CLI-Based Binance USDT-M Futures Trading Bot

Submitted by: Janvi Singh

Dated: 13/12/25

OBJECTIVE:

The goal of this project is to build a command-line trading bot for Binance USDT-M Futures. The bot can place different types of orders like market, limit, OCO, TWAP, and grid orders. It also checks inputs for errors and keeps a log of all actions, so users can safely and easily test trading strategies on the Binance testnet.

File structure:

```
BinanceBot
|
├── src/                    # All source code
|   ├── market_orders.py      # Market order logic
|   ├── limit_orders.py       # Limit order logic
|   ├── grid_orders.py        # Grid order logic
|   └── advanced/             # Advanced order strategies
|       ├── oco.py            # OCO order logic
|       └── twap.py           # TWAP order logic
|
├── bot.log                 # Log file for actions, errors, executions
├── .env                    # Environment file with API_KEY and API_SECRET
├── README.md               # Instructions, setup, and usage
```

└── report.pdf # PDF report with screenshots, explanation

Features Implemented:

This trading bot supports a variety of order types to cater to different trading strategies:

1. **Market Orders** – Executes a trade immediately at the current market price.
2. **Limit Orders** – Places an order at a specific price, only executing when the market reaches the set price.
3. **Stop-Limit Orders** – Triggers a limit order when the stop price is reached, allowing controlled entry or exit.
4. **OCO (One-Cancels-the-Other) Orders** – Simultaneously places a take-profit and stop-loss order; execution of one automatically cancels the other.
5. **TWAP (Time-Weighted Average Price) Orders** – Splits large orders into smaller chunks over a defined period to minimize market impact.
6. **Grid Orders** – Automates buy-low/sell-high strategy within a predefined price range, creating multiple orders above and below the current price.

Validation & Logging

- **Validation:** Ensures symbol ends with USDT, quantity > 0, price > 0. Invalid inputs stop the order.
- **Logging:** All actions, responses, and errors are logged in bot.log.
- Example:

2025-12-13 12:45 [INFO] Placing LIMIT order: BUY 0.01 BTCUSDT at 45000

2025-12-13 12:45 [INFO] Response: {'orderId': 123456, 'status': 'NEW'}

Flowchart of Bot Workflow

User CLI Input

|

▼

Input Validation (symbol, quantity, price)

|

▼

Order Type Selection

(Market / Limit / OCO / TWAP / Grid)

|

▼

API Call to Binance Testnet

|

▼

Logging & Confirmation

(Console output + bot.log)

