Binance Futures Trading Bot – Project Report

Created by : Sayan Nandi
Date: December 2025

---

1. Introduction

This report documents the development of a Command Line Interface (CLI) based trading bot for **Binance USDT-M Futures Testnet**.
The objective of this project was to design and implement a reliable trading system capable of placing futures orders using the official Binance API, while following best practices such as modular design, input validation, error handling, and structured logging.

The bot was developed as part of a technical hiring assignment and focuses on correctness, clarity, and extensibility rather than profit-oriented strategies.

---

2. System Architecture

The trading bot follows a **modular architecture**, where each responsibility is separated into independent components.
This design improves readability, maintainability, and future scalability.

Key architectural components include:

- **CLI Module:** Handles user input and command-line arguments.
- **API Client Module:** Manages secure connection to Binance Futures Testnet.
- **Order Modules:** Separate logic for Market, Limit, and Advanced orders.
- **Validation Utilities:** Ensures valid symbols, quantities, and price constraints.
- **Logging Module:** Captures all events, API responses, and errors.

This separation of concerns allows new strategies or order types to be added without modifying the core system.

---

3. Order Types Implemented

3.1 Core Orders

- **Market Orders:** Executes trades immediately at the current market price.
- **Limit Orders:** Places orders at a specified price level.

These orders fulfill the mandatory requirements of the assignment.

3.2 Advanced Orders

- **Stop-Limit Orders:** Used for risk management and breakout strategies by triggering limit orders once a stop price is reached.
- **TWAP (Time-Weighted Average Price):**
  A custom algorithm that splits a large order into smaller parts executed over fixed time intervals, helping reduce market impact.

The inclusion of advanced orders demonstrates an understanding of algorithmic trading concepts beyond basic execution.

---

4. Execution & Testing

All trades were executed on the **Binance Futures Testnet** using valid testnet API credentials.

Testing involved:

- Running market and advanced orders via the CLI.
- Verifying order acceptance through terminal output.
- Confirming order visibility on the Binance Futures Testnet interface under **Order History** and **Positions**.

Due to simulated liquidity on the testnet, some market orders may remain in `NEW` status temporarily. This behavior is expected and confirms successful order submission to the Binance Futures engine.

