Build a Crypto Trading Bot Using TradingView Webhooks and Python

MUHAMMAD UMER KHAN 22I-0780
ABDUL MUNHIM HUSSAIN 22I-1021
MUHAMMAD HAMZA IMRAN 22I-0865

OVERVIEW OF TRADING STRATEGY

The automated trading bot implements a multi-indicator technical analysis strategy that generates buy and sell signals based on the convergence of four independent technical indicators. The strategy operates on the principle that multiple confirmations reduce false signals and improve trade quality.

1. STRATEGY APPROACH:

The bot uses a conservative "all-or-nothing" approach where a signal is only generated when ALL four technical indicators align in the same direction. This ensures high-quality signals with reduced false positives, though at the cost of lower signal frequency.

2. TECHNICAL INDICATORS USED:

RSI (Relative Strength Index) - Measures momentum and identifies overbought (>70) and oversold (<30) conditions.

EMA Crossover - Fast EMA (12 periods) and Slow EMA (26 periods) to identify trend direction.

MACD (Moving Average Convergence Divergence) - Detects momentum changes and trend reversals.

Bollinger Bands - Identifies volatility and price levels relative to standard deviations.

3. BUY SIGNAL CONDITIONS (ALL must be true):

Fast EMA crosses above Slow EMA (bullish trend)

MACD line crosses above signal line (positive momentum)

RSI is oversold (<30) or recovering (<50)

Price is at or below lower Bollinger Band (support level)

4. **SELL SIGNAL (ALL must be true):**

Fast EMA crosses below Slow EMA

MACD line crosses below signal line

RSI is overbought (>70) or declining (>50)

Price is at or above upper Bollinger Band

EXPLANATION OF PINE SCRIPT

The Pine Script v5 strategy is structured into four main components:

1. **INPUT PARAMETERS:**

RSI: length 14, overbought 70, oversold 30

EMA: fast 12 periods, slow 26 periods

MACD: fast 12, slow 26, and signal 9

Bollinger Bands: length 20, multiplier 2.0

2. **INDICATOR CALCULATION:**

rsi = ta.rsi(close, rsi_length)

ema_fast_line = ta.ema(close, ema_fast)

ema_slow_line = ta.ema(close, ema_slow)

[macd_line, signal_line, hist_line] = ta.macd(close, macd_fast, macd_slow, macd_signal)

[bb_upper, bb_middle, bb_lower] = ta.bb(close, bb_length, bb_mult)

3. **SIGNAL GENERATION:**

Buy and sell signals are generated when ALL four conditions align:

Buy Signal = (EMA crossover) AND (MACD bullish) AND (RSI oversold) AND (Price at BB lower)

Sell Signal = (EMA crossunder) AND (MACD bearish) AND (RSI overbought) AND (Price at BB upper)

Code implementation:

buy_signal = ema_bullish and macd_bullish and rsi_bullish and bb_bullish

sell_signal = ema_bearish and macd_bearish and rsi_bearish and bb_bearish

4. **ALERT SYSTEM:**

When a signal is confirmed, the alert() function sends a JSON-formatted message:

```
if (barstate.isconfirmed and buy_signal)
    alert('{"signal": "buy", "symbol": "' + syminfo.ticker + '",
        "price": ' + str.tostring(close) + '}', alert.freq_once_per_bar)
```

Key features:

- **barstate.isconfirmed** prevents duplicate alerts (fires once per completed bar)
- JSON format enables easy parsing by webhook server
- Visual indicators: Green triangles for buy signals, red triangles for sell signals

PYTHON CODE EXPLANATION

The Flask-based webhook server integrates with Binance Testnet API:

MAIN COMPONENTS:

1. **Configuration:**

Loads API keys from **.env**, initializes Binance client with testnet=True, sets up logging to file and console.

2. **Trade Execution:**

```
def execute_buy_order(symbol, quantity):
    order = client.create_order(
        symbol=symbol, side=Client.SIDE_BUY,
        type=Client.ORDER_TYPE_MARKET, quantity=quantity)
    return order
```

Similar function for sell orders. Both handle BinanceAPIException.

3. **Webhook Endpoint (/webhook):**

- Parses JSON/form/raw data formats

- Validates signal ("buy" or "sell")

- Checks balance (USDT for buy, BTC for sell)

- Executes trade via execute_buy_order() or execute_sell_order()

