Title: Binance Futures Trading Bot

Author: `Ujjwal Tiwari` **Date:** August 12, 2025

1. Introduction

Objective The primary objective of this project was to develop a command-line interface (CLI) based trading bot for the Binance USDT-M Futures market. The bot needed to support multiple order types with robust logging, input validation, and clear documentation.

Scope & Key Features This project successfully implements the mandatory market and limit orders, along with two advanced order types: stop-limit orders and a Time-Weighted Average Price (TWAP) algorithmic strategy.

The key features of the bot include:

- A modular file structure separating different order types.
- A robust CLI for executing trades, built with Python's `argparse` library.
- Structured logging of all actions, errors, and API responses to `bot.log`.
- Direct interaction with the Binance Futures Testnet API.

2. Project Architecture

File Structure The project is organized into a `src` directory containing the core logic and a main `bot.py` file that serves as the entry point for the CLI.

- `bot.py`: Handles command-line argument parsing and calls the appropriate functions.
- `src/client.py`: Manages the connection to the Binance API.
- `src/logger.py`: Configures the structured logger.
- `src/market_orders.py` & `src/limit_orders.py`: Contain the logic for the core order types.
- `src/advanced/`: A dedicated folder for advanced order strategies like Stop-Limit and TWAP.

CLI Design The bot uses Python's `argparse` library with subparsers for each command (`market`, `limit`, `stoplimit`, `twap`). This design makes the CLI intuitive to use and easy to extend with new order types in the future.

3. Implemented Features

Core Orders (Market & Limit) Market and limit orders were implemented using the `futures_create_order` method from the `python-binance` library. The functions include validation to ensure order quantity and price are positive numbers before sending the request to the API.

Advanced Orders

- **Stop-Limit Order:** This was also implemented using the `futures_create_order` method, but with `type='STOP'`. This order type requires both a `stopPrice` (the trigger price) and a `price` (the limit price of the subsequent order), providing a powerful tool for risk management.
- **TWAP Strategy:** This algorithmic strategy was implemented to execute a large order over a specified period without requiring a special API endpoint. The `execute_twap_order` function takes a total quantity and duration in minutes. It then calculates the number of smaller "chunk" orders to place (one per minute) and enters a loop. In each iteration, it places a standard market order for the chunk size and then pauses execution for 60 seconds using `time.sleep()`. This effectively spreads the trade over time to reduce market impact.

Logging and Error Handling Structured logging was implemented using Python's built-in `logging` module. All actions and API responses are logged with timestamps to both the console and the `bot.log` file. All API calls are wrapped in `try...except` blocks to gracefully handle potential API errors from Binance, logging the specific error code and message without crashing the program.

4. Challenges and Solutions

Several challenges were encountered during development:

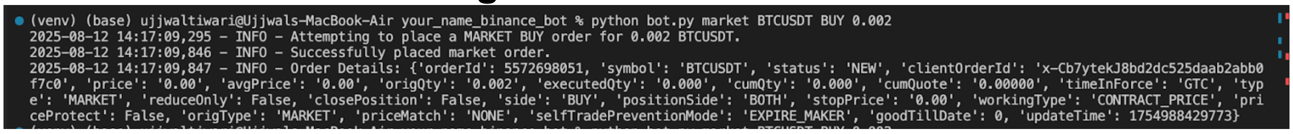
- **Python Environment Conflict:** A significant challenge was a persistent `ModuleNotFoundError`. Diagnostic scripts revealed a conflict between a global `pyenv` Python installation and the project's local `venv`. The solution was to abandon the `venv` and directly fix the library in the `pyenv` installation that was being used by the system. A compatible, older version of the `binance.client` was used to ensure stability.
- **API Trading Rule Compliance:** After establishing a connection, the bot faced several API errors related to Binance's strict trading rules.

