Project Report :

BINANCE FUTURES ORDER BOT

Done by

Fathima AM

24-10-2025

CONTENT

1. Project Overview

Binance Futures Order Bot is a Python-based command-line tool that helps automate cryptocurrency trading on Binance. With this bot, users can quickly place market orders (buy or sell at the current price) and limit orders (buy or sell at a specific price). It also checks that all inputs are correct, keeps logs of every action, and is easy to use.

The bot can run in a simulated mode or on the Binance Testnet, so you can safely test your trading strategies without using real money. This project shows how Python, APIs, and automation can be combined to create a practical trading tool.

This project shows how Python, API integration, and automation can be combined to create a practical trading tool.

2 . Objective

- Develop a user-friendly CLI: Allow users to execute trading commands without needing to interact with complex APIs directly.
- Support multiple order types (market and limit).
- Validate inputs: Ensure that users provide valid trading pairs, quantities, and order types to prevent invalid trades.
- Log all actions: Record every activity, including order placement and API responses, for debugging and auditing purposes.
- Support testing on testnet: Allow users to safely test trading commands on Binance's test environment without using real funds.
- Implement advanced trading strategies (optional): Provide support for OCO (One Cancels the Other) and TWAP (Time Weighted Average Price) orders in the advanced module.

3. Project Structure

- src/ – Contains all the Python code for the bot, including modules for market orders, limit orders, API handling, validation, logging, and optional advanced strategies.
- bot.log – Automatically generated log file that records all bot activities and order responses.
- advanced/ – Holds optional advanced order strategies like OCO and TWAP.
- README.md – Provides an overview of the project, usage instructions, and features.
- report.pdf – Final project report with objectives, features, testing details, and conclusions.

```
binance_future_order_bot/
└── src/
    ├── cli.py              # Main CLI interface for user commands
    ├── binance_client.py   # Handles Binance API requests and responses
    ├── market_orders.py    # Logic for placing market orders
    ├── limit_orders.py     # Logic for placing limit orders
    ├── validation.py       # Validates CLI inputs (symbol, side, quantity, price)
    ├── logger_config.py    # Configures logging for tracking bot actions
    └── advanced/
        ├── oco.py          # Optional advanced order strategy (One Cancels the Other)
        └── twap.py         # Optional advanced order strategy (Time Weighted Average Price)
    ├── README.md           # Project description and usage guide
    └── report.pdf          # Final project report for submission
```

4 . Features

1 . Market Orders:

- Executes trades at the current market price.
- Logs every action and shows order status.
- Example:

  " python -m src.cli market BTCUSDT BUY 0.001"

- Returns a simulated FILLED order.

2 . Limit Orders:

- Executes trades only at a specific price.
- Orders remain open until the price is reached.
- Example:

  " python -m src.cli limit BTCUSDT SELL 0.01 62000 "

- Returns a simulated NEW limit order.

3 . Command-Line Interface (CLI):

- Easy commands: market or limit.
- Requires arguments: symbol, side (BUY/SELL), quantity, and price (for limit orders).
- Case-sensitive inputs (BUY/SELL must be uppercase).

4 . Logging:

- Every action is logged with timestamp and details in bot.log.

5. Validation:

- Checks symbols, sides, quantities, and prices to prevent errors.

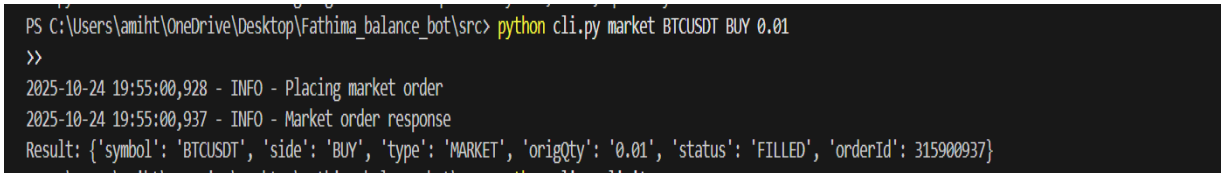
6. Advanced Orders (Optional):

- OCO**:** Two orders where executing one cancels the other.
- TWAP**:** Spread a large order over time to get an average price.

4 . How I Tested

1 . Simulated Market Order:

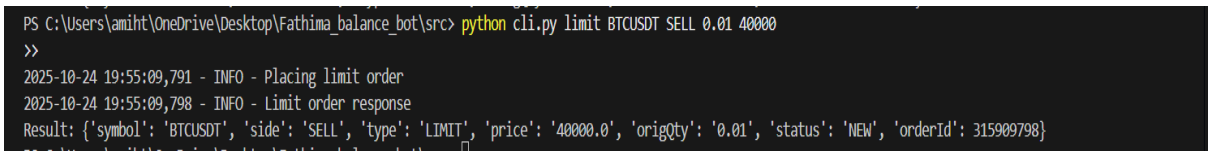
" python cli.py market BTCUSDT BUY 0.01 "



Key	Meaning
symbol	Trading pair → BTCUSDT (Bitcoin/USDT)
side	Order type → BUY
type	Order type → MARKET (executed at current market price)
origQty	Quantity ordered → 0.01 BTC
status	Order status → FILLED (successfully executed)
orderId	Unique order ID → 315900937 (for tracking)

2 . Simulated Limit Order:

"python cli.py limit BTCUSDT SELL 0.01 40000"



Key	Meaning
symbol	Trading pair → BTCUSDT
side	Order type → SELL
type	Order type → LIMIT (will execute only at a specific price)
price	Limit price → 40000.0 USDT
origQty	Quantity → 0.01 BTC
status	NEW → the order is placed but not yet executed
orderId	Unique order ID → 315909798 (for tracking)

6 . Conclusion

Binance Futures Order Bot is an effective tool for automating USDT-M Futures trading on Binance. It allows users to place both market and limit orders with ease, while ensuring all inputs are correct to prevent errors. Every action, including order placement and responses, is recorded in a log for easy tracking and debugging. The bot can be safely tested using simulated orders or on the Binance Testnet, making it possible to practice and verify trading strategies without risking real funds. Overall, this project demonstrates the practical application of Python programming, API integration, and trading automation, resulting in a reliable, user-friendly, and well-documented tool for futures trading.