- Saves to trade_history.csv regardless of success/failure

- Returns JSON with status, order_id, quantity, timestamp

4. **Error Handling:**

- Invalid signals: Returns HTTP 400 with error message

- Insufficient balance: Catches exception, saves with error status

- API errors: Catches BinanceAPIException, logs detailed error

- All errors logged to trading_bot.log and saved to CSV

5. **Additional Endpoints:**

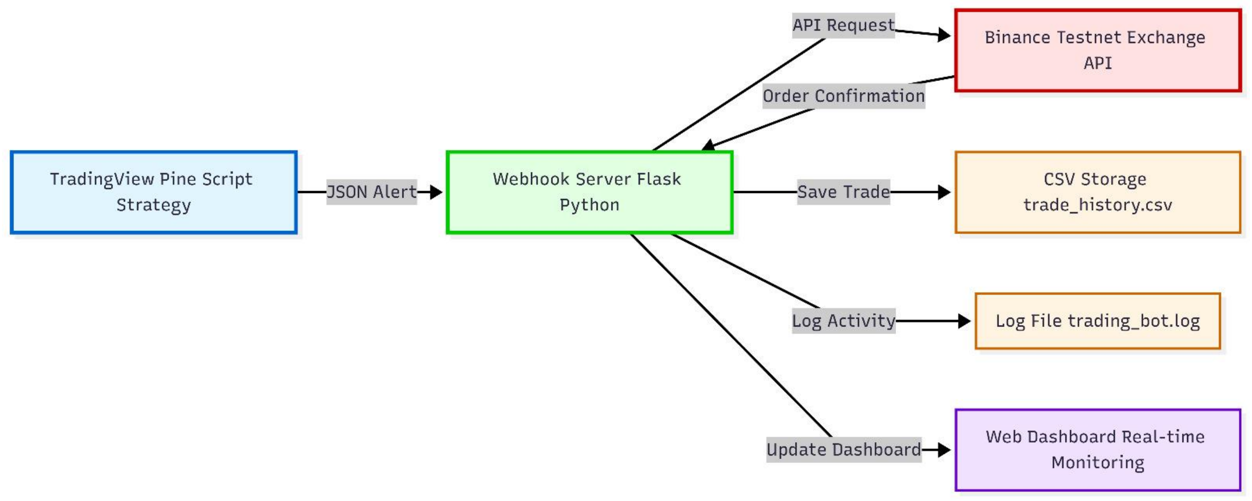
- /health: Server and Binance connection status

- /balance: Account balances for trading assets

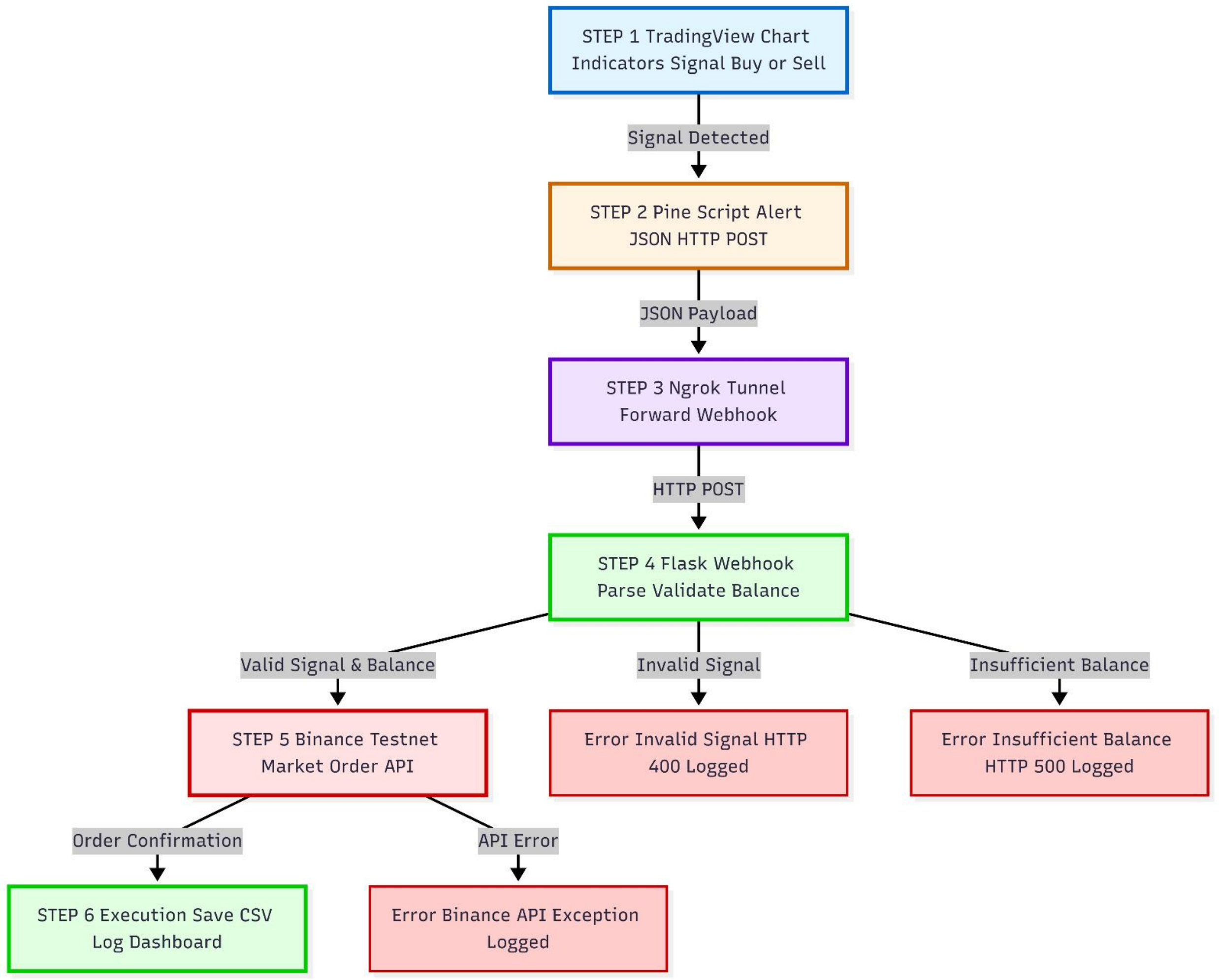
- /history: All trades from trade_history.csv

