNewTrade())

{

Int tradesToOpen = MaxTrades – totalTrades;

If (CanOpenNewTrade())

{

// Open trade logic

}

}

// Check current trades

If (OrderSelect(I, SELECT\_BY\_POS) && OrderMagicNumber() == MagicNumber && OrderSymbol() == Symbol())

{

totalTrades++;

}

}

// Strategy logic

Int buySignals = 0;

Int sellSignals = 0;

// NRTR Volatility

Double NRTR = iCustom(Symbol(), PERIOD\_CURRENT, “NRTR Volatility”, NRTRHistory, NRTRATRPeriod, NRTRATRKoef, NRTRPricePeriod, NRTRPriceMode, NRTRPricePrice, NRTRPriceShift);

If (NRTR > 0) buySignals++; else if (NRTR < 0) sellSignals++;

// BrainTrend2

Double Brain = iCustom(Symbol(), PERIOD\_CURRENT, “BrainTrend2”, BrainTrendNumBars, BrainTrendArtP, BrainTrendDartP, BrainTrendCecf);

If (Brain > 0) buySignals++; else if (Brain < 0) sellSignals++;

// MegaTrend HMA

Double HMA = iCustom(Symbol(), PERIOD\_CURRENT, “MegaTrend HMA”, HMATrendPeriod, HMATrendMethod, HMATrendPrice, HMATrendShift);

If (HMA > 0) buySignals++; else if (HMA < 0) sellSignals++;

// Check for two identical signals

If (buySignals >= 2 && totalTrades == 0)

// Strategy logic

Int buySignals = 0;

Int sellSignals = 0;

// NRTR Volatility

Double NRTR = iCustom(Symbol(), PERIOD\_CURRENT, “NRTR Volatility”,

NRTRHistory, NRTRATRPeriod, NRTRATRKoef,

NRTRPricePeriod, NRTRPriceMode, NRTRPricePrice,

NRTRPriceShift);

If (NRTR > 0) buySignals++; else if (NRTR < 0) sellSignals++;

// BrainTrend2

Double Brain = iCustom(Symbol(), PERIOD\_CURRENT, “BrainTrend2”,

BrainTrendNumBars, BrainTrendArtP, BrainTrendDartP,

BrainTrendCecf);

If (Brain > 0) buySignals++; else if (Brain < 0) sellSignals++;

// Close trade logic

If (sellSignals >= 2 && OrdersTotal() > 0) {

For (int I = OrdersTotal() – 1; I >= 0; i--) {

OrderSelect(I, SELECT\_BY\_POS);

If (OrderMagicNumber() == MagicNumber && OrderSymbol() == Symbol()) {

If (OrderType() == OP\_BUY) {

OrderClose(OrderTicket(), OrderLots(), Bid, 3, Violet);

Alert(“Buy trade closed successfully. Ticket: “, OrderTicket());

}

}

}

} else if (buySignals >= 2 && OrdersTotal() > 0) {

For (int I = OrdersTotal() – 1; I >= 0; i--) {

OrderSelect(I, SELECT\_BY\_POS);

If (OrderMagicNumber() == MagicNumber && OrderSymbol() == Symbol()) {

If (OrderType() == OP\_SELL) {

OrderClose(OrderTicket(), OrderLots(), Ask, 3, Violet);

Alert(“Sell trade closed successfully. Ticket: “, OrderTicket());

}

}

}

}

// Open trade logic

If (CanOpenNewTrade()) {

If (buySignals >= 2) {

// Open buy trade

Int ticket = OrderSend(Symbol(), OP\_BUY, 0.1, Ask, 3, Bid – 50 \* Point, Bid + 50 \* Point, “Jupiter EA Buy”, MagicNumber, 0, Green);

If (ticket > 0) {

Alert(“Buy trade opened successfully. Ticket: “, ticket);

} else {

Alert(“Error opening buy trade. Error code: “, GetLastError());

}

} else if (sellSignals >= 2) {

// Open sell trade

Int ticket = OrderSend(Symbol(), OP\_SELL, 0.1, Bid, 3, Ask + 50 \* Point, Ask – 50 \* Point, “Jupiter EA Sell”, MagicNumber, 0, Red);

