Binance Futures Order Bot

Name: Golu Kumar Gupta
Role Applied: Python Developer Intern
Project Type: CLI-Based Trading Bot
Mode: Binance USDT-M Futures Testnet

---

1. Introduction

This project is a command-line based trading bot developed in Python for Binance USDT-M Futures.
The bot allows users to place different types of futures orders using a simple CLI interface with built-in input validation, structured logging, and modular architecture.

The primary goal of this project is to demonstrate:

- Python programming proficiency
- REST API integration
- Error handling and validation
- Clean modular code design
- Practical understanding of trading order types

---

2. Features Implemented

Core Orders (Mandatory)

- Market Orders
- Limit Orders

Advanced Orders

- OCO (One-Cancels-the-Other) – Stop Loss + Take Profit
- TWAP (Time Weighted Average Price)

Additional Features

- CLI using argparse
- Interactive menu mode
- Input validation
- Structured logging (bot.log)

- Binance Futures Testnet support

---

3. Technology Stack

- Python 3
- python-binance
- argparse
- python-dotenv
- Logging module

---

4. Project Structure

```
golu_kumar_gupta_binance_bot/
 ├── src/
|   ├── market_orders.py
|   ├── limit_orders.py
|   ├── main.py
|   ├── utils.py
|   ├── config.py
|   └── advanced/
|       ├── oco.py
|       └── twap.py
 ├── bot.log
 ├── README.md
 ├── report.pdf
 ├── requirements.txt
└── .env.example
```

---

5. Setup Instructions

1. Install dependencies:

pip install -r requirements.txt

2. Create .env file:

BINANCE_API_KEY=your_testnet_key

BINANCE_API_SECRET=your_testnet_secret

3. Run the bot:

python src/main.py

---

6. Usage Examples

Market Order

python src/main.py market BTCUSDT BUY 0.001

Limit Order

python src/main.py limit BTCUSDT BUY 0.001 50000

OCO Order

python src/main.py oco BTCUSDT BUY 0.001 40000 60000

TWAP Strategy

python src/main.py twap BTCUSDT BUY 0.01 60 10

---

7. Logging Mechanism

All actions such as order placement, errors, and validations are logged into bot.log with timestamps.

Log format:

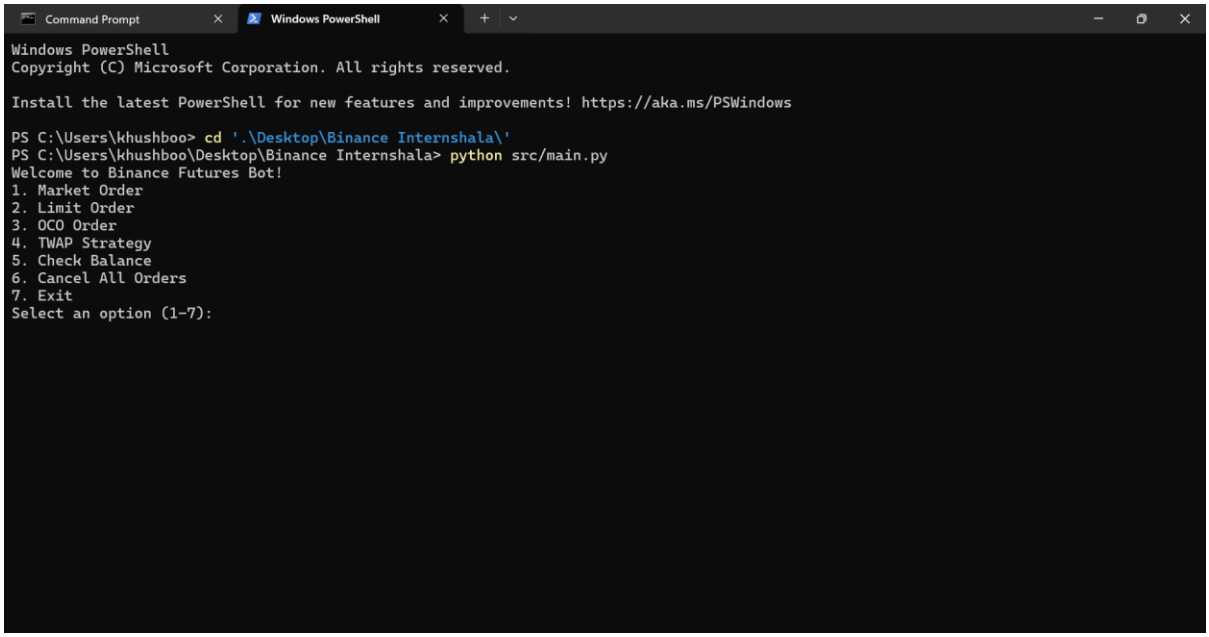
YYYY-MM-DD HH:MM:SS - LEVEL - Message

---

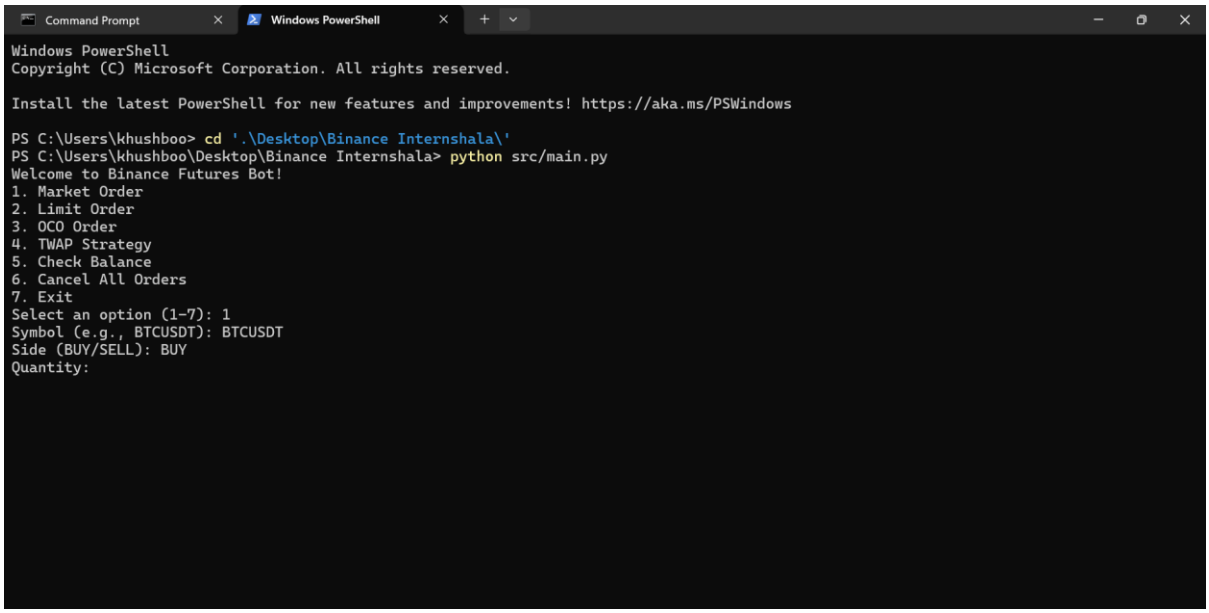
8. Screenshots & Execution Proof

[We can just run main.py (and use CLI interface) or type all command at a time to save time.]