- /: Web dashboard (index.html)

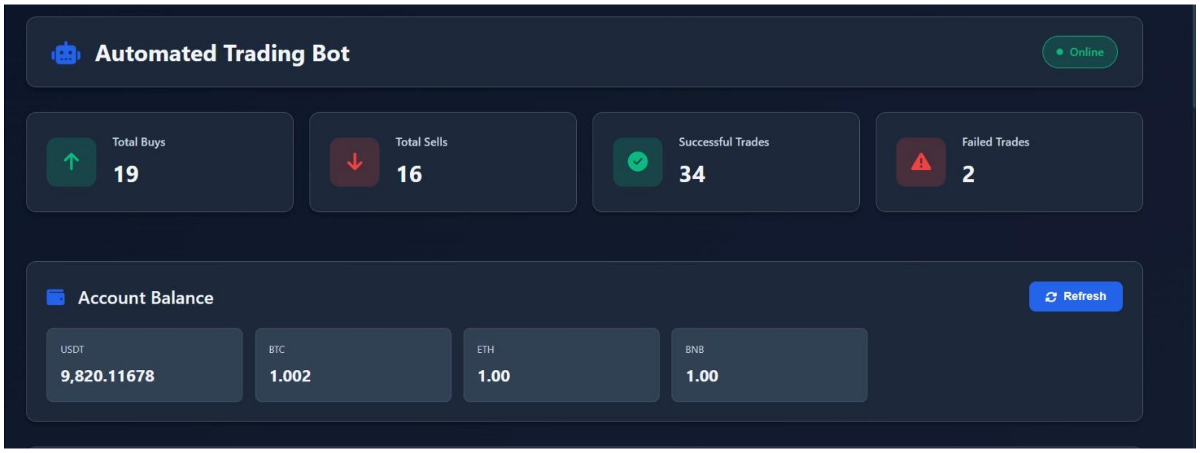
SYSTEM FLOW DIAGRAM


TradingView Pine Script Strategy
JSON Alert
Webhook Server Flask Python
API Request
Order Confirmation
Binance Testnet Exchange API
Save Trade
CSV Storage trade_history.csv
Log Activity
Log File trading_bot.log
Update Dashboard
Web Dashboard Real-time Monitoring

WEBHOOK WORKFLOW DIAGRAM


STEP 1 TradingView Chart Indicators Signal Buy or Sell
Signal Detected
STEP 2 Pine Script Alert JSON HTTP POST
JSON Payload
STEP 3 Ngrok Tunnel Forward Webhook
HTTP POST
STEP 4 Flask Webhook Parse Validate Balance
Valid Signal & Balance
Invalid Signal
Insufficient Balance
STEP 5 Binance Testnet Market Order API
Error Invalid Signal HTTP 400 Logged
Error Insufficient Balance HTTP 500 Logged
Order Confirmation
API Error
STEP 6 Execution Save CSV Log Dashboard
Error Binance API Exception Logged