If (ticket > 0) {

Alert(“Sell trade opened successfully. Ticket: “, ticket);

} else {

Alert(“Error opening sell trade. Error code: “, GetLastError());

}

}

}

// Open buy trade

OpenTrade(OP\_BUY, Ask);

ObjectCreate(“BuyLine”, OBJ\_HLINE, 0, 0, Ask);

ObjectSet(“BuyLine”, OBJPROP\_COLOR, Cyan);

ObjectSet(“BuyLine”, OBJPROP\_STYLE, STYLE\_DASH);

}

Else if (sellSignals >= 2 && totalTrades == 0)

{

// Open sell trade

OpenTrade(OP\_SELL, Bid);

ObjectCreate(“SellLine”, OBJ\_HLINE

```

// Risk per trade label

ObjectCreate(“RiskPerTradeLabel”, OBJ\_LABEL, 0, 110, “Risk per Trade: “ + DoubleToString(AutoLotRisk, 2) + “%”);

ObjectSet(“RiskPerTradeLabel”, OBJPROP\_FONT, FONT\_NAME);

ObjectSet(“RiskPerTradeLabel”, OBJPROP\_FONTSIZE, FONT\_SIZE);

ObjectSet(“RiskPerTradeLabel”, OBJPROP\_COLOR, COLOR\_TEXT);

// Risk per trade input

ObjectCreate(“RiskPerTradeInput”, OBJ\_EDIT, 150, 110, 50);

ObjectSet(“RiskPerTradeInput”, OBJPROP\_BACKCOLOR, White);

ObjectSet(“RiskPerTradeInput”, OBJPROP\_BORDERCOLOR, COLOR\_TEXT);

ObjectSet(“RiskPerTradeInput”, OBJPROP\_BORDERWIDTH, 1);

ObjectSet(“RiskPerTradeInput”, OBJPROP\_FONT, FONT\_NAME);

ObjectSet(“RiskPerTradeInput”, OBJPROP\_FONTSIZE, FONT\_SIZE);

ObjectSetText(“RiskPerTradeInput”, DoubleToString(AutoLotRisk, 2));

}

// Update labels

ObjectSetText(“PairLabel”, “Pair: “ + Symbol(), 12, FONT\_NAME, COLOR\_TEXT);

ObjectSetText(“TimeframeLabel”, “Timeframe: “ + Period(), 12, FONT\_NAME, COLOR\_TEXT);

ObjectSetText(“FloatingPLLabel”, “Floating PL: “ + DoubleToString(AccountInfoDouble(ACCOUNT\_PROFIT), 2), 12, FONT\_NAME, (AccountInfoDouble(ACCOUNT\_PROFIT) >= 0) ? COLOR\_PROFIT : COLOR\_LOSS);

ObjectSetText(“DailyProfitTargetLabel”, “Daily Profit Target: “, 12, FONT\_NAME, COLOR\_TEXT);

ObjectSetText(“DailyStopLossLabel”, “Daily Stop Loss: “ + DoubleToString(DailyStopLoss, 2), 12, FONT\_NAME, COLOR\_TEXT);

ObjectSetText(“RiskPerTradeLabel”, “Risk per Trade: “ + DoubleToString(AutoLotRisk, 2) + “%”, 12, FONT\_NAME, COLOR\_TEXT);

}

//+------------------------------------------------------------------+

Maximum Trades:

Input int MaxTrades = 5; // Maximum number of trades

Int totalTrades = 0;

For (int I = 0; I < OrdersTotal(); i++) {

If (OrderSelect(I, SELECT\_BY\_POS) && OrderMagicNumber() == MagicNumber && OrderSymbol() == Symbol()) {

totalTrades++;

}

}

If (totalTrades >= MaxTrades) {

// Do not open new trades

Return;

}

//+------------------------------------------------------------------+

//| Expert deinitialization function |

//+------------------------------------------------------------------+

Void OnDeinit(const int reason)

{

ObjectDelete(“ProfitFactorLabel”);

ObjectDelete(“ProfitFactorGraph”);

// Delete unnecessary objects

For (int I = ObjectsTotal() – 1; I >= 0; i--)

{

String objName = ObjectName(i);

If (StringFind(objName, “HistoryLabel”) >= 0)

{

ObjectDelete(objName);

}

}

//+------------------------------------------------------------------+

//| Expert deinitialization function |

//+------------------------------------------------------------------+

Void OnDeinit(const int reason)

{

Try {

// Remove chart objects

ObjectDelete(“ChartPanel”);

ObjectDelete(“PairLabel”);

ObjectDelete(“TimeframeLabel”);

ObjectDelete(“FloatingPLLabel”);

ObjectDelete(“DailyProfitTargetLabel”);

