<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Pro Market Analyzer</title>

<style>

body { font-family: Arial, sans-serif; padding: 20px; max-width: 1000px; margin: 0 auto; background: #1a1a1a; color: #fff; }

.container { display: grid; grid-template-columns: repeat(3, 1fr); gap: 15px; }

.signal-box {

padding: 20px; margin: 10px; border-radius: 8px; text-align: center;

box-shadow: 0 4px 6px rgba(0,0,0,0.3); transition: all 0.3s ease;

}

.buy { background: linear-gradient(145deg, #006400, #00aa00); }

.sell { background: linear-gradient(145deg, #8B0000, #ff0000); }

.neutral { background: linear-gradient(145deg, #404040, #606060); }

.chart {

height: 300px; background: #2a2a2a; padding: 15px;

border-radius: 8px; grid-column: 1 / -1;

}

.indicator {

padding: 15px; background: #333; border-radius: 5px;

margin: 5px 0; text-align: left;

}

.market-controls { margin: 20px 0; display: flex; gap: 10px; }

button { padding: 8px 15px; border: none; border-radius: 4px; cursor: pointer; }

.timeframe-btn.active { background: #007bff; color: white; }

</style>

</head>

<body>

<h1>Professional Binary Market Analyzer</h1>

<div class="market-controls">

<select id="assetSelect">

<option>BTC/USD</option>

<option>EUR/USD</option>

<option>Gold</option>

<option>OTC Index</option>

</select>

<button class="timeframe-btn active" data-tf="1">1M</button>

<button class="timeframe-btn" data-tf="5">5M</button>

<button id="OTCtoggle">OTC Mode: OFF</button>

</div>

<div class="container">

<div class="signal-box neutral" id="mainSignal">

<h3 id="signalText">Analyzing Market...</h3>

<div id="confidence"></div>

<div id="signalReasons"></div>

</div>

<div class="indicator">

<h4>Next Candle Prediction</h4>

<div id="nextCandle"></div>

<div id="priceTargets"></div>

</div>

<div class="indicator">

<h4>Key Levels</h4>

<div>Support: <span id="supportLevel"></span></div>

<div>Resistance: <span id="resistanceLevel"></span></div>

<div>Current Spot: <span id="spotPrice"></span></div>

</div>

<div class="chart" id="priceChart">

<!-- Would integrate tradingview widget in real implementation -->

Price Chart Area

</div>

</div>

<script>

const state = {

price: 100,

history: [],

trend: 'neutral',

volatility: 0.5,

otcMode: false,

currentTimeframe: 1

};

// Advanced market simulation parameters

const STRATEGY\_PARAMS = {

rsiWeights: { oversold: 0.3, overbought: 0.25 },

macdWeights: { crossover: 0.25 },

priceAction: { supportResistance: 0.2 },

volumeWeight: 0.15,

trendWeight: 0.3

};

function generateMarketData() {

const isOTC = state.otcMode;

const baseVolatility = isOTC ? 0.8 : 0.4;

const volatility = baseVolatility \* (1 + Math.random()\*0.5);

const priceChange = (Math.random()\*2 - 1) \* volatility;

const newPrice = state.price \* (1 + priceChange/100);

state.price = newPrice;

// Simulate professional indicators

const rsi = calculateAdvancedRSI();

const macd = calculateMACD();

const candles = analyzeCandlePatterns();

// Update support/resistance

updateKeyLevels(newPrice);

return {

time: new Date(),

price: newPrice.toFixed(4),

rsi: rsi,

macd: macd,

pattern: candles.pattern,

probability: candles.probability,

support: state.support,

resistance: state.resistance,

volume: (Math.random() \* 1000 + 500).toFixed(2)

};

}

function calculateAdvancedRSI() {

// Advanced RSI calculation with smoothing

const period = 14;

if(state.history.length < period) return 50;

let gains = 0, losses = 0;

for(let i = 1; i <= period; i++) {

const change = state.history[state.history.length - i].price -

state.history[state.history.length - i - 1].price;

if(change > 0) gains += change;

else losses += Math.abs(change);

}

const rs = gains / losses;

return 100 - (100 / (1 + rs));

}

function analyzeCandlePatterns() {

const patterns = ['Bullish Engulfing', 'Bearish Harami', 'Doji', 'Hammer', 'Shooting Star'];

const lastCandle = state.history[state.history.length - 1];

