Binance Futures Testnet Trading Bot

Name: Amish Nair

Date: 08/07/2025

Project Overview

This project involves building a simplified trading bot for the Binance Futures Testnet using Python. The bot supports placing market, limit, and advanced order types such as Stop-Limit, OCO, TWAP, and Grid strategies. The bot also includes CLI interaction, logging, and error handling. At last, I implemented it making a lightweight frontend using Streamlit.

Technologies Used

- Python 3.10

- python-binance library

- dotenv for key management

- Logging module

- Binance Futures Testnet API

-Streamlit for implementation

Implementation Details

client_setup.py

- Loads API keys securely from .env

market_orders.py

- Places buy/sell market orders
- Example used: BTCUSDT, 0.002 qty

limit_orders.py

- Places limit orders with adjustable price and quantity

cli.py

- Accepts CLI input for symbol, side, quantity, etc.
- Makes bot interactive

advanced/stop_limit.py

- Uses stop + limit to create conditional orders

advanced/oco.py

- Simulates an OCO (One-Cancels-the-Other) order
- Places TP + SL simultaneously

advanced/twap.py

- Breaks up large orders into smaller chunks over time

advanced/grid.py

- Places multiple limit orders across price range

- Used error checking for Binance's minimum notional limit

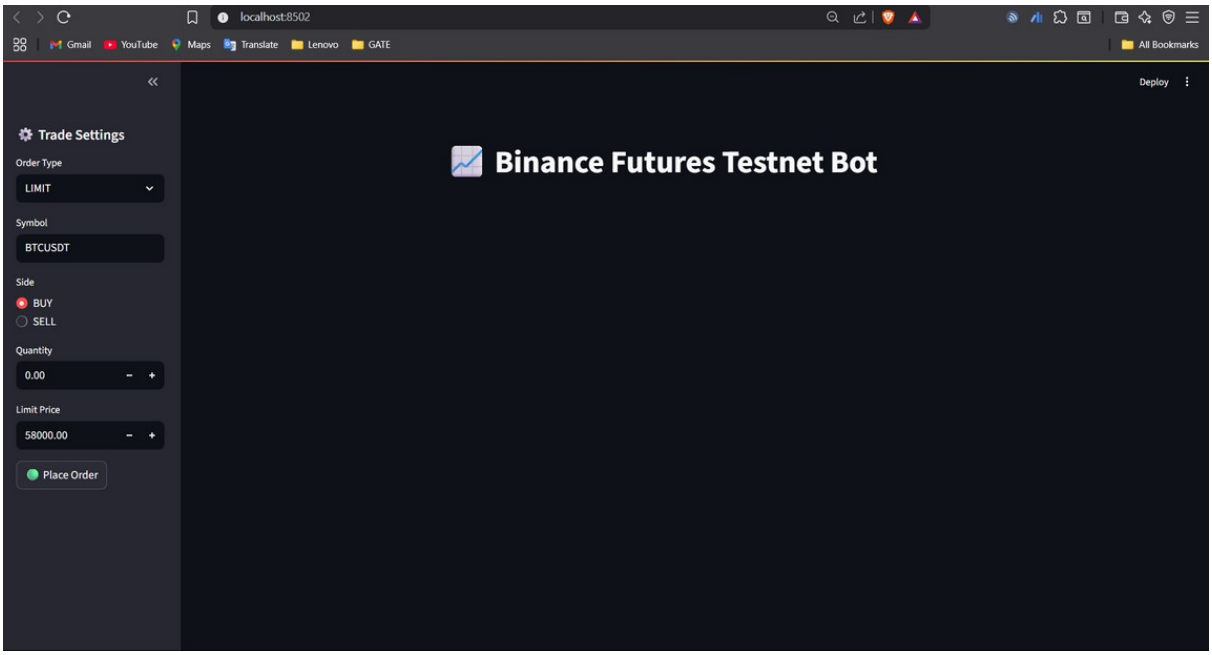
streamlit_app.py

- Creates a simple and lightweight Frontend UI to execute the orders

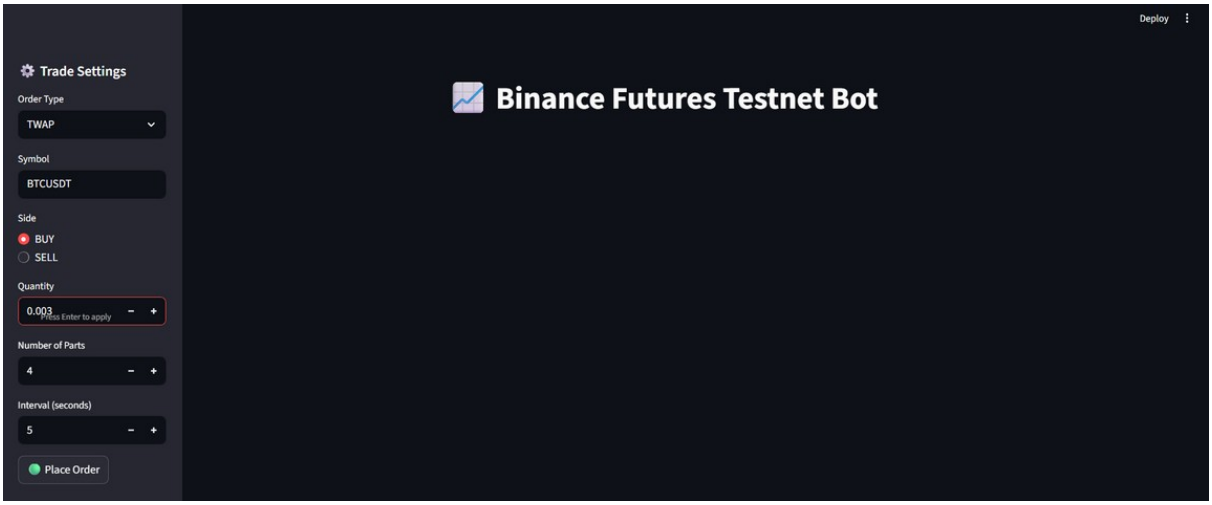
Screenshots

Using Stremlit for lightweight Frontend

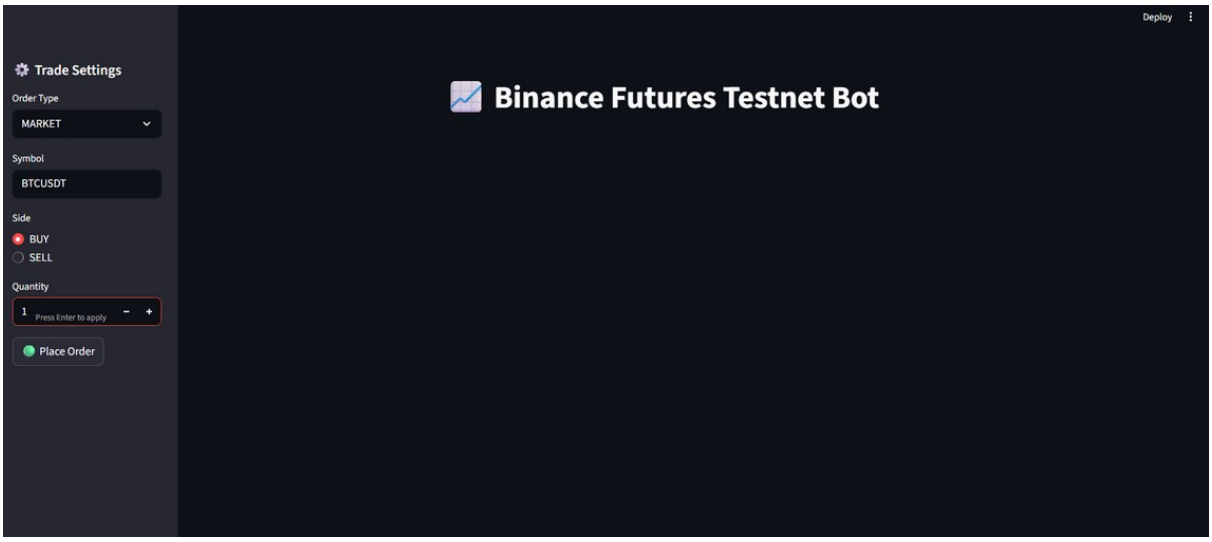
1- Limit Order UI



2- TWAP UI



3- Market Order UI



4- Order Execution using TWAP

```python
from advanced.twap import place_twap_order

orders = place_twap_order(
    symbol="BTCUSDT",
    side="BUY",
    total_quantity=0.01,
    chunks=5,
    interval_sec=3  # 3 seconds between each order
)

print(orders)
```

```
🔄 Executing TWAP: 5 x 0.002 BUY orders every 3 sec

✅ Placed order 1/5 | Order ID: 5222275221
✅ Placed order 2/5 | Order ID: 5222275387
✅ Placed order 3/5 | Order ID: 5222275549
✅ Placed order 4/5 | Order ID: 5222275680
✅ Placed order 5/5 | Order ID: 5222275780
[{'orderId': 5222275221, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJa9d6d36ffe96ac4d5ca744', 'price': '0.00', 'avgPrice': '0.00',
'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'MARKET', 'reduceOnly': False, 'close
Position': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'MARKE
T', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1751965388181}, {'orderId': 5222275387, 'symbol':
'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ435140ae27c292233351a', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.002', 'executedQty':