ObjectDelete(“DailyStopLossLabel”);

ObjectDelete(“RiskPerTradeLabel”);

// Remove news request timer

EventKillTimer();

// Save data when EA stops trading

If (!IsTesting() && !IsOptimization()) {

GlobalVariableSet(“DailyProfitFactorHistory”, historyProfitFactor);

}

} catch (int err) {

ConsoleWrite(“Error on deinit: “, err);

}

}

//+------------------------------------------------------------------+

//| Expert timer function |

//+------------------------------------------------------------------+

Void OnTimer()

{

// Update chart objects

ObjectSet(“ProfitFactorLabel”, OBJPROP\_TEXT,

“Daily Profit Factor: “ + DoubleToString(dailyProfitFactor, 2));

ObjectSet(“ProfitFactorGraph”, OBJPROP\_DATA, historyProfitFactor);

// Save data every hour

If (!IsTesting() && !IsOptimization())

{

GlobalVariableSet(“DailyProfitFactorHistory”, historyProfitFactor);

}

}

//+------------------------------------------------------------------+

//| Chart event function |

//+------------------------------------------------------------------+

Void OnChartEvent(const int id, const long &lparam, const double &dparam, const string &sparam)

{

// Handle user interactions (e.g., clicking on chart objects)

If (id == CHARTEVENT\_OBJECT\_CLICK)

{

// Handle clicks on chart objects

}

}

// Delete chart panel

ObjectDelete(“ChartPanel”);

// Delete labels

ObjectDelete(“PairLabel”);

ObjectDelete(“TimeframeLabel”);

ObjectDelete(“FloatingPLLabel”);

ObjectDelete(“DailyProfitTargetLabel”);

ObjectDelete(“DailyStopLossLabel”);

ObjectDelete(“RiskPerTradeLabel”);

ObjectDelete(“DailyProfitTargetInput”);

ObjectDelete(“RiskPerTradeInput”);

}

//+------------------------------------------------------------------+

//| Expert tick function |

//+------------------------------------------------------------------+

//+------------------------------------------

//+------------------------------------------------------------------+

//| Expert tick function |

//+------------------------------------------------------------------+

Void OnTick()

{

Try {

// Check spread and slippage

CheckSpreadAndSlippage();

// Check daily profit target

CheckDailyProfitTarget();

// Check daily stop loss

CheckDailyStopLoss();

// Goal-based profit strategy

CheckGoalProfit();

// Open trade logic

OpenTradeLogic();

// Manage trades

ManageTrades();

// Update chart panel

UpdateChartPanel();

// Request news

RequestNews();

} catch (int err) {

ConsoleWrite(“Error on tick: “, err);

}

}

Void OnTick()

{

// Check trading hours

If (Hour() >= StartHour && Minute() >= StartMinute && Hour() <= EndHour && Minute() <= EndMinute)

{

// Check daily profit target

If (AccountInfoDouble(ACCOUNT\_PROFIT) >= DailyTakeProfitTarget)

{

// Close all trades

CloseTrades();

}

// Check daily stop loss

If (AccountInfoDouble(ACCOUNT\_PROFIT) <= -DailyStopLoss)

{

// Close all trades

CloseTrades();

}

// Check buy signals

If (buySignals >= 2 && totalTrades == 0)

{

// Open buy trade

OpenTrade(OP\_BUY, Ask);

}

Else if (sellSignals >= 2 && totalTrades == 0)

{

// Open sell trade

OpenTrade(OP\_SELL, Bid);

}

// ATR Trailing Stop

Double atr = iATR(Symbol(), PERIOD\_CURRENT, ATRPeriod);

Double trailingStopDistance = atr \* ATRMultiplier;

For (int I = 0; I < OrdersTotal(); i++)

{

