# Binance USDT-M Futures Order Bot

**Author:** Supriya S
**Date:** 09-08-2025

## Overview

This project implements a command-line trading bot for Binance USDT-M Futures, designed to place various order types including essential ones like market and limit orders, along with advanced order strategies such as stop-limit, OCO (One-Cancels-the-Other), TWAP (Time-Weighted Average Price), and a simulated grid trading strategy.

The bot supports:

- **Input validation** for all order parameters (symbol, quantity, price, side).
- **Dry-run simulation mode** to test orders without actual API calls, ensuring safety during development and testing.
- **Structured JSON logging** of all actions, requests, responses, and errors, both to console and a persistent log file (bot.log).

This combination makes the bot suitable for further expansion and real trading, while offering strong traceability and robustness.

## Architecture and Code Structure

The project is modularized to separate concerns clearly:

- **src/cli.py**
  Handles CLI input parsing and routes commands to appropriate order handling functions.
- **src/logger.py**
  Implements a logging utility that writes structured JSON logs to both the console and the bot.log file. This ensures uniform and searchable log entries for audit and debugging.
- **src/binance_client.py**
  Wraps the Binance Futures Testnet API with transparent support for dry-run mode. This means if dry-run is enabled, no real API calls are made; instead, simulated responses are returned.
- **Order handling modules:**
  - src/market_orders.py — Market order placement and validation
  - src/limit_orders.py — Limit order placement and validation

    - src/advanced/stop_limit.py — Stop-limit order logic
    - src/advanced/oco.py — Simulates OCO orders (since Binance Testnet may not support OCO futures directly)
    - src/advanced/twap.py — Implements TWAP order slicing over time
    - src/advanced/grid.py — Simulates grid trading by placing staggered buy and sell orders around a price grid
- **Logfile:bot.log**
  Contains time-stamped JSON records of every order attempt, API response, errors, and validation results, enabling detailed post-run analysis.

## Testing and Validation

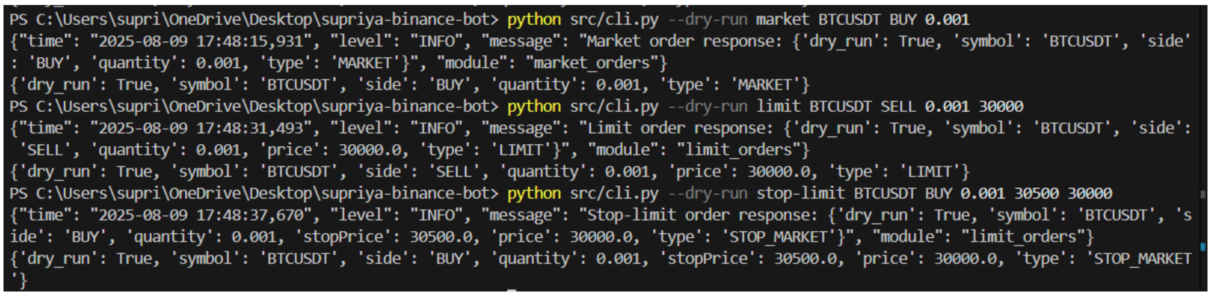
Testing was conducted in a **Windows PowerShell environment** within a **Python virtual environment** to maintain isolation and control over dependencies.

- All tests used the **dry-run mode**, ensuring no real orders were placed on Binance Testnet, minimizing risk.
- Commands were exercised with various inputs for **market**, **limit**, and **stop-limit** orders to validate input handling, parameter parsing, and API response simulation.
- Logs were verified to confirm that every action, including invalid inputs and errors, was logged with detailed JSON messages.
- Advanced order types such as OCO, TWAP, and grid strategies were simulated and logs validated to ensure correct internal logic.

## Analysis and Observations

- **Modular design** greatly simplifies maintenance and extension. Each order type is isolated in its own module, encouraging single responsibility and easy testing.
- **Dry-run mode** is a crucial safety feature, especially when developing strategies that interact with real money and real markets.
- **Comprehensive logging** with JSON structured output allows integration with log parsers, visualization tools, and audit systems.
- The bot currently covers a wide range of order types, making it flexible for multiple trading strategies.
- Optional advanced order types like TWAP and grid trading demonstrate the bot's ability to handle more complex logic beyond simple orders.

- The CLI interface provides a simple and efficient user experience for testing and manual order placement.



```
PS C:\Users\supri\OneDrive\Desktop\supriya-binance-bot> python src/cli.py --dry-run market BTCUSDT BUY 0.001
{"time": "2025-08-09 17:48:15,931", "level": "INFO", "message": "Market order response: {'dry_run': True, 'symbol': 'BTCUSDT', 'side'
: 'BUY', 'quantity': 0.001, 'type': 'MARKET'}", "module": "market_orders"}
{'dry_run': True, 'symbol': 'BTCUSDT', 'side': 'BUY', 'quantity': 0.001, 'type': 'MARKET'}
PS C:\Users\supri\OneDrive\Desktop\supriya-binance-bot> python src/cli.py --dry-run limit BTCUSDT SELL 0.001 30000
{"time": "2025-08-09 17:48:31,493", "level": "INFO", "message": "Limit order response: {'dry_run': True, 'symbol': 'BTCUSDT', 'side':
 'SELL', 'quantity': 0.001, 'price': 30000.0, 'type': 'LIMIT'}", "module": "limit_orders"}
{'dry_run': True, 'symbol': 'BTCUSDT', 'side': 'SELL', 'quantity': 0.001, 'price': 30000.0, 'type': 'LIMIT'}
PS C:\Users\supri\OneDrive\Desktop\supriya-binance-bot> python src/cli.py --dry-run stop-limit BTCUSDT BUY 0.001 30500 30000
{"time": "2025-08-09 17:48:37,670", "level": "INFO", "message": "Stop-limit order response: {'dry_run': True, 'symbol': 'BTCUSDT', 's
ide': 'BUY', 'quantity': 0.001, 'stopPrice': 30500.0, 'price': 30000.0, 'type': 'STOP_MARKET'}", "module": "limit_orders"}
{'dry_run': True, 'symbol': 'BTCUSDT', 'side': 'BUY', 'quantity': 0.001, 'stopPrice': 30500.0, 'price': 30000.0, 'type': 'STOP_MARKET
'}
```

Dry-run execution logs for Market, Limit, and Stop-Limit orders on Binance USDT-M Futures trading bot CLI showing successful order simulation responses with detailed JSON output.

- Future improvements could include:
  - Adding support for **real API key configuration** with environment variables or encrypted storage.
  - Building a **web or GUI frontend** for easier interaction.
  - Integrating **WebSocket** streams for live price and order book updates to enable reactive strategies.
  - Implementing **error recovery and retries** for robustness under API rate limits or network issues.

## Conclusion

This Binance USDT-M Futures Order Bot successfully meets the initial project requirements by providing:

- Market and limit order support with full input validation.
- Advanced order strategies including stop-limit, OCO, TWAP, and grid simulation.
- Dry-run mode for safe testing.
- Structured, detailed JSON logging for transparency and traceability.
- A clean, modular codebase ready for future extensions.

The bot serves as a strong foundation for a more advanced crypto futures trading system, suitable for both testing and real deployments after appropriate key management and live environment configuration.