<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" />

<title>Futures & Options AI Trading Assistant</title>

<style>

/\* Reset and base \*/

\* {

box-sizing: border-box;

}

body {

margin: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: linear-gradient(135deg, #0f2027, #203a43, #2c5364);

color: #eee;

display: flex;

flex-direction: column;

align-items: center;

max-width: 350px;

margin: auto;

padding: 10px 10px 30px;

height: 600px;

overflow: hidden;

}

h1 {

font-weight: 900;

font-size: 1.6rem;

margin: 10px 0 5px;

text-align: center;

letter-spacing: 1.8px;

color: #00d8ff;

text-shadow: 0 0 10px #00d8ff99;

}

h2 {

font-weight: 700;

font-size: 1.1rem;

margin: 3px 0 20px;

color: #00ffc6cc;

text-align: center;

}

label {

font-size: 0.9rem;

margin-right: 10px;

user-select: none;

}

select, button {

font-size: 1rem;

padding: 6px 12px;

border-radius: 6px;

border: none;

cursor: pointer;

font-weight: 700;

transition: background-color 0.25s ease;

}

select {

background-color: #074057;

color: #00d9ff;

box-shadow: 0 0 7px #00d9ff44;

}

button {

background-color: #00d8ff;

color: #002a34;

margin-left: 10px;

}

button:hover {

background-color: #00ffc6;

}

.controls {

display: flex;

justify-content: center;

align-items: center;

margin-bottom: 15px;

flex-wrap: wrap;

}

.chart-container {

position: relative;

width: 320px;

height: 320px;

border-radius: 15px;

background: rgba(20, 30, 40, 0.9);

box-shadow: 0 0 30px #00d8ff88;

padding: 10px;

}

#tradingChart {

width: 100% !important;

height: 100% !important;

border-radius: 12px;

}

.info-panel {

margin-top: 15px;

background: rgba(0, 216, 255, 0.1);

border-radius: 12px;

padding: 12px 16px;

color: #00e1ff;

font-size: 0.9rem;

max-height: 195px;

overflow-y: auto;

box-shadow: inset 0 0 12px #00d8ff88;

}

.info-panel h3 {

margin-top: 0;

font-weight: 900;

color: #00ffffdd;

text-align: center;

}

.signal {

font-weight: 900;

font-size: 1.3rem;

margin: 10px 0 3px;

text-align: center;

user-select: none;

}

.signal.buy {

color: #00ffcc;

text-shadow: 0 0 8px #00ffccaa;

}

.signal.sell {

color: #ff4c4c;

text-shadow: 0 0 10px #ff4c4caa;

}

.footer {

margin-top: auto;

font-size: 0.75rem;

color: #00aaccee;

text-align: center;

user-select: none;

}

@media (max-width: 350px) {

body {

max-width: 100vw;

padding: 5px 5px 20px;

height: 100vh;

}

.chart-container {

width: 100vw;

height: 280px;

padding: 8px;

}

}

</style>

</head>

<body>

<h1>F&O AI Trading Assistant</h1>

<h2>Real-Time Put/Call Prediction & Signals</h2>

<div class="controls" role="region" aria-label="Trading controls">

<label for="optionType">Option Type</label>

<select id="optionType" aria-describedby="optionTypeHelp">

<option value="call">Call Option</option>

<option value="put">Put Option</option>

</select>

<button id="startBtn" aria-live="polite" aria-label="Start Simulation">Start Simulation</button>

<button id="stopBtn" disabled aria-live="polite" aria-label="Stop Simulation">Stop Simulation</button>

</div>

<div class="chart-container" role="img" aria-label="Trading price chart with indicators and signals">

<canvas id="tradingChart" width="320" height="320"></canvas>

</div>

<div class="info-panel" role="region" aria-live="polite" aria-atomic="true">

<h3>AI Trading Signal</h3>

<div id="signal" class="signal" aria-live="assertive" aria-atomic="true">Start simulation to see signals</div>