SCREENSHOTS

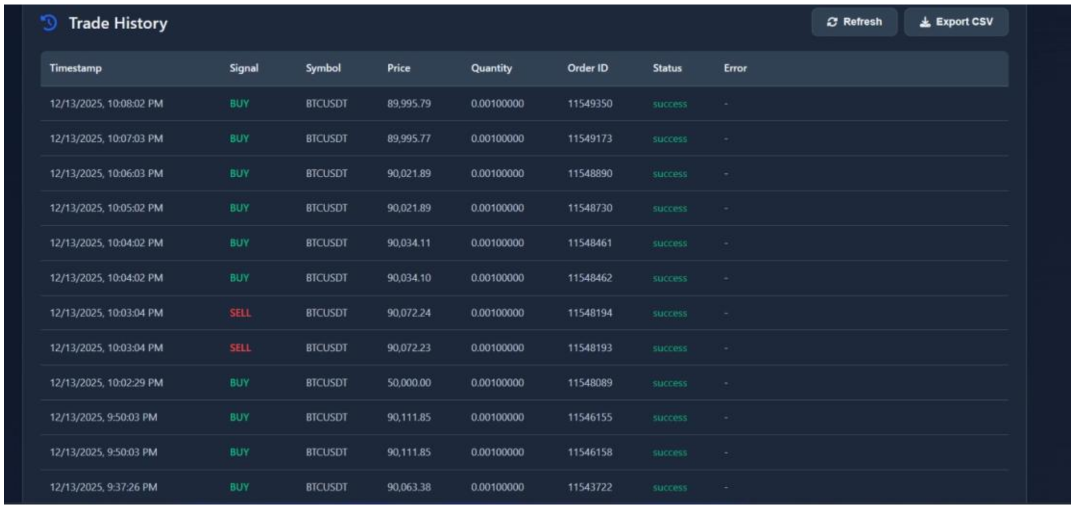

Automated Trading Bot

Total Buys	Total Sells	Successful Trades	Failed Trades
19	16	34	2

Account Balance

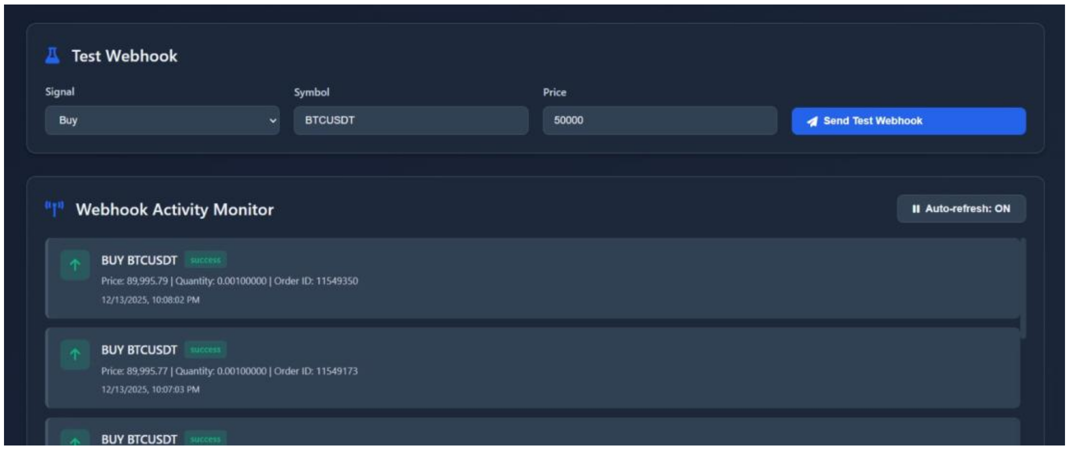
USDT	BTC	ETH	BNB
9,820.11678	1.002	1.00	1.00

Trading bot dashboard with statistics (19 buys, 16 sells, 34 successful) and account balances.


Trade History

Timestamp	Signal	Symbol	Price	Quantity	Order ID	Status	Error
12/13/2025, 10:08:02 PM	BUY	BTCUSDT	89,995.79	0.00100000	11549350	success	-
12/13/2025, 10:07:03 PM	BUY	BTCUSDT	89,995.77	0.00100000	11549173	success	-
12/13/2025, 10:06:03 PM	BUY	BTCUSDT	90,021.89	0.00100000	11548890	success	-
12/13/2025, 10:05:02 PM	BUY	BTCUSDT	90,021.89	0.00100000	11548730	success	-
12/13/2025, 10:04:02 PM	BUY	BTCUSDT	90,034.11	0.00100000	11548461	success	-
12/13/2025, 10:04:02 PM	BUY	BTCUSDT	90,034.10	0.00100000	11548462	success	-
12/13/2025, 10:03:04 PM	SELL	BTCUSDT	90,072.24	0.00100000	11548194	success	-
12/13/2025, 10:03:04 PM	SELL	BTCUSDT	90,072.23	0.00100000	11548193	success	-
12/13/2025, 10:02:29 PM	BUY	BTCUSDT	50,000.00	0.00100000	11548089	success	-
12/13/2025, 9:50:03 PM	BUY	BTCUSDT	90,111.85	0.00100000	11546155	success	-
12/13/2025, 9:50:03 PM	BUY	BTCUSDT	90,111.85	0.00100000	11546158	success	-
12/13/2025, 9:37:26 PM	BUY	BTCUSDT	90,063.38	0.00100000	11543722	success	-

Trade history table with successful BUY/SELL orders and order IDs.


