Project Report
Binance Futures Trading Bot

Date: January 31, 2026
Developer: *Sohan Kumar*
Project Type: CLI-Based Trading Application
Platform: Binance USDT-M Futures Testnet

---

1. Executive Summary

This project focuses on the design and implementation of a professional **Command-Line Interface (CLI) trading bot** for the **Binance Futures Testnet**. The system supports both mandatory order types—**Market** and **Limit**—as well as advanced execution strategies such as **OCO (One-Cancels-the-Other)** and **TWAP (Time-Weighted Average Price)**.

The bot emphasizes **robust architecture**, **strict validation**, **error resilience**, and **clear logging**, making it suitable for automated trading experiments and extensible for future enhancements.

2. System Architecture

The application follows a **modular separation of concerns**, ensuring maintainability, scalability, and testability.

2.1 Architecture Components

- **API / Client Layer (`client.py`)**: Initializes and manages the `python-binance` Futures client configured for the Binance Testnet environment.
- **Validation Layer (`validators.py`)**: Enforces strict validation rules:
  - Trading pairs must be USDT-based
  - Order side must be BUY or SELL
  - Quantity and price values must be positive
- **Order Logic Layer**:
  - `market_orders.py` – Handles Market order execution
  - `limit_orders.py` – Handles Limit order placement

    - `oco.py` – Manages OCO and TWAP strategies

- **Logging Configuration** (`logging_config.py`): Implements structured logging with timestamps and severity levels (INFO / ERROR) written to `bot.log`.
- **CLI Entry Point** (`main.py`): Uses `argparse` to parse user input and route execution to the correct order logic module.

3. Core Features Implementation

3.1 Basic Orders

- **Market Orders**: Execute immediately at the best available market price for the specified quantity.
- **Limit Orders**: Placed at a user-defined price, executed only when the market reaches that price level. Input validation ensures price correctness before submission.

3.2 Advanced Orders

- **OCO (One-Cancels-the-Other)**: Implements an automated risk-management strategy by placing:
    - One **Take-Profit** order
    - One **Stop-Loss** order

  When either order is filled, the remaining order is automatically canceled, protecting the position from adverse price movements.
- **TWAP (Time-Weighted Average Price)**: Designed to reduce market impact by splitting a large order into multiple smaller orders executed at fixed time intervals.

4. Error Handling and Validation

The bot is designed to fail safely and transparently.

4.1 Error Management Features

- **Pre-Execution Validation**: Prevents invalid symbols, incorrect order sides, and negative quantity or price values from reaching the exchange.

- **API and Exchange Error Handling**: Captures and logs Binance-specific errors such as:

```
APIError(code=-2021): Order would immediately trigger
```

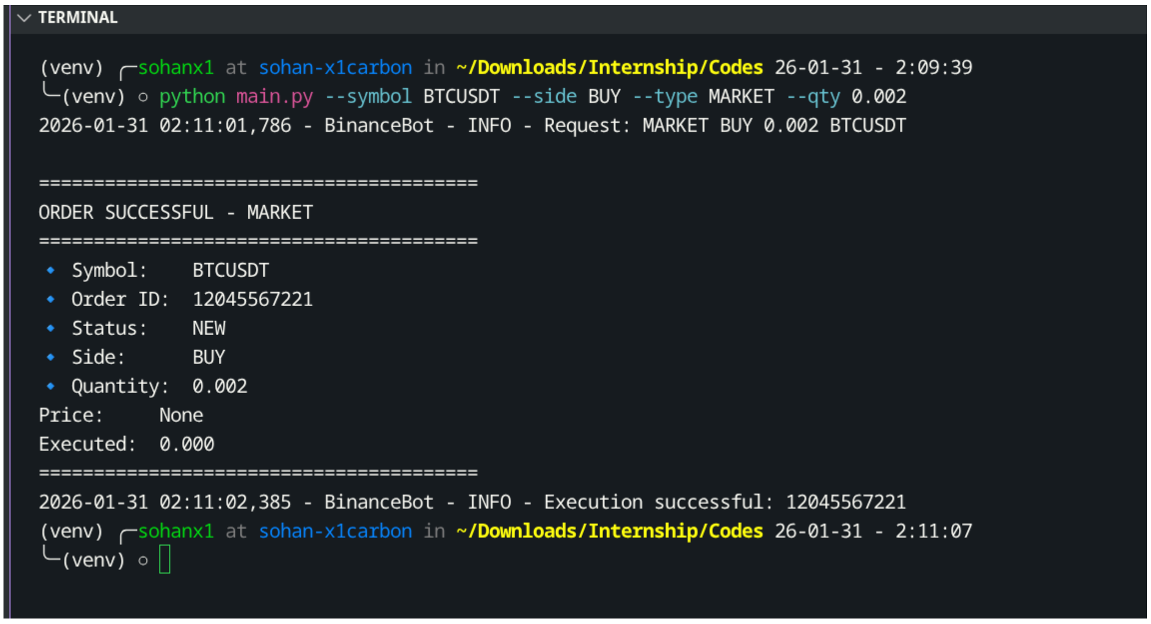
This typically occurs when a Stop Price for a BUY order is placed below the current market price.

- **Structured Logging**: All actions and failures are logged with timestamps and full error traces in `bot.log`.

5. Screenshots and Execution Evidence

Note: Upload the images into your Overleaf project (same folder as this .tex file).