<div id="details"></div>

</div>

<div class="footer">

&copy; 2024 AI-Driven Trading Assistant | Data simulated for demo purposes

</div>

<script src="https://cdn.jsdelivr.net/npm/chart.js@4.3.0/dist/chart.umd.min.js"></script>

<script>

const ctx = document.getElementById('tradingChart').getContext('2d');

// Helper functions for indicators

function SMA(data, window\_size) {

let r = [];

for (let i = 0; i< data.length; i++) {

if (i< window\_size - 1) {

r.push(null);

continue;

}

let sum = 0;

for (let j = i - window\_size + 1; j<= i; j++) {

sum += data[j];

}

r.push(sum / window\_size);

}

return r;

}

function RSI(data, period = 14) {

let rsi = new Array(data.length).fill(null);

let gains = 0, losses = 0;

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

let delta = data[i] - data[i-1];

if (delta > 0) gains += delta;

else losses -= delta;

}

let avgGain = gains / period;

let avgLoss = losses / period;

if (avgLoss === 0) {

for (let k = period; k< data.length; k++) rsi[k] = 100;

return rsi;

}

let rs = avgGain / avgLoss;

rsi[period] = 100 - (100 / (1 + rs));

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

let delta = data[i] - data[i-1];

let gain = delta > 0 ? delta : 0;

let loss = delta< 0 ? -delta : 0;

avgGain = (avgGain \* (period -1) + gain) / period;

avgLoss = (avgLoss \* (period -1) + loss) / period;

rs = avgGain / avgLoss;

rsi[i] = 100 - (100 / (1 + rs));

}

return rsi;

}

function MACD(data, shortPeriod = 12, longPeriod = 26, signalPeriod = 9) {

// EMA helper

function EMA(arr, period) {

let k = 2 / (period + 1);

let emaArray = [];

emaArray[0] = arr[0];

for (let i = 1; i< arr.length; i++) {

emaArray[i] = arr[i] \* k + emaArray[i -1] \* (1 - k);

}

return emaArray;

}

const shortEMA = EMA(data, shortPeriod);

const longEMA = EMA(data, longPeriod);

let macd = new Array(data.length).fill(null);

for (let i = 0; i< data.length; i++) {

if (shortEMA[i] !== undefined && longEMA[i] !== undefined) {

macd[i] = shortEMA[i] - longEMA[i];

}

}

const signalLine = EMA(macd.filter(v => v !== null), signalPeriod);

let signalLineFull = new Array(data.length).fill(null);

// Fill signalLineFull aligned to macd starting from longPeriod + signalPeriod index roughly

for(let i = (longPeriod + signalPeriod\*2); i< data.length; i++) {

signalLineFull[i] = signalLine[i - (longPeriod + signalPeriod\*2)] || null;

}

return { macd, signalLine: signalLineFull };

}

// Bollinger Bands

function BollingerBands(data, window = 20, stdMultiplier = 2) {

let middleBand = SMA(data, window);

let bands = {upper: [], middle: middleBand, lower: []};

for (let i = 0; i< data.length; i++) {

if (i< window - 1) {

bands.upper.push(null);

bands.lower.push(null);

continue;

}

let slice = data.slice(i - window + 1, i + 1);

let mean = middleBand[i];

let variance = slice.reduce((acc, val) => acc + (val - mean) \*\* 2, 0) / window;

let std = Math.sqrt(variance);

bands.upper.push(mean + stdMultiplier \* std);

bands.lower.push(mean - stdMultiplier \* std);

}

return bands;

}

// Generate simulated price data (random walk with volatility)

function generatePriceData(length=100, start=100, volatility=2) {

let prices = [start];

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

let change = (Math.random() - 0.5) \* volatility;

prices.push(Math.max(1, prices[i-1] + change));

}

return prices;

}

// AI heuristic prediction for buy/sell based on indicators for demonstration

function aiPredictSignals(data, optionType) {

const sma20 = SMA(data, 20);

const sma50 = SMA(data, 50);

const rsi = RSI(data);

const macdData = MACD(data);

const bb = BollingerBands(data);

