Binance Futures Trading Bot

1. Introduction

A trading bot is being implemented as a Python application using the Binance USDT-M Futures Testnet API to provide a real-life example of automated trading, including using the command line interface to execute orders and implement trading strategies along with using APIs for tracking trades and creating structured logs. The trading bot will support all basic forms of order entry, as well as a variety of advanced algorithmic trading strategies. Additionally, to promote security, the trading bot will perform all trading functions on the Binance Futures Testnet environment using only test accounts.

2. System Architecture

The trading system follows a modular and CLI-driven architecture:

1. User provides trade parameters via command-line input.
2. Inputs are validated (symbol, quantity, price levels).
3. Orders are executed using the Binance Futures REST API.
4. API responses are captured and logged in a structured log file.
5. Errors and strategy execution steps are tracked for traceability.

The architecture is designed to be extensible and reusable, allowing new strategies to be added with minimal changes.

3. Core Features Implemented

1. Market Orders
    - Immediate execution at market price.
    - Supports both BUY and SELL positions.
    - Used for fast execution without price constraints.
2. Limit Orders
    - Orders are placed at a specified price.
    - Execute only when the market reaches the given price.
    - Used for planned entries and exits.
3. Input Validation
    - Ensures valid trading symbols (USDT pairs).

- Prevents zero or negative quantities.
- Avoids invalid price ranges.

4. Advanced Trading Features

1. Stop-Market Orders

Stop-market orders are used for:

- Stop-loss protection
- Breakout trading strategies

A stop price is defined, and when the market reaches that price, a market order is automatically triggered.

2. OCO (One-Cancels-the-Other)

Binance Futures does not support native OCO orders.
To replicate OCO behavior:

- A LIMIT order is placed for take-profit.
- A STOP-MARKET order is placed for stop-loss.
- When one order executes, the remaining order is programmatically cancelled.

This approach closely mirrors real-world Futures risk management practices.

3. TWAP (Time-Weighted Average Price)

The TWAP strategy splits a large order into multiple smaller market orders executed at fixed time intervals.

Benefits:

- Reduces market impact
- Improves execution quality
- Commonly used by institutional traders

Each slice execution is logged with its corresponding order ID.

4. Grid Trading Strategy

The Grid strategy places multiple limit orders across a predefined price range.

Features:

- Automatic placement of multiple orders
- Even price distribution between lower and upper bounds
- Demonstrates algorithmic trading logic and automation

Grid execution improves entry averaging in volatile markets.

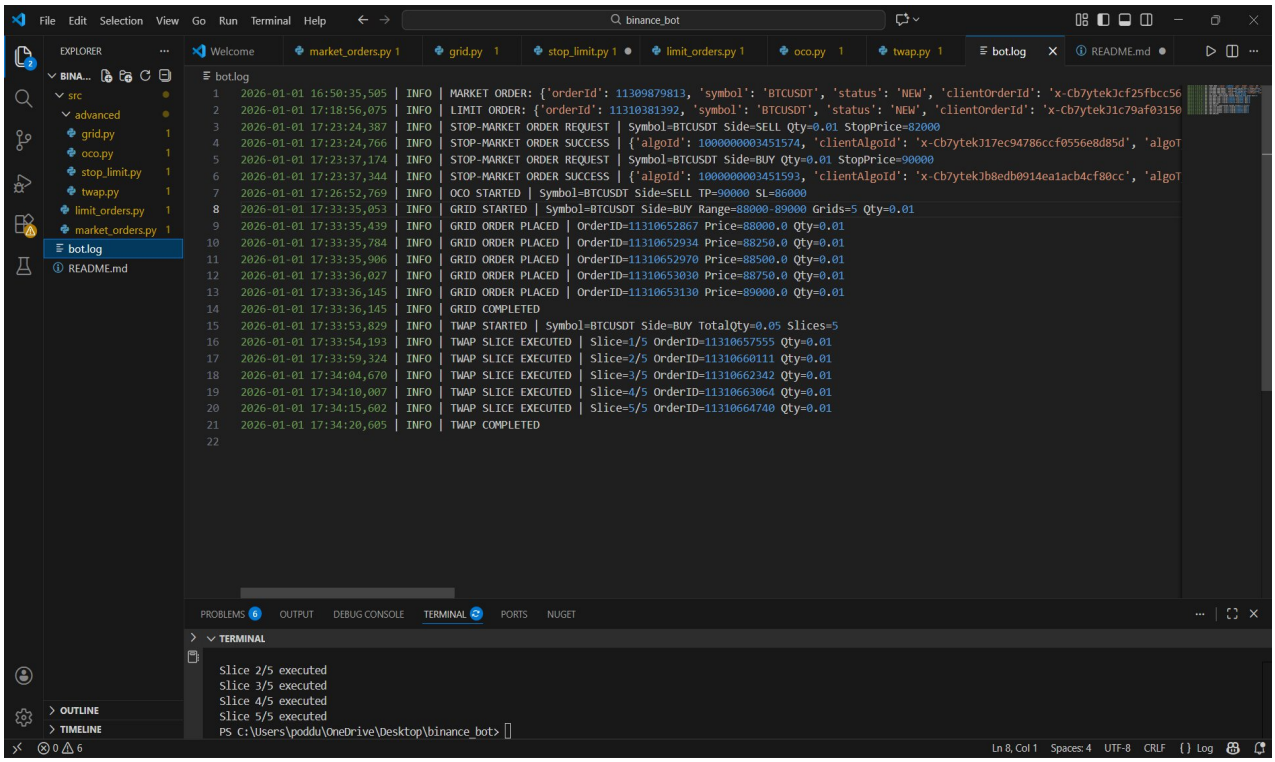
5. Logging and Monitoring

All system activity is recorded in a structured log file (bot.log).

