Binance Futures Order Bot – CLI Based

Name: Mohammed Irfan

Assignment: Binance Futures Order Bot

Date: 9/1/2026

1. Introduction

This project implements a CLI-based trading bot for Binance USDT-M Futures.

The bot supports Market and Limit orders along with advanced strategies such as OCO and TWAP.

The focus is on modular design, input validation, and structured logging.

2. Project Architecture

The system follows a modular structure, with each order type implemented as a separate Python module.

A centralized logger is used to ensure consistent logging across all modules.

Project Structure:

```
binance_bot/
├── src/
|   ├── market_orders.py
|   ├── limit_orders.py
|   ├── logger.py
|   └── advanced/
```

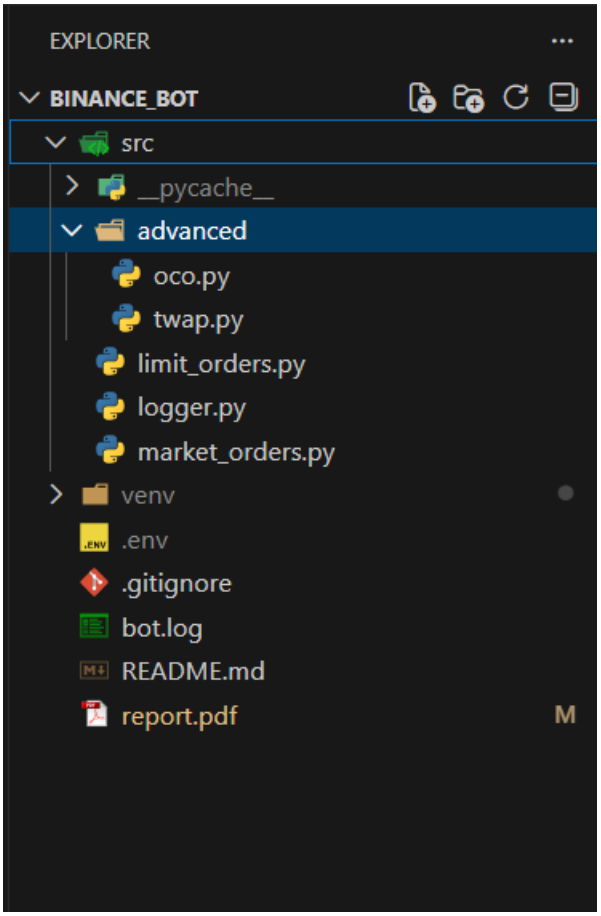
| ├── oco.py

| └── twap.py

├── bot.log

├── README.md

└── report.pdf



3. Implemented Features

Market Order

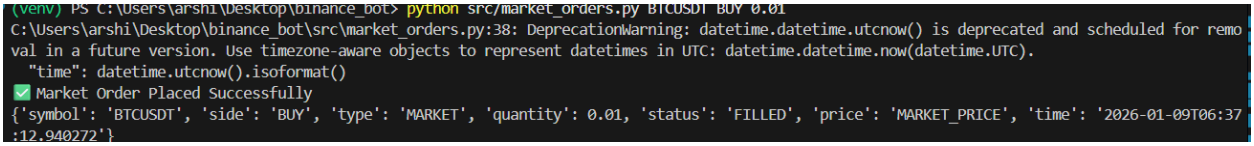
Executes instantly at market price

Validates inputs

Logs execution

Command:

python src/market_orders.py BTCUSDT BUY 0.01



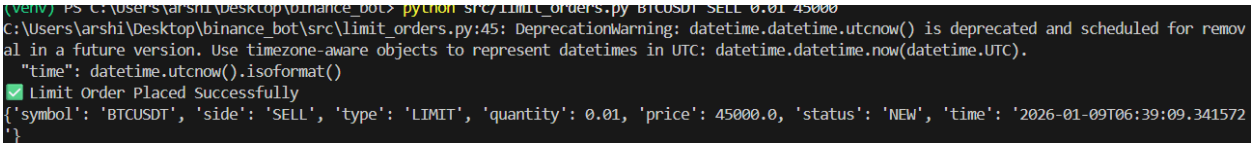
4.Limit Order

Places order at a specified price

Logs order placement

Command:

python src/limit_orders.py BTCUSDT SELL 0.01 45000



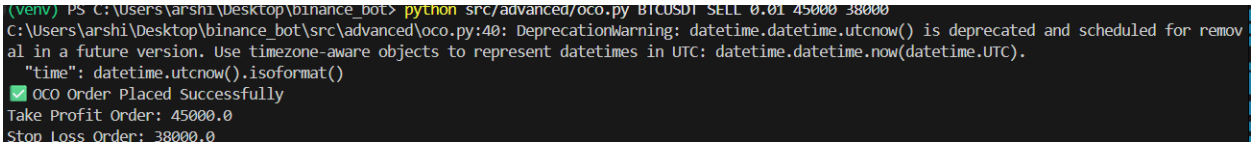
5.OCO Order

OCO places take-profit and stop-loss orders simultaneously.