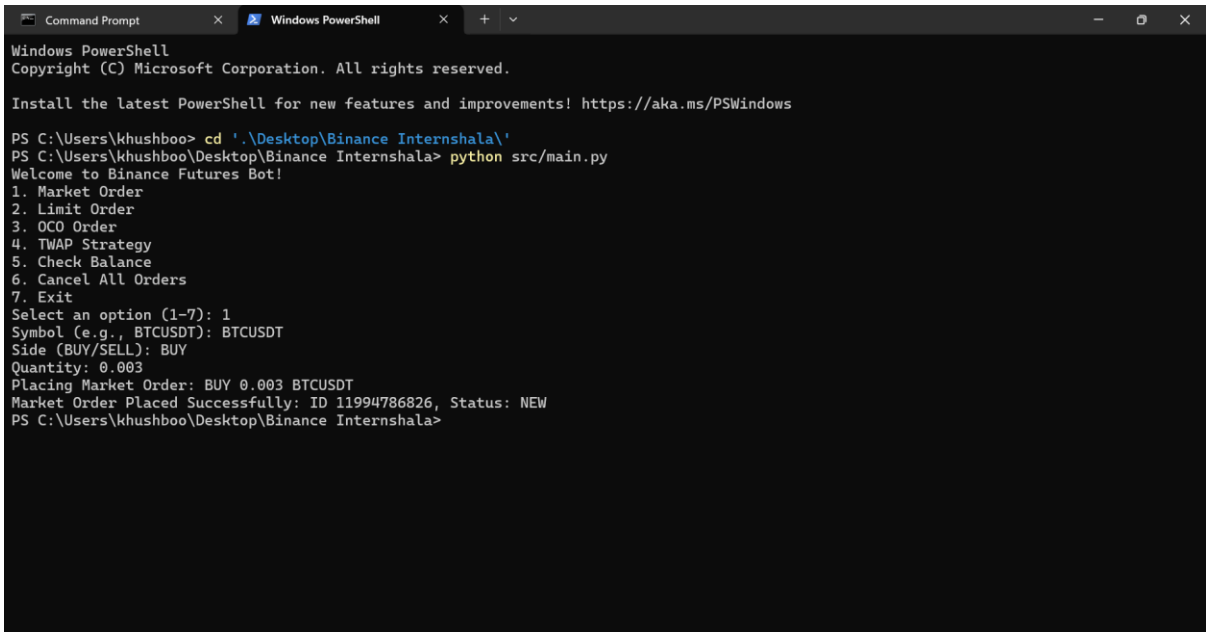
1.Using CLI (step by step to make operations):



After selecting based on our choice:



Similarly selecting any option and giving input then we can perform any operation :



2. And can also directly insert all command at a time to make order at single time

8.1 Market Order Execution

```
PS C:\Users\khushboo\Desktop\Binance Internshala> python src/main.py market BTCUSDT BUY 0.003
Placing Market Order: BUY 0.003 BTCUSDT
Market Order Placed Successfully: ID 11994861758, Status: NEW
```

8.2 Limit Order Execution

```
PS C:\Users\khushboo\Desktop\Binance Internshala> python src/main.py limit BTCUSDT BUY 0.003 50000
Placing Limit Order: BUY 0.003 BTCUSDT at 50000.0
Limit Order Placed Successfully: ID 11994863586, Status: NEW
```

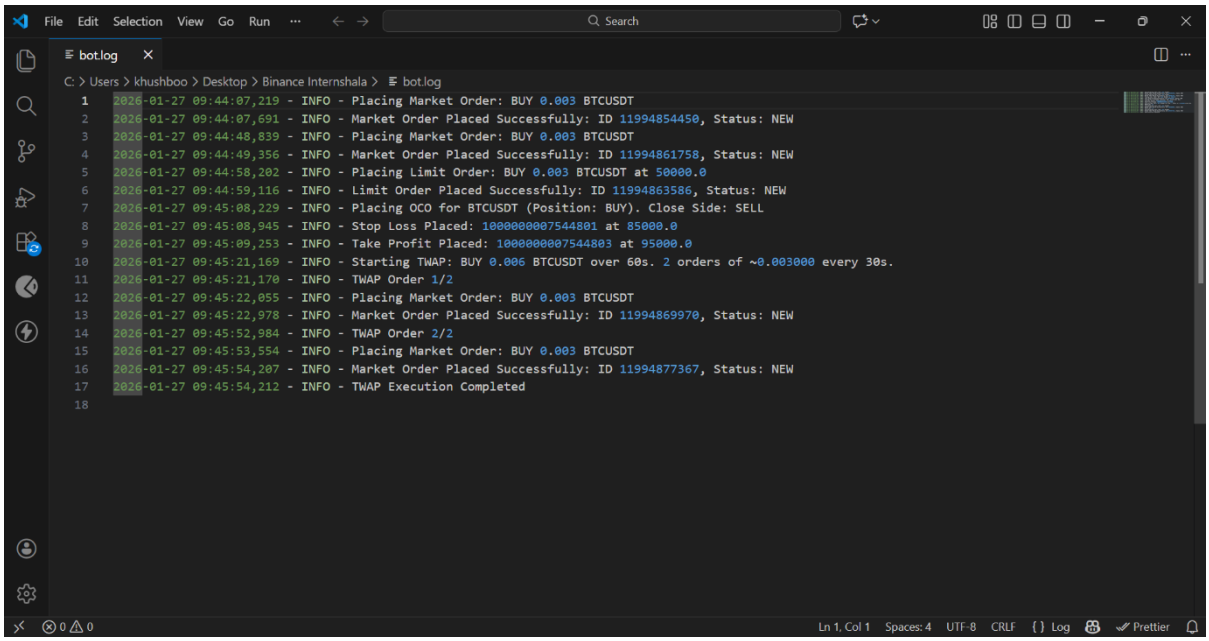
8.3 OCO Order Execution

```
PS C:\Users\khushboo\Desktop\Binance Internshala> python src/main.py oco BTCUSDT BUY 0.003 85000 95000
Placing OCO for BTCUSDT (Position: BUY). Close Side: SELL
Stop Loss Placed: 1000000007544801 at 85000.0
Take Profit Placed: 1000000007544803 at 95000.0
```