If (OrderSelect(I, SELECT\_BY\_POS) && OrderMagicNumber() == MagicNumber && OrderSymbol() == Symbol())

{

If (OrderType() == OP\_BUY)

{

If (Bid – OrderOpenPrice() >= BreakEvenThreshold)

{

// Move stop-loss to break-even

OrderModify(OrderTicket(), OrderOpenPrice(), 0, 0);

}

Else

{

// Move stop-loss using ATR

OrderModify(OrderTicket(), Bid – trailingStopDistance, 0, 0);

}

}

Else if (OrderType() == OP\_SELL)

{

If (OrderOpenPrice() – Ask >= BreakEvenThreshold)

{

// Move stop-loss to break-even

OrderModify(OrderTicket(), OrderOpenPrice(), 0, 0);

}

Else

{

// Move stop

ObjectCreate(“SellLine”, OBJ\_HLINE, 0, 0, Bid);

ObjectSet(“SellLine”, OBJPROP\_COLOR, Magenta);

ObjectSet(“SellLine”, OBJPROP\_STYLE, STYLE\_DASH);

}

}

}

//+------------------------------------------------------------------+

//| Open trade function |

//+------------------------------------------------------------------+

Void OpenTrade(int type, double price)

{Void ManageTrades()

{

Try {

// Loop through trades

For (int I = OrdersTotal() – 1; I >= 0; i--) {

OrderSelect(I, SELECT\_BY\_POS);

// Check trade magic number

If (OrderMagicNumber() == MagicNumber) {

// Manage trade

ManageTrade(OrderTicket(), OrderOpenPrice());

}

}

} catch (int err) {

ConsoleWrite(“Error managing trades: “, err);

}

}

Int ticket = OrderSend(Symbol(), type, AutoLot(), price, 3, Bid – 20 \* Point, Ask + 20 \* Point, “MyExpertAdvisor”, MagicNumber, 0, Green);

If (ticket > 0)

{

// Send notification

}

//+------------------------------------------------------------------+

//| Close trades function |

//+------------------------------------------------------------------+

Void CloseTrades()

{

For (int I = 0; I < OrdersTotal(); i++)

{

If (OrderSelect(I, SELECT\_BY\_POS) && OrderMagicNumber() == MagicNumber && OrderSymbol() == Symbol())

{

OrderClose(OrderTicket(), OrderLots(), OrderClosePrice(), 3);

// Send notification

SendNotification(NotificationEmail, NotificationPhone,

}

}

}

//+------------------------------------------------------------------+

//| Send notification function |

//+------------------------------------------------------------------+

Void SendNotification(string email, string phone, string message)

{SendMail(NotificationEmail, “Daily Profit Target Reached”, message);

// Send SMS notification

SendSMS(NotificationPhone, message);

}

SendMail(email, “Notification”, message);

SendSMS(phone, message);

}

// Check goal-based profit target

If (GoalBasedProfit && GoalProfitAmount > 0) {

If (totalProfit >= GoalProfitAmount) {

String message = “Goal-Based Profit Target Reached: $” + DoubleToString(GoalProfitAmount, 2);

SendNotification(NotificationEmail, NotificationPhone, message);

}

}

}

//+------------------------------------------------------------------+

//| Expert deinitialization function |

//+------------------------------------------------------------------+

Void OnDeinit(const int reason)

{

// Delete chart panel

ObjectDelete(“ChartPanel”);

// Delete labels

ObjectDelete(“PairLabel”);

ObjectDelete(“TimeframeLabel”);

ObjectDelete(“FloatingPLLabel”);

ObjectDelete(“DailyProfitTargetLabel”);

ObjectDelete(“DailyStopLossLabel”);

ObjectDelete(“RiskPerTradeLabel”);

// Delete lines

ObjectDelete(“BuyLine”);

ObjectDelete(“SellLine”);

}

```