Test Webhook

Signal: Buy
Symbol: BTCUSDT
Price: 50000

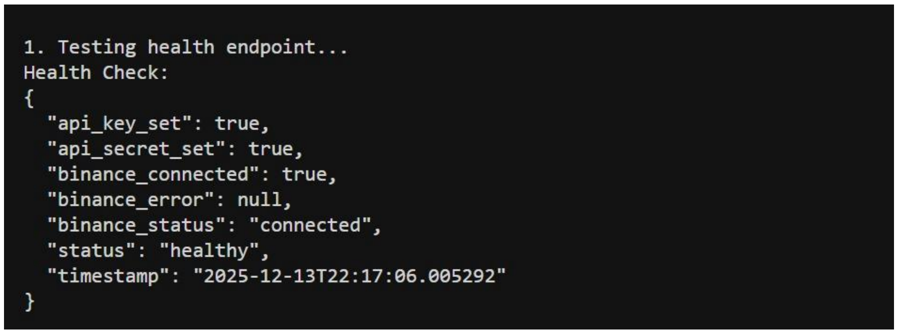
Webhook Activity Monitor

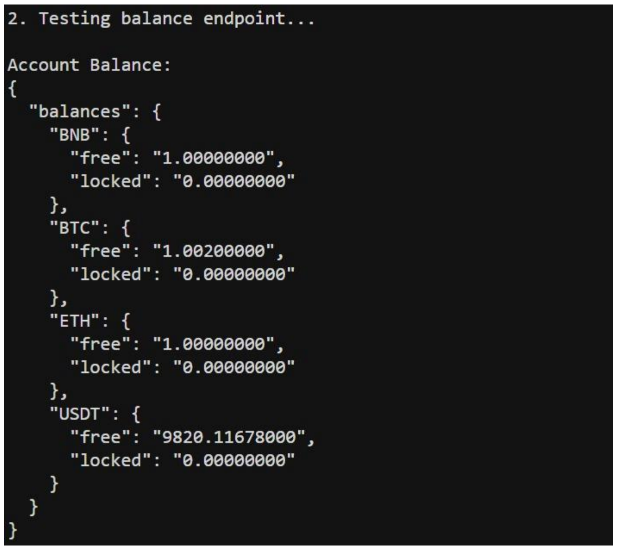
BUY BTCUSDT success
Price: 89,995.79 | Quantity: 0.00100000 | Order ID: 11549350
12/13/2025, 10:08:02 PM

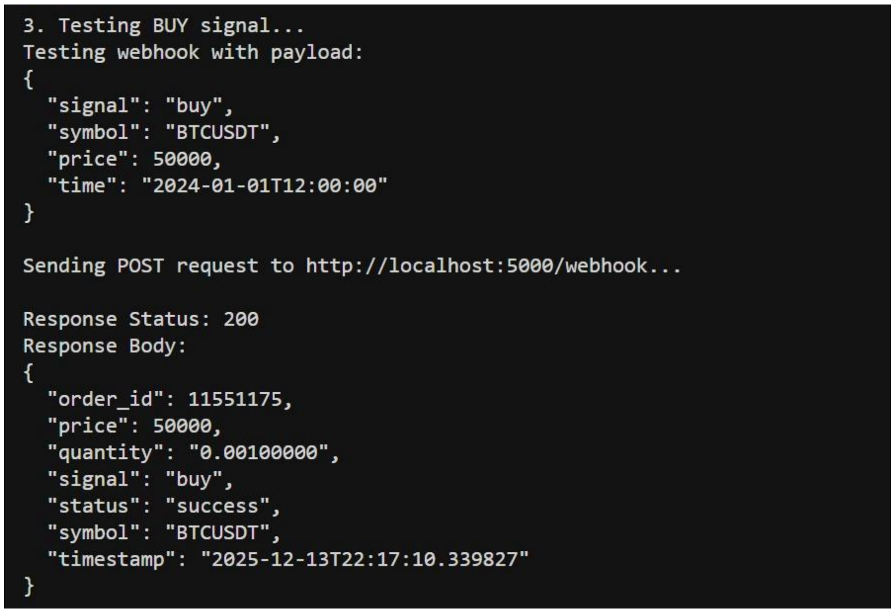
BUY BTCUSDT success
Price: 89,995.77 | Quantity: 0.00100000 | Order ID: 11549173
12/13/2025, 10:07:03 PM

BUY BTCUSDT success

Webhook testing interface and real-time activity monitor.