// Simplified pattern detection

let detectedPattern = patterns[Math.floor(Math.random()\*patterns.length)];

let probability = Math.random() \* 0.8 + 0.2; // Between 20-100%

return { pattern: detectedPattern, probability: probability.toFixed(2) };

}

function updateKeyLevels(currentPrice) {

// Track last 50 prices for key levels

const lookback = 50;

const prices = state.history.slice(-lookback).map(p => parseFloat(p.price));

state.support = Math.min(...prices).toFixed(4);

state.resistance = Math.max(...prices).toFixed(4);

}

function calculateSignal(data) {

let score = 0;

const reasons = [];

// RSI Analysis

if(data.rsi < 30) {

score += STRATEGY\_PARAMS.rsiWeights.oversold \* 100;

reasons.push(`RSI Oversold (${data.rsi.toFixed(1)})`);

}

if(data.rsi > 70) {

score -= STRATEGY\_PARAMS.rsiWeights.overbought \* 100;

reasons.push(`RSI Overbought (${data.rsi.toFixed(1)})`);

}

// MACD Analysis

if(data.macd.histogram > 0) {

score += STRATEGY\_PARAMS.macdWeights.crossover \* 100;

reasons.push(`MACD Bullish Crossover (${data.macd.histogram.toFixed(2)})`);

} else {

score -= STRATEGY\_PARAMS.macdWeights.crossover \* 100;

reasons.push(`MACD Bearish Crossover (${data.macd.histogram.toFixed(2)})`);

}

// Price Action Analysis

const price = parseFloat(data.price);

const proximityToSupport = (price - data.support)/(data.resistance - data.support);

if(proximityToSupport < 0.1) {

score += STRATEGY\_PARAMS.priceAction.supportResistance \* 100;

reasons.push(`Near Support (${data.support})`);

}

if(proximityToSupport > 0.9) {

score -= STRATEGY\_PARAMS.priceAction.supportResistance \* 100;

reasons.push(`Near Resistance (${data.resistance})`);

}

// Pattern Recognition

const patternStrength = data.pattern.probability \* 100;

if(['Bullish Engulfing', 'Hammer'].includes(data.pattern.pattern)) {

score += patternStrength \* 0.01;

reasons.push(`${data.pattern.pattern} (${patternStrength}%)`);

} else {

score -= patternStrength \* 0.01;

reasons.push(`${data.pattern.pattern} (${patternStrength}%)`);

}

// Determine Signal

const confidence = Math.min(Math.abs(score), 100);

const direction = score > 10 ? 'BUY' : score < -10 ? 'SELL' : 'NEUTRAL';

return { direction, confidence, reasons };

}

// UI Update Functions

function updateDisplay(data, signal) {

// Update Main Signal

const signalBox = document.getElementById('mainSignal');

signalBox.className = `signal-box ${signal.direction.toLowerCase()}`;

document.getElementById('signalText').textContent = `${signal.direction} | ${signal.confidence}% Confidence`;

// Update Reasons

const reasonsDiv = document.getElementById('signalReasons');

reasonsDiv.innerHTML = signal.reasons.map(r => `<div>✓ ${r}</div>`).join('');

// Update Key Levels

document.getElementById('supportLevel').textContent = data.support;

document.getElementById('resistanceLevel').textContent = data.resistance;

document.getElementById('spotPrice').textContent = data.price;

// Update Next Candle Prediction

const nextCandleDiv = document.getElementById('nextCandle');

const bullishProb = 50 + (signal.confidence \* (signal.direction === 'BUY' ? 1 : -1));

nextCandleDiv.innerHTML = `

<div>Bullish: ${bullishProb.toFixed(0)}%</div>

<div>Bearish: ${(100 - bullishProb).toFixed(0)}%</div>

<div>Expected Range: ±${(state.volatility \* 100).toFixed(1)}%</div>

`;

}

// Simulate real-time updates

setInterval(() => {

const marketData = generateMarketData();

const signal = calculateSignal(marketData);

state.history.push(marketData);

updateDisplay(marketData, signal);

}, 1500);

// Event Listeners

document.querySelectorAll('.timeframe-btn').forEach(btn => {

btn.addEventListener('click', () => {

document.querySelectorAll('.timeframe-btn').forEach(b => b.classList.remove('active'));

btn.classList.add('active');

state.currentTimeframe = parseInt(btn.dataset.tf);

});

});

document.getElementById('OTCtoggle').addEventListener('click', function() {

state.otcMode = !state.otcMode;

this.textContent = `OTC Mode: ${state.otcMode ? 'ON' : 'OFF'}`;

});

</script>

</body>

</html>