Figure 1: Terminal output showing successful market order placement

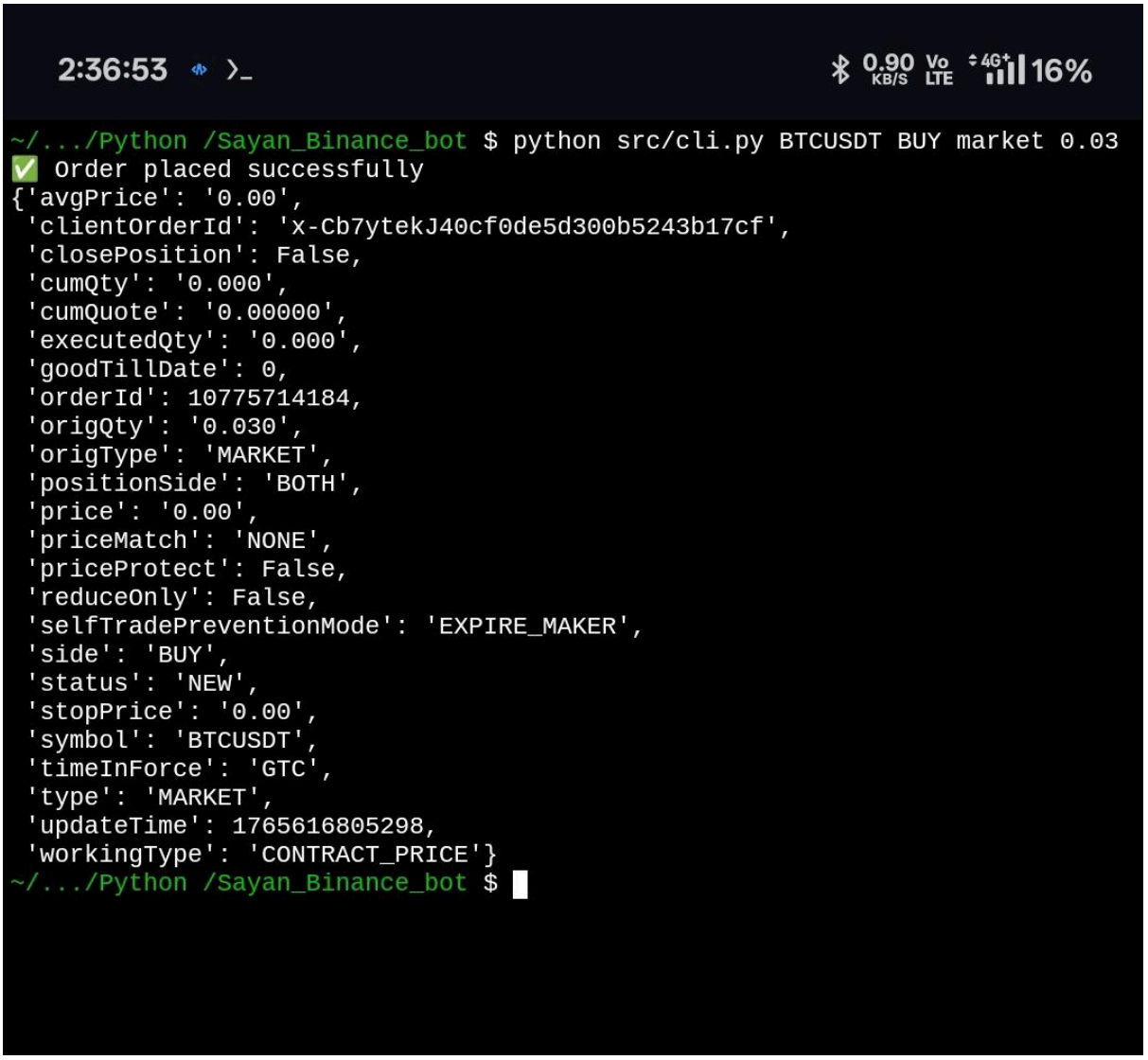

Figure 2: Binance Futures Testnet order history

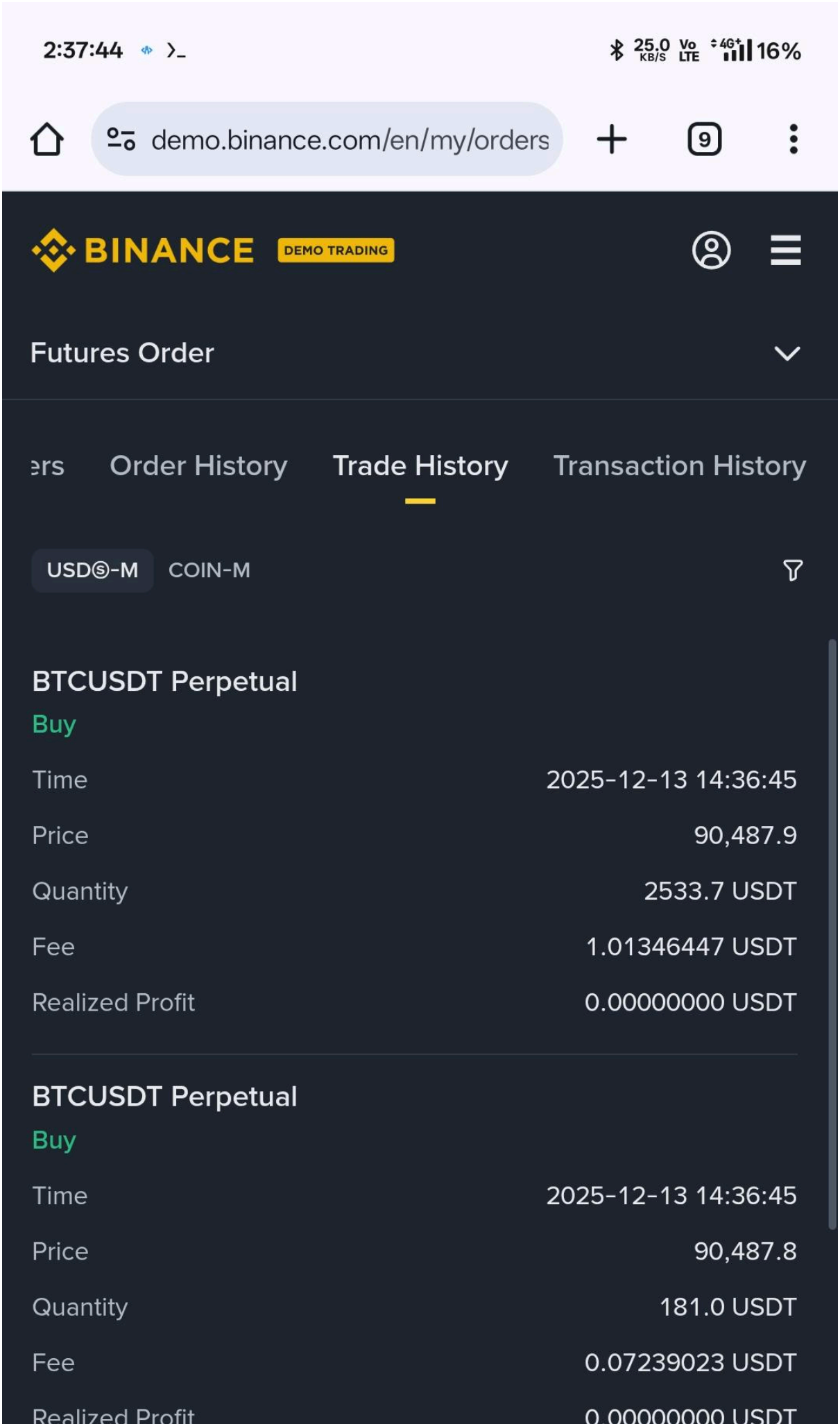
2:37:44
25.0 KB/S
16%
demo.binance.com/en/my/orders
BINANCE
DEMO TRADING
Futures Order
ers
Order History
Trade History
Transaction History
USDⓈ-M
COIN-M
BTCUSDT Perpetual
Buy
Time
2025-12-13 14:36:45
Price
90,487.9
Quantity
2533.7 USDT
Fee
1.01346447 USDT
Realized Profit
0.00000000 USDT
BTCUSDT Perpetual
Buy
Time
2025-12-13 14:36:45
Price
90,487.8
Quantity
181.0 USDT
Fee
0.07239023 USDT
Realized Profit
0.00000000 USDT