// Signals object will have buy/sell or hold with reason

let signals = [];

for (let i = 0; i< data.length; i++) {

if (i< 50) {

signals.push({signal: 'hold', reason: 'Insufficient data for indicators'});

continue;

}

let signal = 'hold';

let reasons = [];

// Strategy example: For call options buy signal when:

// Price above SMA50 and SMA20 crossing above SMA50, RSI below 70 (not overbought), MACD line crossing signal line upwards

// For put options buy signal when:

// Price below SMA50 and SMA20 crossing below SMA50, RSI above 30 (not oversold), MACD line crossing signal line downwards

// Also use Bollinger Bands breakout for extra signals.

let price = data[i];

let prevPrice = data[i-1];

let sma20Cur = sma20[i], sma50Cur = sma50[i];

let sma20Prev = sma20[i-1], sma50Prev = sma50[i-1];

let rsiCur = rsi[i];

let macdCur = macdData.macd[i];

let signalLineCur = macdData.signalLine[i];

let macdPrev = macdData.macd[i-1];

let signalLinePrev = macdData.signalLine[i-1];

let bbUpper = bb.upper[i];

let bbLower = bb.lower[i];

// Detect SMA20 crossing SMA50

let smaCrossUp = sma20Prev && sma50Prev && (sma20Prev< sma50Prev) && (sma20Cur > sma50Cur);

let smaCrossDown = sma20Prev && sma50Prev && (sma20Prev > sma50Prev) && (sma20Cur< sma50Cur);

// Detect MACD crossing signal line

let macdCrossUp = macdPrev && signalLinePrev && (macdPrev< signalLinePrev) && (macdCur > signalLineCur);

let macdCrossDown = macdPrev && signalLinePrev && (macdPrev > signalLinePrev) && (macdCur< signalLineCur);

if (optionType === 'call') {

// Buy signal conditions

if (price > sma50Cur && smaCrossUp && rsiCur< 70 && macdCrossUp) {

signal = 'buy';

reasons.push('Price > SMA50, SMA20 crosses above SMA50, RSI < 70, MACD crosses signal line up');

} else if (price > bbUpper) {

// Overbought condition - sell signal

signal = 'sell';

reasons.push('Price above upper Bollinger Band - potential overbought');

} else if (price< sma50Cur && smaCrossDown && macdCrossDown) {

signal = 'sell';

reasons.push('Price < SMA50, SMA20 crosses below SMA50, MACD crosses signal line down');

}

} else {

// Put option

if (price< sma50Cur && smaCrossDown && rsiCur > 30 && macdCrossDown) {

signal = 'buy';

reasons.push('Price < SMA50, SMA20 crosses below SMA50, RSI > 30, MACD crosses signal line down');

} else if (price< bbLower) {

signal = 'sell';

reasons.push('Price below lower Bollinger Band - potential oversold');

} else if (price > sma50Cur && smaCrossUp && macdCrossUp) {

signal = 'sell';

reasons.push('Price > SMA50, SMA20 crosses above SMA50, MACD crosses signal line up');

}

}

if (signal === 'hold') reasons.push('No strong signals detected');

signals.push({signal, reason: reasons.join('; ')});

}

return signals;

}

// Application state

let priceData = [];

let signals = [];

let optionType = 'call';

let simulationInterval = null;

let maxDataPoints = 100;