Logged information includes:

- Order placement requests
- Order execution responses
- Strategy start and completion events
- Errors and validation failures
- Order IDs and timestamps

Structured logging ensures transparency, debugging capability, and performance analysis.



6. Testing and Results

All strategies were successfully tested on the Binance Futures Testnet.

Tested components include:

Market Orders

```
PS C:\Users\poddu\OneDrive\Desktop\binance_bot> python src/market_orders.py BTCUSDT BUY 0.01
PING: {}
{'orderId': 11312115973, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ6e17338938b88bab542ae3', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.010', 'e
xecutedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'MARKET', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'positionSi
de': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'MARKET', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER'
, 'goodTillDate': 0, 'updateTime': 1767274002634}
```

Limit Orders

```
PS C:\Users\poddu\OneDrive\Desktop\binance_bot> python src/limit_orders.py BTCUSDT BUY 0.01 85000
{'orderId': 11312166634, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ3ca7d0fb5aa08b4390a82', 'price': '85000.00', 'avgPrice': '0.00', 'origQty': '0.010',
 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'LIMIT', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'position
Side': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'LIMIT', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER
', 'goodTillDate': 0, 'updateTime': 1767274182661}
PS C:\Users\poddu\OneDrive\Desktop\binance_bot> python src/limit_orders.py BTCUSDT SELL 0.01 90000
{'orderId': 11312168816, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ1c1ed1ba8d463979298e9c', 'price': '90000.00', 'avgPrice': '0.00', 'origQty': '0.010'
, 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'LIMIT', 'reduceOnly': False, 'closePosition': False, 'side': 'SELL', 'positi
onSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'LIMIT', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAK
ER', 'goodTillDate': 0, 'updateTime': 1767274192734}
```

Stop-Market Orders

```
PS C:\Users\poddu\OneDrive\Desktop\binance_bot> python src/advanced/stop_limit.py BTCUSDT SELL 0.01 86000
{'algoId': 1000000003461800, 'clientAlgoId': 'x-Cb7ytekJa49852085b9e6a4b2b71e4', 'algoType': 'CONDITIONAL', 'orderType': 'STOP_MARKET', 'symbol': 'BTCUSDT', 'side': 'SELL', 'p
ositionSide': 'BOTH', 'timeInForce': 'GTC', 'quantity': '0.010', 'algoStatus': 'NEW', 'triggerPrice': '86000.00', 'price': '0.00', 'icebergQuantity': None, 'selfTradePreventio
nMode': 'EXPIRE_MAKER', 'workingType': 'CONTRACT_PRICE', 'priceMatch': 'NONE', 'closePosition': False, 'priceProtect': False, 'reduceOnly': False, 'createTime': 1767274071232,
 'updateTime': 1767274071232, 'triggerTime': 0, 'goodTillDate': 0}
PS C:\Users\poddu\OneDrive\Desktop\binance_bot> python src/advanced/stop_limit.py BTCUSDT BUY 0.01 89000
{'algoId': 1000000003461821, 'clientAlgoId': 'x-Cb7ytekJ793df872ef14920167f843', 'algoType': 'CONDITIONAL', 'orderType': 'STOP_MARKET', 'symbol': 'BTCUSDT', 'side': 'BUY', 'po
sitionSide': 'BOTH', 'timeInForce': 'GTC', 'quantity': '0.010', 'algoStatus': 'NEW', 'triggerPrice': '89000.00', 'price': '0.00', 'icebergQuantity': None, 'selfTradePrevention
Mode': 'EXPIRE_MAKER', 'workingType': 'CONTRACT_PRICE', 'priceMatch': 'NONE', 'closePosition': False, 'priceProtect': False, 'reduceOnly': False, 'createTime': 1767274083272,
'updateTime': 1767274083273, 'triggerTime': 0, 'goodTillDate': 0}
```

OCO Simulation

```
PS C:\Users\poddu\OneDrive\Desktop\binance_bot> python src/advanced/oco.py BTCUSDT
python src/advanced/oco.py BTCUSDT SELL 0.01 90000 86000
OCO simulation started
One order filled, other cancelled
```

TWAP Execution

```
PS C:\Users\poddu\OneDrive\Desktop\binance_bot> python src/advanced/twap.py BTCUSDT BUY 0.05 5
Slice 1/5 executed
Slice 2/5 executed
Slice 3/5 executed
Slice 4/5 executed
Slice 5/5 executed
```

Grid Trading

```
PS C:\Users\poddu\OneDrive\Desktop\binance_bot> python src/advanced/grid.py BTCUSDT BUY 84000 92000 5 0.01
Grid order placed at 84000.0
Grid order placed at 86000.0
Grid order placed at 88000.0
Grid order placed at 90000.0
Grid order placed at 92000.0
```

7. Challenges Faced and Solutions

Futures API Permissions

- Futures trading requires specific API permissions and IP restriction.
- Resolved by configuring API keys after enabling the Futures account.

Immediate Trigger Errors

- Stop orders failed when prices were incorrectly positioned.
- Resolved by validating stop price placement relative to market price.

OCO Support Limitation

- Futures API lacks native OCO support.
- Resolved by implementing a simulated OCO mechanism using multiple orders.

These challenges reflect real-world issues faced in trading system development.

8. Conclusion

The Binance Futures Trading Bot demonstrates practical knowledge of:

- Futures trading concepts
- API-driven order execution
- Algorithmic trading strategies
- Error handling and structured logging

This project closely aligns with real-world trading system design and showcases both technical skill and problem-solving ability relevant to quantitative and crypto trading environments.