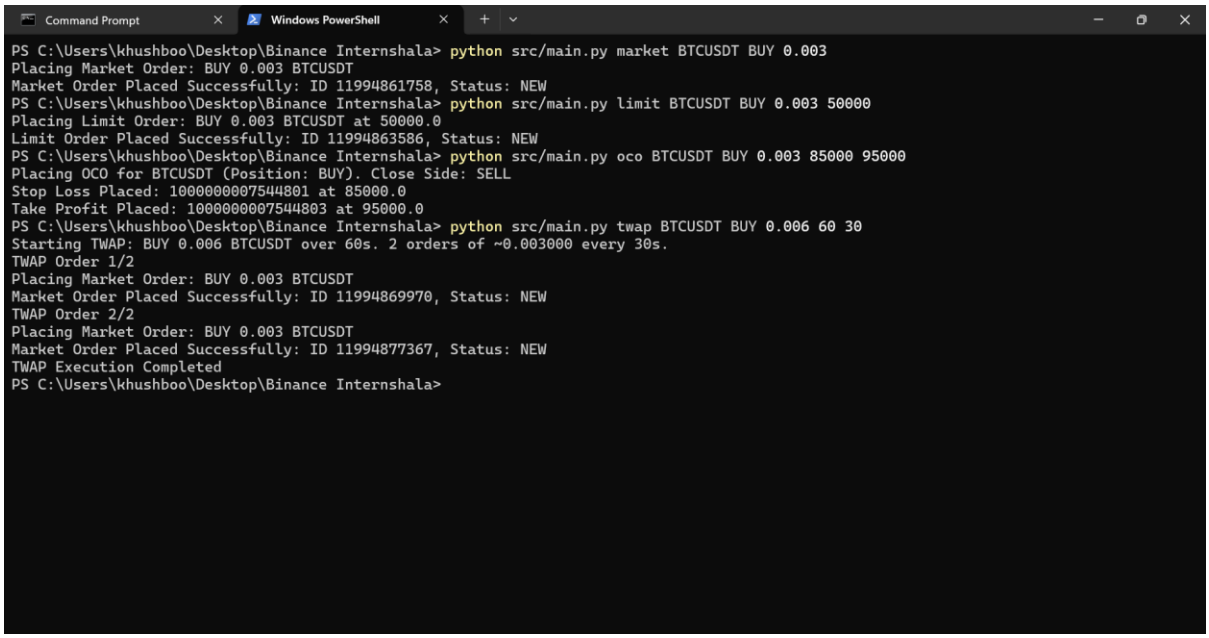
8.4 TWAP Execution

```
PS C:\Users\khushboo\Desktop\Binance Internshala> python src/main.py twap BTCUSDT BUY 0.006 60 30
Starting TWAP: BUY 0.006 BTCUSDT over 60s. 2 orders of ~0.003000 every 30s.
TWAP Order 1/2
Placing Market Order: BUY 0.003 BTCUSDT
Market Order Placed Successfully: ID 11994869970, Status: NEW
TWAP Order 2/2
Placing Market Order: BUY 0.003 BTCUSDT
Market Order Placed Successfully: ID 11994877367, Status: NEW
TWAP Execution Completed
PS C:\Users\khushboo\Desktop\Binance Internshala>
```

8.5 bot.log File Entries



Screenshot containing all operations:



9. Validation & Error Handling

The bot validates:

- Trading symbol format
- Order side (BUY/SELL)
- Positive quantity and prices

If invalid input is detected, execution stops and an appropriate error message is logged.

---

10. Conclusion

This project demonstrates the development of a robust and modular Binance Futures trading bot using Python.
The implementation follows clean coding practices, includes advanced trading strategies, and provides reproducible execution steps.