// Chart.js setup

const chartConfig = {

type: 'line',

data: {

labels: [],

datasets: [

{

label: 'Price',

data: [],

borderColor: '#00d8ff',

backgroundColor: 'rgba(0, 216, 255, 0.3)',

pointRadius: 0,

borderWidth: 2,

tension: 0.3,

yAxisID: 'y',

},

{

label: 'SMA 20',

data: [],

borderColor: '#ffcc00',

borderDash: [6, 4],

pointRadius: 0,

borderWidth: 1.5,

tension: 0.4,

yAxisID: 'y',

},

{

label: 'SMA 50',

data: [],

borderColor: '#33ff88',

borderDash: [4, 4],

pointRadius: 0,

borderWidth: 1.5,

tension: 0.4,

yAxisID: 'y',

},

{

label: 'Upper Bollinger Band',

data: [],

borderColor: '#ff0055',

borderWidth: 1,

borderDash: [8, 8],

fill: false,

pointRadius: 0,

tension: 0.1,

yAxisID: 'y',

},

{

label: 'Lower Bollinger Band',

data: [],

borderColor: '#ff0055',

borderWidth: 1,

borderDash: [8, 8],

fill: '-1',

pointRadius: 0,

tension: 0.1,

yAxisID: 'y',

}

],

},

options: {

animation: false,

maintainAspectRatio: false,

interaction: {

mode: 'nearest',

intersect: false,

},

scales: {

y: {

ticks: { color: '#00d8ff' },

border: { color: '#00d8ff' },

},

x: {

ticks: { color: '#00d8ff' },

border: { color: '#00d8ff' },

display: true,

}

},

plugins: {

legend: { labels: { color: '#00d8ff' } },

tooltip: {

backgroundColor: '#0a2731',

titleColor: '#00d8ff',

bodyColor: '#aaffff',

usePointStyle: true,

cornerRadius: 8,

callbacks: {

label: context => {

return context.dataset.label + ': ' + context.parsed.y.toFixed(2);

}

}

}

}

}

};

const tradingChart = new Chart(ctx, chartConfig);

// Update chart datasets

function updateChart() {

tradingChart.data.labels = priceData.map((\_, i) => i + 1);

tradingChart.data.datasets[0].data = priceData;

tradingChart.data.datasets[1].data = SMA(priceData, 20);

tradingChart.data.datasets[2].data = SMA(priceData, 50);

const bb = BollingerBands(priceData);

tradingChart.data.datasets[3].data = bb.upper;

tradingChart.data.datasets[4].data = bb.lower;

tradingChart.update('none');

}

// Update signal info panel

function updateSignalInfo(latestIndex) {

const signalElem = document.getElementById('signal');

const detailsElem = document.getElementById('details');

if (!signals.length || latestIndex == null || latestIndex< 0) {

signalElem.textContent = 'No signal data available';

detailsElem.textContent = '';

signalElem.className = 'signal';

return;

}

const currentSignal = signals[latestIndex];

signalElem.textContent = currentSignal.signal.toUpperCase();

if (currentSignal.signal === 'buy') {

signalElem.className = 'signal buy';

} else if (currentSignal.signal === 'sell') {

signalElem.className = 'signal sell';

} else {

signalElem.className = 'signal';

}

detailsElem.textContent = currentSignal.reason;

}

// Run simulation step - generate new data point and update signals, chart & info

function simulationStep() {

// Random walk next price

const lastPrice = priceData[priceData.length - 1];

let volatility = 2 + Math.random(); // dynamic volatility

let change = (Math.random() - 0.5) \* volatility;

let nextPrice = Math.max(1, lastPrice + change);

priceData.push(nextPrice);

if (priceData.length > maxDataPoints) priceData.shift();

signals = aiPredictSignals(priceData, optionType);

updateChart();

updateSignalInfo(signals.length - 1);

}

// Start/stop simulation handlers

document.getElementById('startBtn').addEventListener('click', () => {

optionType = document.getElementById('optionType').value;

// Initialize starting data

priceData = generatePriceData(maxDataPoints, 100, 2);

signals = aiPredictSignals(priceData, optionType);

updateChart();

updateSignalInfo(signals.length - 1);

document.getElementById('startBtn').disabled = true;

document.getElementById('stopBtn').disabled = false;

document.getElementById('optionType').disabled = true;

// Begin simulation updates every 2 seconds

simulationInterval = setInterval(simulationStep, 2000);

});

document.getElementById('stopBtn').addEventListener('click', () => {

clearInterval(simulationInterval);

simulationInterval = null;

document.getElementById('startBtn').disabled = false;

document.getElementById('stopBtn').disabled = true;

document.getElementById('optionType').disabled = false;

});

</script>

</body>

</html>