'0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'MARKET', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'p
ositionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'MARKET', 'priceMatch': 'NONE', 'selfTrade
PreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1751965391653}, {'orderId': 5222275549, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOr
derId': 'x-Cb7ytekJc4fc74d0c31a59ac76f072', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuot
e': '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', 'g
oodTillDate': 0, 'updateTime': 1751965395065}, {'orderId': 5222275680, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ5ec1e7845f08c3b1
53d78d', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.002', '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_PRIC
E', 'priceProtect': False, 'origType': 'MARKET', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 175196
5398471}, {'orderId': 5222275780, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJbaa6ef8d70bf99eb2dec39', 'price': '0.00', 'avgPrice':
'0.00', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'MARKET', 'reduceOnly': Fals
e, 'closePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origTyp
e': 'MARKET', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1751965401901}]
```

5- Execution using Grid Orders

```python
from advanced.grid import place_grid_orders

orders = place_grid_orders(
    symbol="BTCUSDT",
    side="BUY",              # Place buy limit orders at grid levels
    quantity=0.002,
    lower_price=57500,
    upper_price=58000,
    grid_levels=5
)

print(orders)
```

```
📈 Placing 5 BUY grid orders from 57500 to 58000
✅ Order 1/5 @ 57500.0
✅ Order 2/5 @ 57625.0
✅ Order 3/5 @ 57750.0
✅ Order 4/5 @ 57875.0
✅ Order 5/5 @ 58000.0
[{'orderId': 5222286951, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ289eed504b6feffa36fedd', 'price': '57500.00', 'avgPrice': '0.0
0', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'LIMIT', 'reduceOnly': False, 'cl
osePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'LIM
IT', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1751965567449}, {'orderId': 5222286966, 'symbol':
'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ9faa577cd55e95a4eeef96', 'price': '57625.00', 'avgPrice': '0.00', 'origQty': '0.002', 'executedQt
y': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'LIMIT', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY',
'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'LIMIT', 'priceMatch': 'NONE', 'selfTrad
ePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1751965567774}, {'orderId': 5222286997, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientO
rderId': 'x-Cb7ytekJ1724238d7ceaca6c668fbb', 'price': '57750.00', 'avgPrice': '0.00', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cum
Quote': '0.00000', 'timeInForce': 'GTC', 'type': 'LIMIT', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPric
e': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'origType': 'LIMIT', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER',
'goodTillDate': 0, 'updateTime': 1751965568089}, {'orderId': 5222287021, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJcf4a23a0f8f200
f3995827', 'price': '57875.00', 'avgPrice': '0.00', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce':
'GTC', 'type': 'LIMIT', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT
_PRICE', 'priceProtect': False, 'origType': 'LIMIT', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 17
51965568418}, {'orderId': 5222287043, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ2e9d3021267a6caddd632', 'price': '58000.00', 'avg
Price': '0.00', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'type': 'LIMIT', 'reduceOnl
y': False, 'closePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'priceProtect': False, 'o
rigType': 'LIMIT', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1751965568735}]
```

6- Order Status after Placing it

```python
# After placing the market order
order = place_market_order("BTCUSDT", "BUY", 0.002)
order_id = order['orderId']

# Now check its status
status = client.futures_get_order(symbol="BTCUSDT", orderId=order_id)
print("📌 Status:", status['status'])
print("💰 Executed Quantity:", status['executedQty'])
```

```
📌 Status: FILLED
💰 Executed Quantity: 0.002
```

ISSUES FACED DURING IMPLEMENTATION (With Solutions):

- **Notional error (-4164):** Fixed by increasing quantity and adding logic to skip small notional orders.
- **Missing ORDER_TYPE_STOP:** Resolved by using "STOP" as a string instead of constant.

- **Environment setup confusion:** Used .env with dotenv to manage keys securely.

—

Final Notes

This project helped me understand how to interact with live trading APIs securely, structure trading logic modularly, and build reusable components for both basic and advanced order types. I also gained hands-on experience debugging API-level errors and designing a system that could be expanded into a production-level bot. This project helped me understand how to interact with live trading APIs securely, structure trading logic modularly, and build reusable components for both basic and advanced order types. I also gained hands-on experience debugging API-level errors and designing a system that could be expanded into a production-level bot.