|

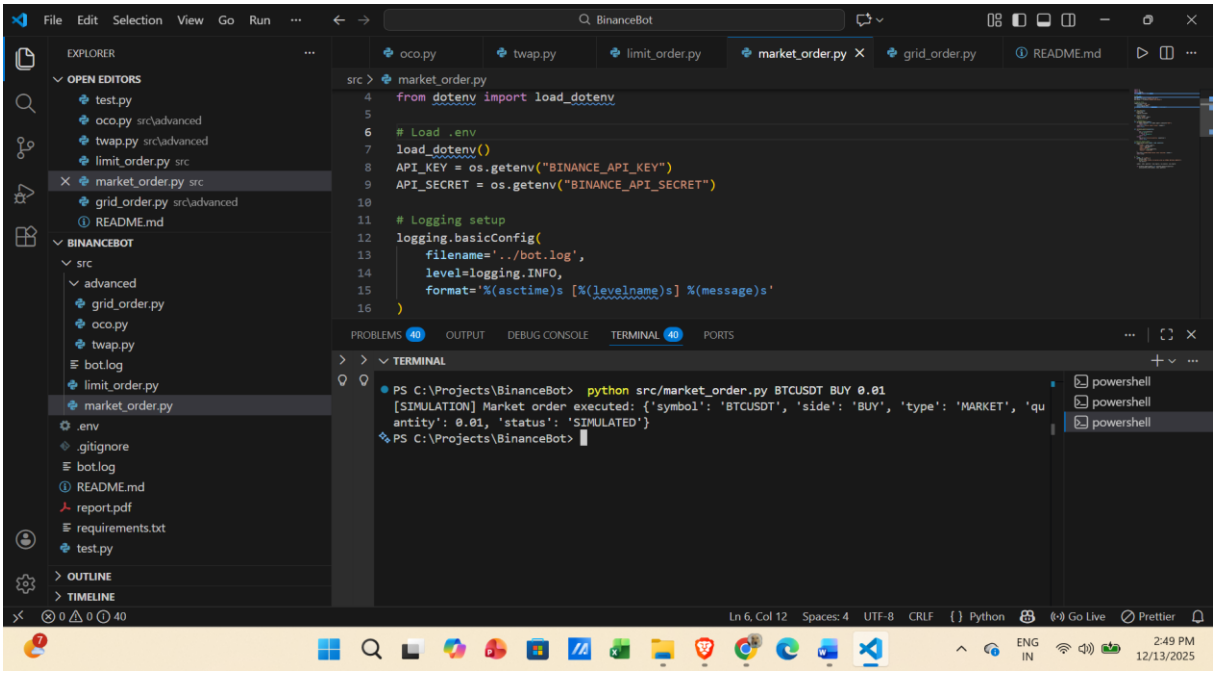
▼

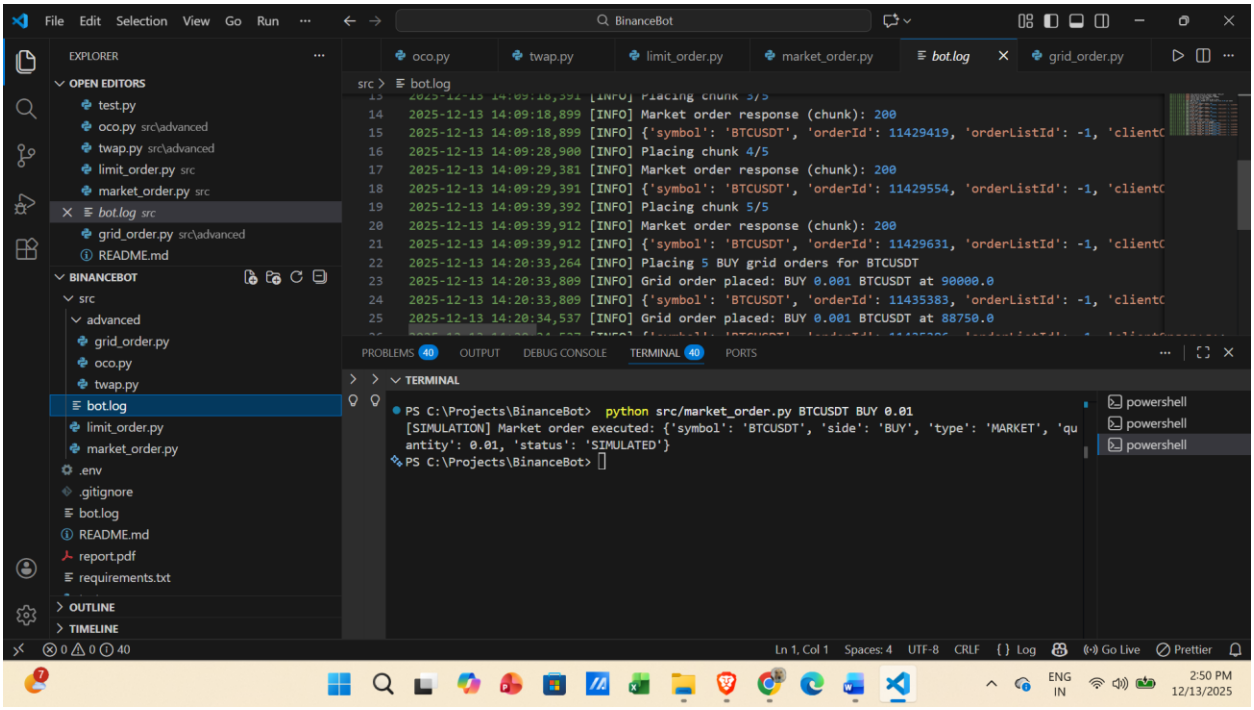
Order Execution Result

Demonstration of Bot Functionality