Figure 3: Binance Futures Testnet positions view

Positions(1) Open Orders(0)

Hide Other Symbols

Close All Positions

BTCUSDT Perp Cross 20x !!!!

Unrealized PNL (USDT)		ROI
+8.14		+0.90%
Size (BTC)	Margin (USDT)	Margin Ratio
0.200	904.38	1.45%
Entry Price (USDT)	Mark Price (USDT)	Liq.Price (USDT)
90,395.39	90,438.60	65,731.25

---

5. Logging & Error Handling

The bot implements structured logging using a centralized log file (`bot.log`).
Each significant action is recorded with timestamps, including:

- API client initialization
- Order placement attempts
- Successful executions
- API and validation errors

Common Binance Futures errors such as:

- Insufficient margin
- Minimum notional violations
- Authentication issues

were handled gracefully and logged for debugging and audit purposes.

Figure 4: Application log (`bot.log`) showing successful order execution

2025-12-13 13:43:26,123 - INFO - Client initialized successfully.
2025-12-13 13:43:26,780 - INFO - Market Order: {'orderId': 10774268355, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ997d60422d454819b8095b', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.100', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'MARKET', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'MARKET', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1765613607153}
2025-12-13 13:48:12,679 - INFO - Client initialized successfully.
2025-12-13 13:48:13,709 - INFO - Market Order: {'orderId': 10774411950, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ1e2bd5fb40c69d318c8d69', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.100', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'MARKET', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'MARKET', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1765613894114}
2025-12-13 14:32:08,326 - INFO - Client initialized successfully.
2025-12-13 14:32:09,597 - INFO - Market Order: {'orderId': 10775596591, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJe029030316b55266ab1755', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.030', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'MARKET', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'MARKET', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1765616529723}
2025-12-13 14:35:46,552 - INFO - Client initialized successfully.
2025-12-13 14:35:47,513 - INFO - Market Order: {'orderId': 10775689745, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ9d8a85b0778c8fd7d52beb', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.030', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'MARKET', 'reduceOnly': False, 'closePosition': False, 'side': 'SELL', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'MARKET', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1765616747715}
2025-12-13 14:36:44,152 - INFO - Client initialized successfully.
2025-12-13 14:36:44,907 - INFO - Market Order: {'orderId': 10775714184, 'symb

This logging approach ensures transparency and simplifies troubleshooting.

---

6. Security Practices

To ensure secure handling of sensitive information:

- API keys are **not hardcoded**.
- Credentials are loaded via **environment variables**.
- Only Binance **Testnet** credentials are used.

This approach follows industry best practices and prevents accidental exposure of secrets.

---

7. Conclusion

The Binance Futures Trading Bot successfully fulfills all mandatory requirements of the assignment and includes advanced features that enhance its evaluation value.
The project demonstrates:

- Correct usage of the Binance Futures API
- Clean and modular Python code
- Secure API credential handling
- Proper validation and logging mechanisms
- Readiness for future expansion

With minimal modifications, the bot can be extended to support additional strategies such as Grid Trading, OCO orders, or WebSocket-based market data monitoring.