- **Minimum Notional Value:** An `APIError(code=-4164)` occurred because the order's total value (quantity * price) was below the 100 USDT minimum for BTCUSDT. This was solved by increasing the order quantity.
- **Quantity Precision:** This led to an `APIError(code=-1111)`, as the new quantity had too many decimal places. The solution was to find the correct quantity precision for BTCUSDT (3 decimal places) and use a quantity that satisfied both the precision and minimum value rules.

5. Results & Screenshots

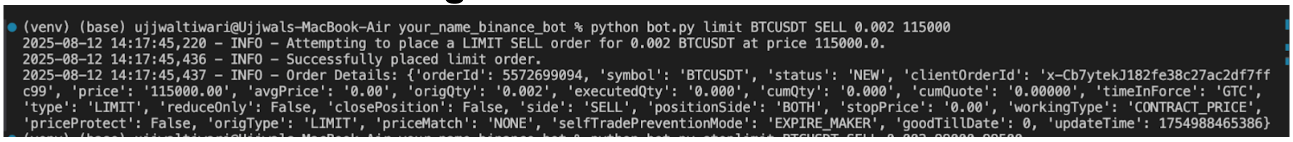
The bot was successfully tested on the Binance Futures Testnet for all implemented order types.

Market Order Success Log:



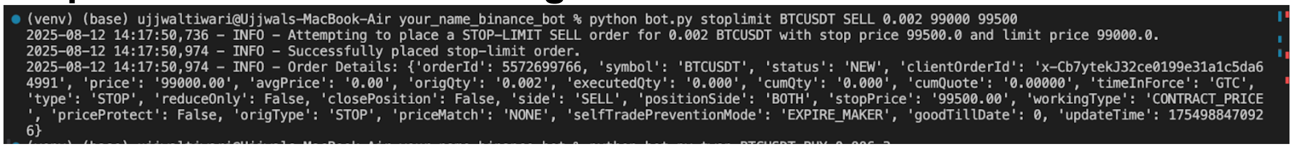
```
● (venv) (base) ujjwaltiwari@Ujjwals-MacBook-Air your_name_binance_bot % python bot.py market BTCUSDT BUY 0.002
  2025-08-12 14:17:09,295 - INFO - Attempting to place a MARKET BUY order for 0.002 BTCUSDT.
  2025-08-12 14:17:09,846 - INFO - Successfully placed market order.
  2025-08-12 14:17:09,847 - INFO - Order Details: {'orderId': 5572698051, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ8bd2dc525daab2abb0
  f7c0', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'typ
  e': 'MARKET', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'pri
  ceProtect': False, 'origType': 'MARKET', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1754988429773}
```

Limit Order Success Log:



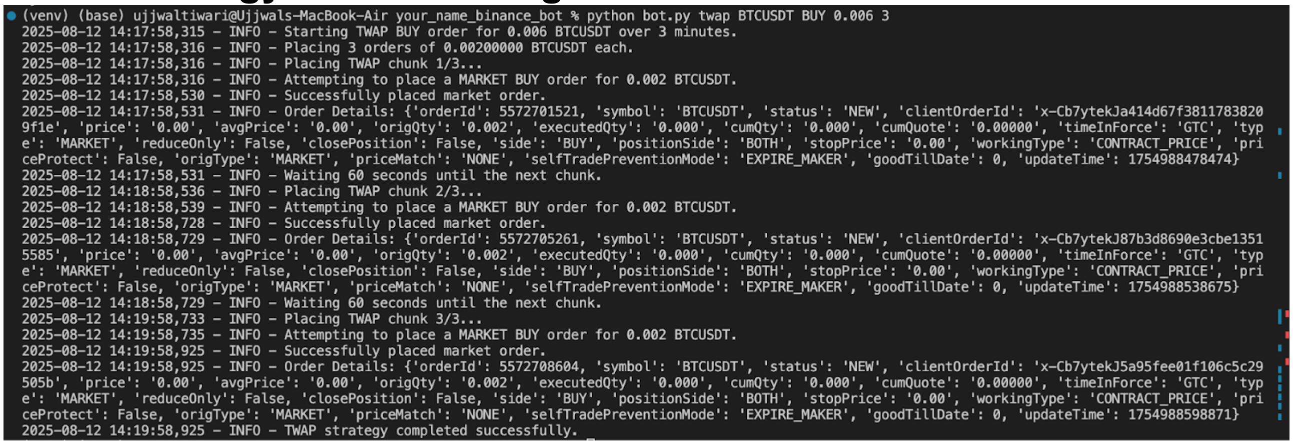
```
● (venv) (base) ujjwaltiwari@Ujjwals-MacBook-Air your_name_binance_bot % python bot.py limit BTCUSDT SELL 0.002 115000
  2025-08-12 14:17:45,220 - INFO - Attempting to place a LIMIT SELL order for 0.002 BTCUSDT at price 115000.0.
  2025-08-12 14:17:45,436 - INFO - Successfully placed limit order.
  2025-08-12 14:17:45,437 - INFO - Order Details: {'orderId': 5572699094, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ182fe38c27ac2df7ff
  c99', 'price': '115000.00', 'avgPrice': '0.00', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC',
  'type': 'LIMIT', 'reduceOnly': False, 'closePosition': False, 'side': 'SELL', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE',
  'priceProtect': False, 'origType': 'LIMIT', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1754988465386}
```

Stop-Limit Order Success Log:

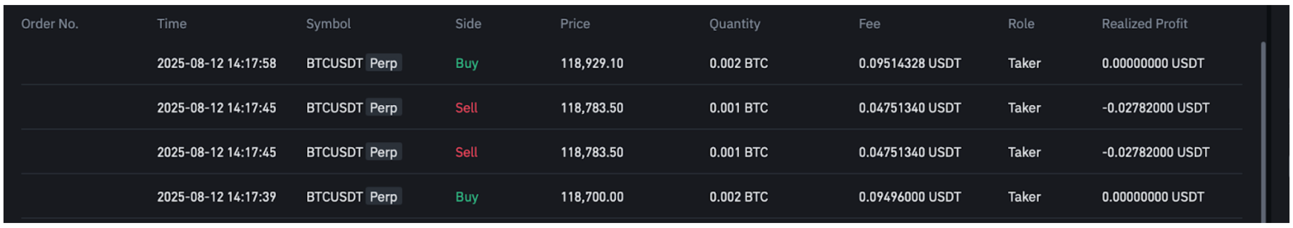


```
● (venv) (base) ujjwaltiwari@Ujjwals-MacBook-Air your_name_binance_bot % python bot.py stoplimit BTCUSDT SELL 0.002 99000 99500
  2025-08-12 14:17:50,736 - INFO - Attempting to place a STOP-LIMIT SELL order for 0.002 BTCUSDT with stop price 99500.0 and limit price 99000.0.
  2025-08-12 14:17:50,974 - INFO - Successfully placed stop-limit order.
  2025-08-12 14:17:50,974 - INFO - Order Details: {'orderId': 5572699766, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ32ce0199e31a1c5da6
  4991', 'price': '99000.00', 'avgPrice': '0.00', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC',
  'type': 'STOP', 'reduceOnly': False, 'closePosition': False, 'side': 'SELL', 'positionSide': 'BOTH', 'stopPrice': '99500.00', 'workingType': 'CONTRACT_PRICE
  ', 'priceProtect': False, 'origType': 'STOP', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 175498847092
  6}
```