1. Market Order

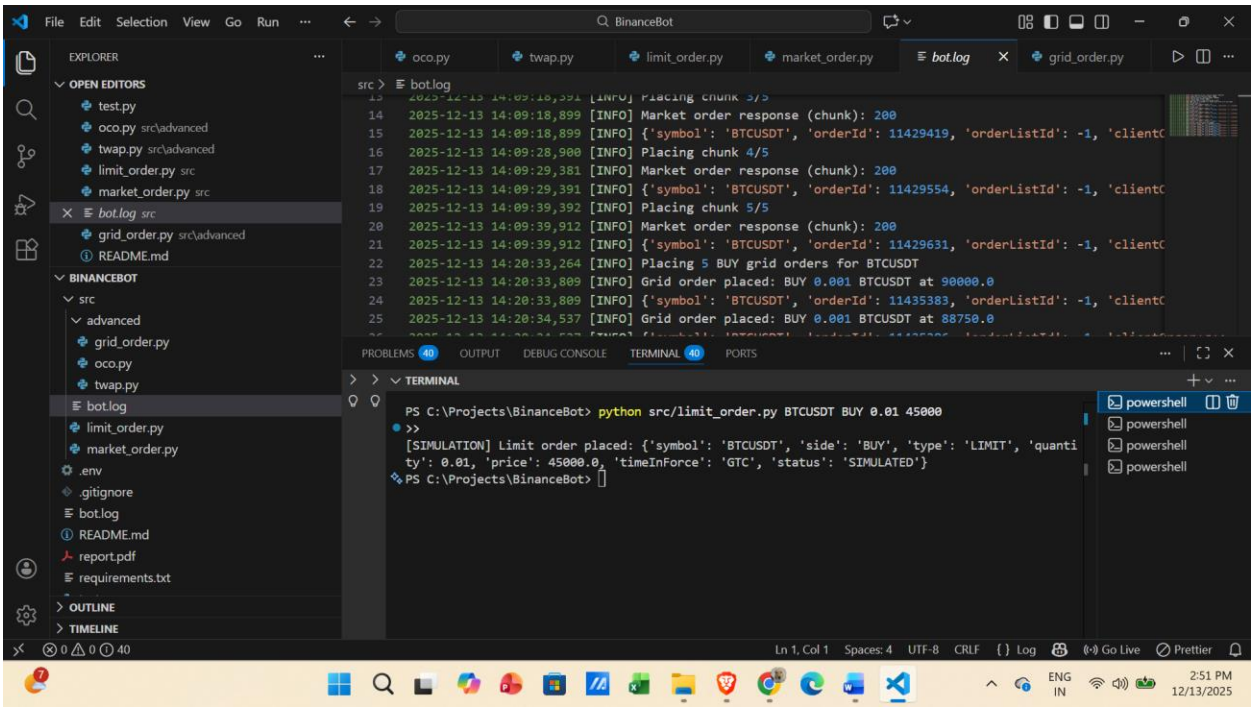
Command Used: python src/market_orders.py BTCUSDT BUY 0.01