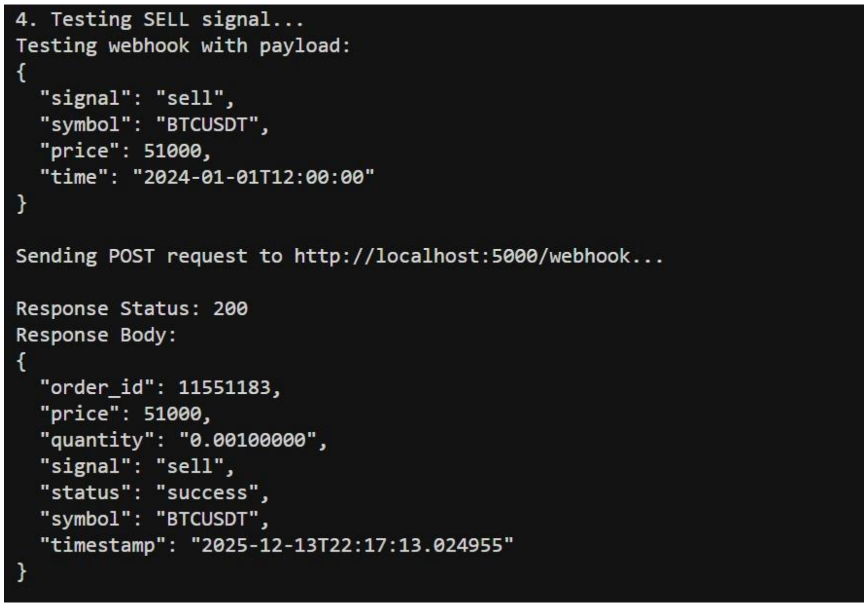

```
1. Testing health endpoint...
Health Check:
{
  "api_key_set": true,
  "api_secret_set": true,
  "binance_connected": true,
  "binance_error": null,
  "binance_status": "connected",
  "status": "healthy",
  "timestamp": "2025-12-13T22:17:06.005292"
}
```
Health check endpoint confirming Binance Testnet connection.


```
2. Testing balance endpoint...

Account Balance:
{
  "balances": {
    "BNB": {
      "free": "1.00000000",
      "locked": "0.00000000"
    },
    "BTC": {
      "free": "1.00200000",
      "locked": "0.00000000"
    },
    "ETH": {
      "free": "1.00000000",
      "locked": "0.00000000"
    },
    "USDT": {
      "free": "9820.11678000",
      "locked": "0.00000000"
    }
  }
}
```
Account balance endpoint showing testnet balances.



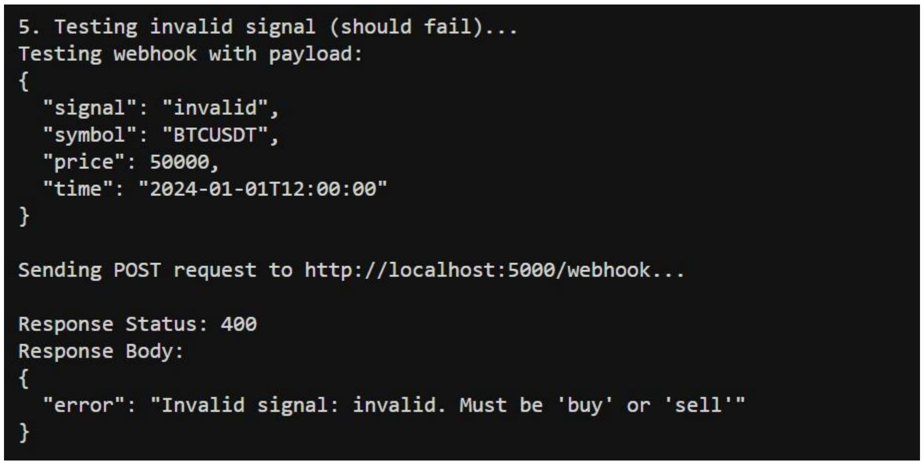
```
3. Testing BUY signal...
Testing webhook with payload:
{
  "signal": "buy",
  "symbol": "BTCUSDT",
  "price": 50000,
  "time": "2024-01-01T12:00:00"
}

Sending POST request to http://localhost:5000/webhook...

Response Status: 200
Response Body:
{
  "order_id": 11551175,
  "price": 50000,
  "quantity": "0.00100000",
  "signal": "buy",
  "status": "success",
  "symbol": "BTCUSDT",
  "timestamp": "2025-12-13T22:17:10.339827"
}
```

Successful BUY webhook test with HTTP 200 and order ID 11551175.



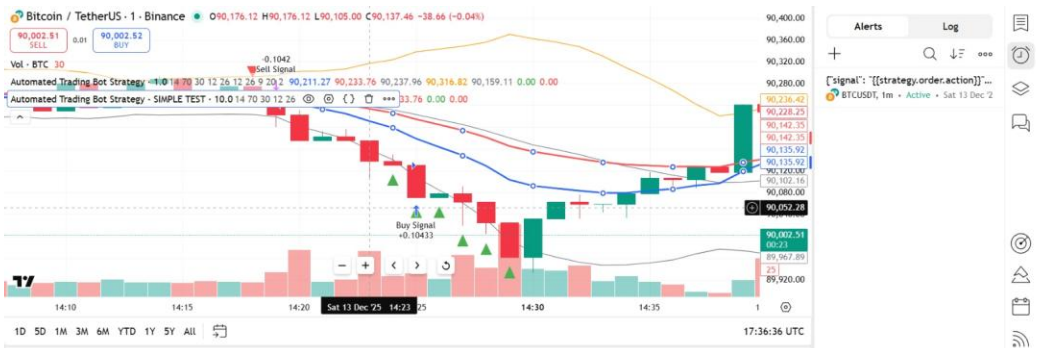
```
4. Testing SELL signal...
Testing webhook with payload:
{
  "signal": "sell",
  "symbol": "BTCUSDT",
  "price": 51000,
  "time": "2024-01-01T12:00:00"
}

Sending POST request to http://localhost:5000/webhook...

Response Status: 200
Response Body:
{
  "order_id": 11551183,
  "price": 51000,
  "quantity": "0.00100000",
  "signal": "sell",
  "status": "success",
  "symbol": "BTCUSDT",
  "timestamp": "2025-12-13T22:17:13.024955"
}
```