TWAP Strategy Execution Log:



```
● (venv) (base) ujjwaltiwari@Ujjwals-MacBook-Air your_name_binance_bot % python bot.py twap BTCUSDT BUY 0.006 3
  2025-08-12 14:17:58,315 - INFO - Starting TWAP BUY order for 0.006 BTCUSDT over 3 minutes.
  2025-08-12 14:17:58,316 - INFO - Placing 3 orders of 0.00200000 BTCUSDT each.
  2025-08-12 14:17:58,316 - INFO - Placing TWAP chunk 1/3...
  2025-08-12 14:17:58,316 - INFO - Attempting to place a MARKET BUY order for 0.002 BTCUSDT.
  2025-08-12 14:17:58,530 - INFO - Successfully placed market order.
  2025-08-12 14:17:58,531 - INFO - Order Details: {'orderId': 5572701521, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJa414d67f3811783820
  9f1e', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'typ
  e': 'MARKET', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'pri
  ceProtect': False, 'origType': 'MARKET', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1754988478474}
  2025-08-12 14:17:58,531 - INFO - Waiting 60 seconds until the next chunk.
  2025-08-12 14:18:58,536 - INFO - Placing TWAP chunk 2/3...
  2025-08-12 14:18:58,539 - INFO - Attempting to place a MARKET BUY order for 0.002 BTCUSDT.
  2025-08-12 14:18:58,728 - INFO - Successfully placed market order.
  2025-08-12 14:18:58,729 - INFO - Order Details: {'orderId': 5572705261, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ87b3d8690e3cbe1351
  5585', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'typ
  e': 'MARKET', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'pri
  ceProtect': False, 'origType': 'MARKET', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1754988538675}
  2025-08-12 14:18:58,729 - INFO - Waiting 60 seconds until the next chunk.
  2025-08-12 14:19:58,733 - INFO - Placing TWAP chunk 3/3...
  2025-08-12 14:19:58,735 - INFO - Attempting to place a MARKET BUY order for 0.002 BTCUSDT.
  2025-08-12 14:19:58,925 - INFO - Successfully placed market order.
  2025-08-12 14:19:58,925 - INFO - Order Details: {'orderId': 5572708604, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clientOrderId': 'x-Cb7ytekJ5a95fee01f106c5c29
  505b', 'price': '0.00', 'avgPrice': '0.00', 'origQty': '0.002', 'executedQty': '0.000', 'cumQty': '0.000', 'cumQuote': '0.00000', 'timeInForce': 'GTC', 'typ
  e': 'MARKET', 'reduceOnly': False, 'closePosition': False, 'side': 'BUY', 'positionSide': 'BOTH', 'stopPrice': '0.00', 'workingType': 'CONTRACT_PRICE', 'pri
  ceProtect': False, 'origType': 'MARKET', 'priceMatch': 'NONE', 'selfTradePreventionMode': 'EXPIRE_MAKER', 'goodTillDate': 0, 'updateTime': 1754988598871}
  2025-08-12 14:19:58,925 - INFO - TWAP strategy completed successfully.
```



Order No.	Time	Symbol	Side	Price	Quantity	Fee	Role	Realized Profit
	2025-08-12 14:17:58	BTCUSDT Perp	Buy	118,929.10	0.002 BTC	0.09514328 USDT	Taker	0.00000000 USDT
	2025-08-12 14:17:45	BTCUSDT Perp	Sell	118,783.50	0.001 BTC	0.04751340 USDT	Taker	-0.02782000 USDT
	2025-08-12 14:17:45	BTCUSDT Perp	Sell	118,783.50	0.001 BTC	0.04751340 USDT	Taker	-0.02782000 USDT
	2025-08-12 14:17:39	BTCUSDT Perp	Buy	118,700.00	0.002 BTC	0.09496000 USDT	Taker	0.00000000 USDT

6. Conclusion

This project successfully created a functional, command-line-based trading bot capable of executing market, limit, stop-limit, and TWAP orders on the Binance Futures Testnet. Key challenges related to environment setup and API rule compliance were overcome, resulting in a robust and well-documented application. Future improvements could include implementing Grid orders, adding a graphical user interface (GUI), or integrating more complex trading indicators.