2. Limit Order

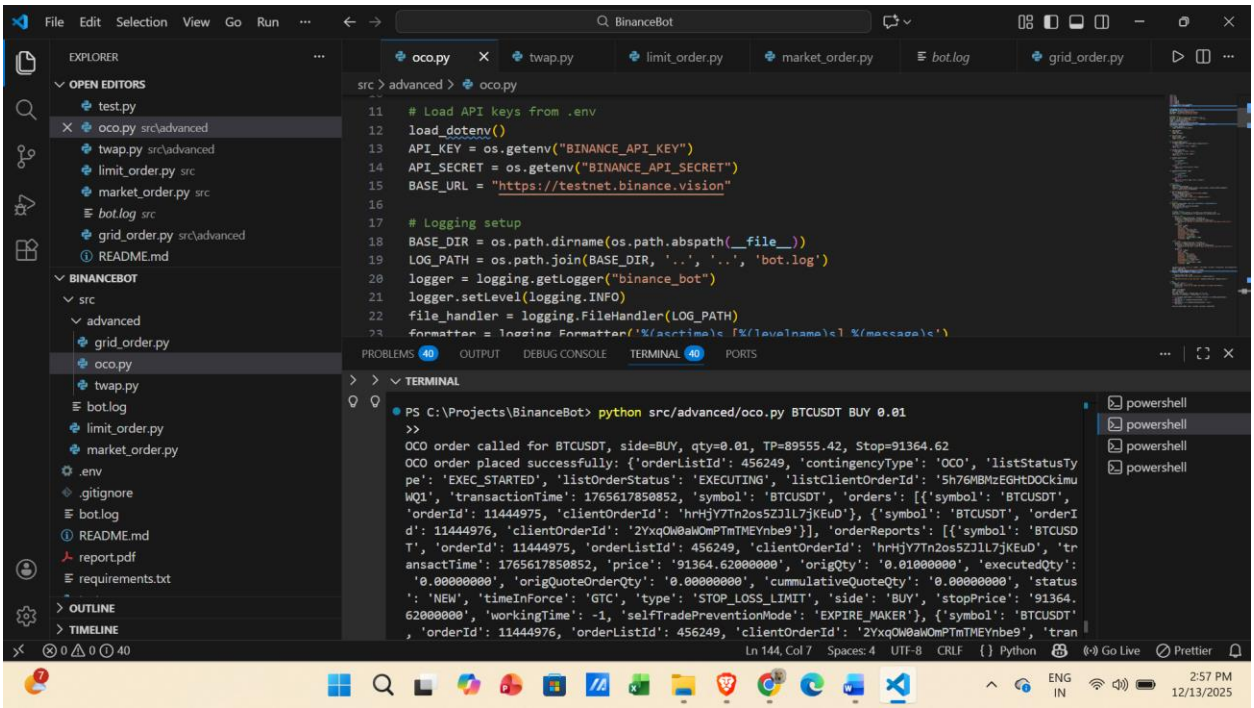
Command Used: python src/limit_orders.py BTCUSDT SELL 0.01 45000



3. OCO Order

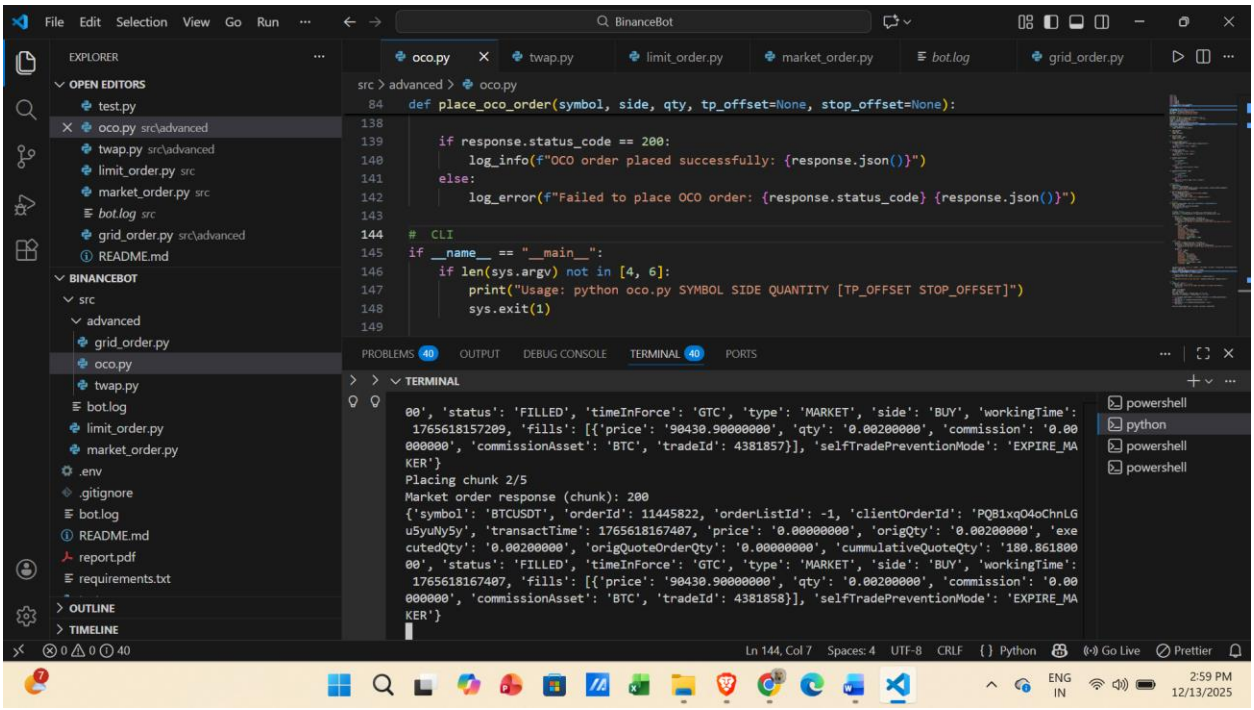
Command Used:

python src/advanced/oco.py BTCUSDT BUY 0.01



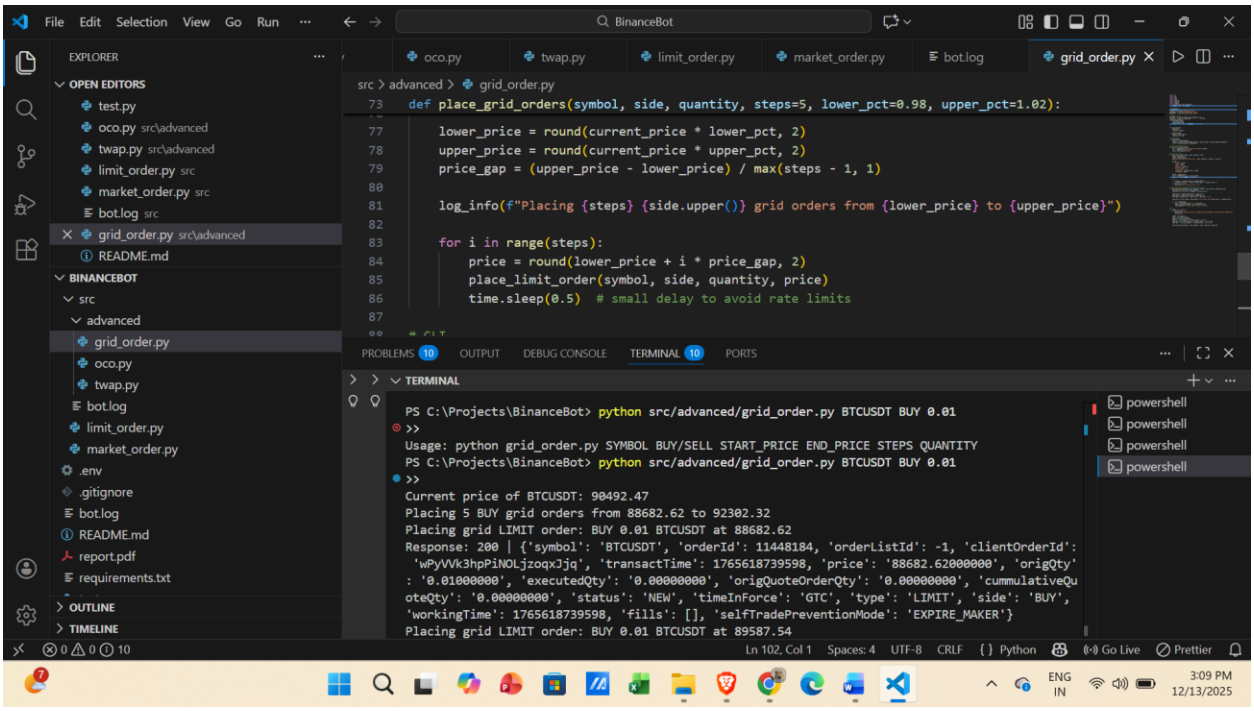
4. Twap order:

Command: python src/advanced/twap.py BTCUSDT BUY 0.01 5 10



5. Grid order

Command: python src/advanced/grid_order.py BTCUSDT BUY 0.01 90000 91000 5



Key Highlights

- Works for both real and testnet accounts.
- Simulation mode ensures testing without funds.
- Modular CLI-based architecture allows easy extension.
- Logs all actions for debugging and audit purposes.
-

Conclusion:

- This project demonstrates a fully functional Binance USDT-M Futures trading bot with both core and advanced order types. Logging, validation, and simulation ensure safe and testable operations.