Successful SELL webhook test HTTP 200

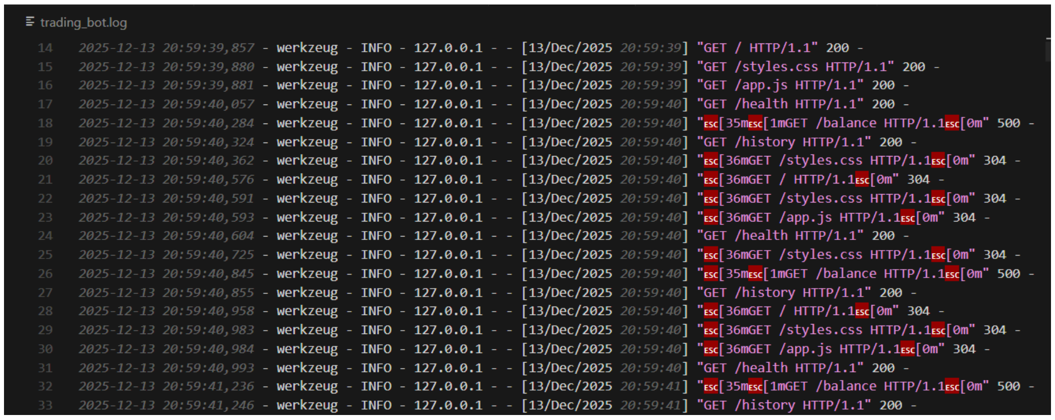


```
5. Testing invalid signal (should fail)...
Testing webhook with payload:
{
  "signal": "invalid",
  "symbol": "BTCUSDT",
  "price": 50000,
  "time": "2024-01-01T12:00:00"
}

Sending POST request to http://localhost:5000/webhook...

Response Status: 400
Response Body:
{
  "error": "Invalid signal: invalid. Must be 'buy' or 'sell'"
}
```

Error handling showing HTTP 400 for invalid signal.