When one executes, the other is cancelled.

Command:

python src/advanced/oco.py BTCUSDT SELL 0.01 45000 38000



6.TWAP Order

TWAP splits a large order into multiple smaller orders executed at fixed intervals to reduce market impact.

Command:

python src/advanced/twap.py BTCUSDT BUY 1 5 2

```
(venv) PS C:\Users\arshi\Desktop\binance_bot> python src/advanced/twap.py BTCUSDT BUY 1 5 2
📊 TWAP Execution Started
C:\Users\arshi\Desktop\binance_bot\src\advanced\twap.py:48: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for remo
val in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  "time": datetime.utcnow().isoformat()
✅ Slice 1/5 executed → Qty: 0.2
✅ Slice 2/5 executed → Qty: 0.2
✅ Slice 3/5 executed → Qty: 0.2
✅ Slice 4/5 executed → Qty: 0.2
✅ Slice 5/5 executed → Qty: 0.2
🎉 TWAP Execution Completed
```

7.Validation & Logging

All inputs are validated before execution

A centralized logging system records:

Order execution

Errors

Logs are stored in bot.log with timestamps

```
2026-01-09 11:28:50,175 | INFO | Market Order Executed: {'symbol': 'BTCUSDT', 'side': 'BUY', 'type': 'MARKET', 'quantity': 0.01, '
2026-01-09 11:29:17,131 | ERROR | Invalid symbol. Must end with USDT (e.g., BTCUSDT)
2026-01-09 11:30:00,717 | INFO | Limit Order Placed: {'symbol': 'BTCUSDT', 'side': 'SELL', 'type': 'LIMIT', 'quantity': 0.01, 'pric
2026-01-09 11:30:17,184 | ERROR | Quantity must be a positive number
2026-01-09 11:38:14,991 | INFO | Ellipsis
2026-01-09 11:38:14,991 | ERROR | Ellipsis
2026-01-09 11:39:39,797 | INFO | Ellipsis
2026-01-09 11:39:39,797 | ERROR | Ellipsis
2026-01-09 11:39:39,797 | INFO | Market Order Executed: {'symbol': 'BTCUSDT', 'side': 'BUY', 'type': 'MARKET', 'quantity': 0.01, '
2026-01-09 11:43:49,734 | INFO | OCO Order Placed: {'symbol': 'BTCUSDT', 'side': 'SELL', 'quantity': 0.01, 'take_profit': 45000.0,
2026-01-09 11:45:02,331 | INFO | TWAP Started | Symbol=BTCUSDT, Side=BUY, TotalQty=1.0, Slices=5, Interval=2s
2026-01-09 11:45:02,331 | INFO | TWAP Order Executed: {'slice': 1, 'symbol': 'BTCUSDT', 'side': 'BUY', 'quantity': 0.2, 'status':
2026-01-09 11:45:04,335 | INFO | TWAP Order Executed: {'slice': 2, 'symbol': 'BTCUSDT', 'side': 'BUY', 'quantity': 0.2, 'status':
2026-01-09 11:45:06,337 | INFO | TWAP Order Executed: {'slice': 3, 'symbol': 'BTCUSDT', 'side': 'BUY', 'quantity': 0.2, 'status':
2026-01-09 11:45:08,340 | INFO | TWAP Order Executed: {'slice': 4, 'symbol': 'BTCUSDT', 'side': 'BUY', 'quantity': 0.2, 'status':
2026-01-09 11:45:10,343 | INFO | TWAP Order Executed: {'slice': 5, 'symbol': 'BTCUSDT', 'side': 'BUY', 'quantity': 0.2, 'status':
2026-01-09 11:45:10,343 | INFO | TWAP Execution Completed
2026-01-09 12:07:12,940 | INFO | Ellipsis
2026-01-09 12:07:12,940 | ERROR | Ellipsis
2026-01-09 12:07:12,940 | INFO | Market Order Executed: {'symbol': 'BTCUSDT', 'side': 'BUY', 'type': 'MARKET', 'quantity': 0.01, '
2026-01-09 12:09:09,341 | INFO | Market Order Executed: {'symbol': 'BTCUSDT', 'side': 'SELL', 'type': 'LIMIT', 'quantity': 0.01, '
2026-01-09 12:09:09,341 | INFO | Limit Order Placed: {'symbol': 'BTCUSDT', 'side': 'SELL', 'type': 'LIMIT', 'quantity': 0.01, 'pric
2026-01-09 12:10:34,871 | INFO | OCO Order Placed: {'symbol': 'BTCUSDT', 'side': 'SELL', 'quantity': 0.01, 'take_profit': 45000.0,
```

8.API Mode

The bot runs in mock mode for safety.

It can be easily connected to Binance Futures API or Testnet by replacing mock logic and adding API keys.

9.Conclusion

This project demonstrates a clean and extensible approach to building automated trading systems.

Advanced order strategies and structured logging improve reliability and maintainability.