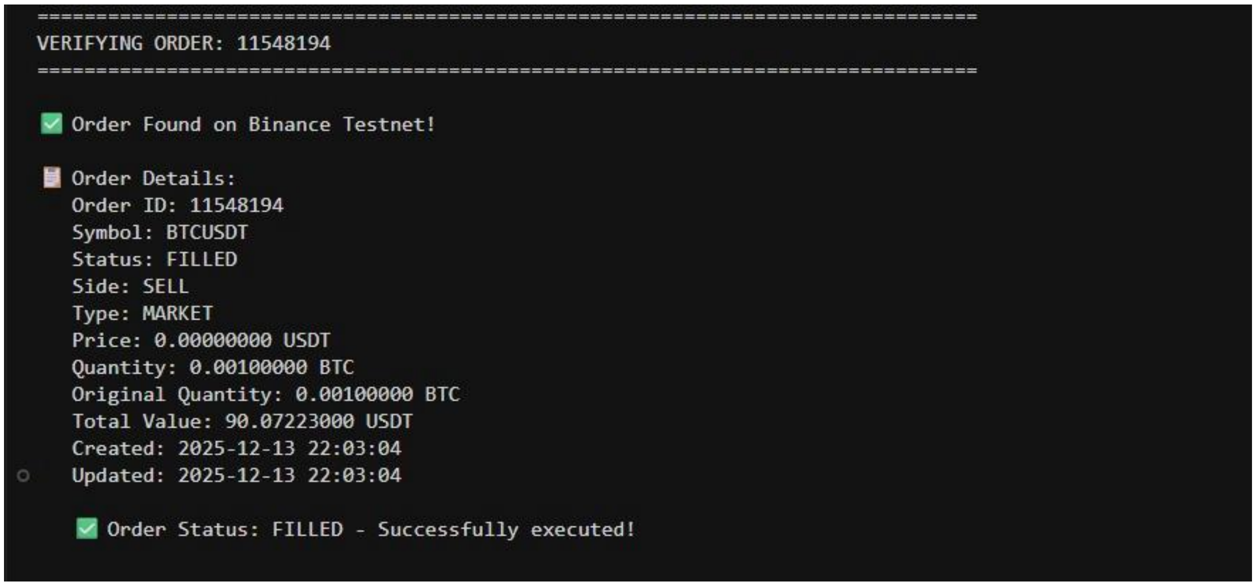

Bitcoin / TetherUS · 1 · Binance O90,176.12 H90,176.12 L90,105.00 C90,137.46 −38.66 (−0.04%)
90,002.51 SELL
0.01
90,002.52 BUY
Vol · BTC 30
−0.1042
Sell Signal
Automated Trading Bot Strategy · 1.0 14 70 30 12 26 12 26 9 20 2 90,211.27 90,233.76 90,237.96 90,316.82 90,159.11 0.00 0.00
Automated Trading Bot Strategy · SIMPLE TEST · 10.0 14 70 30 12 26
Buy Signal
+0.10433
Alerts
Log
{"signal": "{{strategy.order.action}}"...
BTCUSDT, 1m · Active · Sat 13 Dec '2
90,400.00
90,360.00
90,320.00
90,280.00
90,236.42
90,228.25
90,142.35
90,142.35
90,135.92
90,135.92
90,120.00
90,102.16
90,080.00
90,052.28
90,002.51
00:23
89,967.89
89,920.00
14:10
14:15
14:20
Sat 13 Dec '25 14:23
14:30
14:35
1D 5D 1M 3M 6M YTD 1Y 5Y All
17:36:36 UTC

TradingView bot signals



```
trading_bot.log
2025-12-13 20:59:39,857 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:39] "GET / HTTP/1.1" 200 -
2025-12-13 20:59:39,880 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:39] "GET /styles.css HTTP/1.1" 200 -
2025-12-13 20:59:39,881 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:39] "GET /app.js HTTP/1.1" 200 -
2025-12-13 20:59:40,057 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "GET /health HTTP/1.1" 200 -
2025-12-13 20:59:40,284 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "ESC[35mESC[1mGET /balance HTTP/1.1ESC[0m" 500 -
2025-12-13 20:59:40,324 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "GET /history HTTP/1.1" 200 -
2025-12-13 20:59:40,362 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "ESC[36mGET /styles.css HTTP/1.1ESC[0m" 304 -
2025-12-13 20:59:40,576 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "ESC[36mGET / HTTP/1.1ESC[0m" 304 -
2025-12-13 20:59:40,591 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "ESC[36mGET /styles.css HTTP/1.1ESC[0m" 304 -
2025-12-13 20:59:40,593 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "ESC[36mGET /app.js HTTP/1.1ESC[0m" 304 -
2025-12-13 20:59:40,604 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "GET /health HTTP/1.1" 200 -
2025-12-13 20:59:40,725 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "ESC[36mGET /styles.css HTTP/1.1ESC[0m" 304 -
2025-12-13 20:59:40,845 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "ESC[35mESC[1mGET /balance HTTP/1.1ESC[0m" 500 -
2025-12-13 20:59:40,855 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "GET /history HTTP/1.1" 200 -
2025-12-13 20:59:40,958 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "ESC[36mGET / HTTP/1.1ESC[0m" 304 -
2025-12-13 20:59:40,983 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "ESC[36mGET /styles.css HTTP/1.1ESC[0m" 304 -
2025-12-13 20:59:40,984 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "ESC[36mGET /app.js HTTP/1.1ESC[0m" 304 -
2025-12-13 20:59:40,993 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:40] "GET /health HTTP/1.1" 200 -
2025-12-13 20:59:41,236 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:41] "ESC[35mESC[1mGET /balance HTTP/1.1ESC[0m" 500 -
2025-12-13 20:59:41,246 - werkzeug - INFO - 127.0.0.1 - - [13/Dec/2025 20:59:41] "GET /history HTTP/1.1" 200 -
```

Webhook LOGS



```
================================================================================
VERIFYING ORDER: 11548194
================================================================================

✅ Order Found on Binance Testnet!

📋 Order Details:
   Order ID: 11548194
   Symbol: BTCUSDT
   Status: FILLED
   Side: SELL
   Type: MARKET
   Price: 0.00000000 USDT
   Quantity: 0.00100000 BTC
   Original Quantity: 0.00100000 BTC
   Total Value: 90.07223000 USDT
   Created: 2025-12-13 22:03:04
   Updated: 2025-12-13 22:03:04

      ✅ Order Status: FILLED - Successfully executed!
```

Testnet Trade Confirmation

CONCLUSION: PERFORMANCE SUMMARY & IMPROVEMENTS

PERFORMANCE SUMMARY

The bot successfully demonstrates end-to-end integration between TradingView Pine Script and Binance Testnet. Successfully executed 35+ trades (34 successful, 2 failed) with all trades recorded in trade_history.csv. The multi-indicator strategy generates high-quality signals, and the webhook integration works correctly with proper error handling.

FUTURE IMPROVEMENTS:

- Risk Management: Stop-loss orders, position sizing, drawdown limits

- Strategy: Back testing framework, dynamic parameters, multiple trading pairs

- System: Webhook authentication, rate limiting, database storage

- Analytics: Performance metrics (win rate, P/L, Sharpe ratio)

The system is production-ready for testnet trading and provides a solid foundation for algorithmic trading with potential for live trading enhancements.