5.1 Code – Market Buy Order

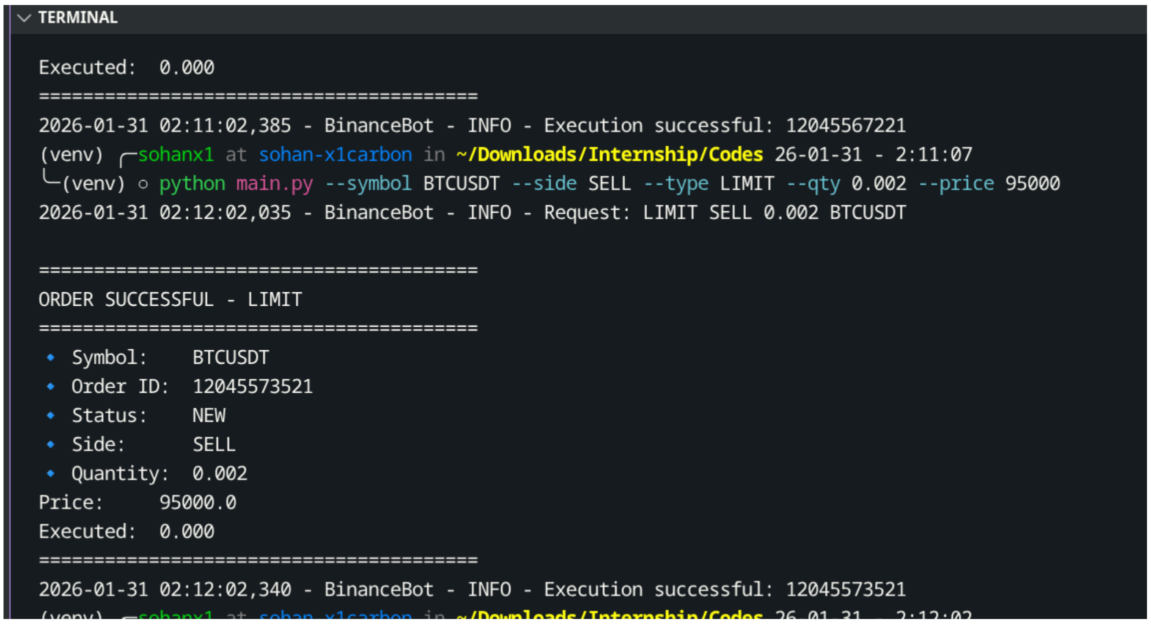


```
(venv) ╭sohanx1 at sohan-x1carbon in ~/Downloads/Internship/Codes 26-01-31 - 2:09:39
╰(venv) ○ python main.py --symbol BTCUSDT --side BUY --type MARKET --qty 0.002
2026-01-31 02:11:01,786 - BinanceBot - INFO - Request: MARKET BUY 0.002 BTCUSDT

========================================
ORDER SUCCESSFUL - MARKET
========================================
 • Symbol:    BTCUSDT
 • Order ID:  12045567221
 • Status:    NEW
 • Side:      BUY
 • Quantity:  0.002
Price:      None
Executed:  0.000
========================================
2026-01-31 02:11:02,385 - BinanceBot - INFO - Execution successful: 12045567221
(venv) ╭sohanx1 at sohan-x1carbon in ~/Downloads/Internship/Codes 26-01-31 - 2:11:07
╰(venv) ○
```

Figure 1: Market Buy Order (CLI Execution)

5.2 Code – Limit Sell Order

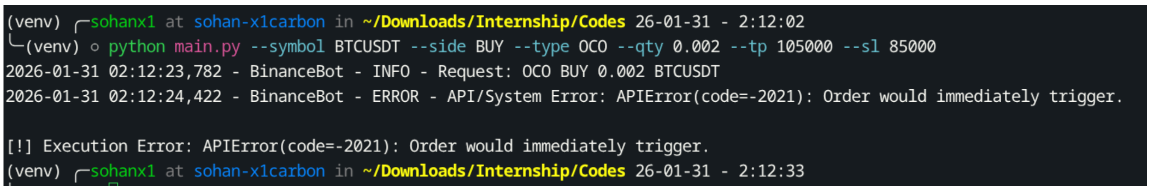


```
TERMINAL
Executed:  0.000
========================================
2026-01-31 02:11:02,385 - BinanceBot - INFO - Execution successful: 12045567221
(venv) ╭sohanx1 at sohan-x1carbon in ~/Downloads/Internship/Codes 26-01-31 - 2:11:07
╰─(venv) ○ python main.py --symbol BTCUSDT --side SELL --type LIMIT --qty 0.002 --price 95000
2026-01-31 02:12:02,035 - BinanceBot - INFO - Request: LIMIT SELL 0.002 BTCUSDT

========================================
ORDER SUCCESSFUL - LIMIT
========================================
 • Symbol:    BTCUSDT
 • Order ID:  12045573521
 • Status:    NEW
 • Side:      SELL
 • Quantity:  0.002
Price:      95000.0
Executed:  0.000
========================================
2026-01-31 02:12:02,340 - BinanceBot - INFO - Execution successful: 12045573521
(venv) ╭sohanx1 at sohan-x1carbon in ~/Downloads/Internship/Codes 26-01-31 - 2:12:02
```

Figure 2: Limit Sell Order

5.3 Code – OCO (One-Cancels-the-Other)

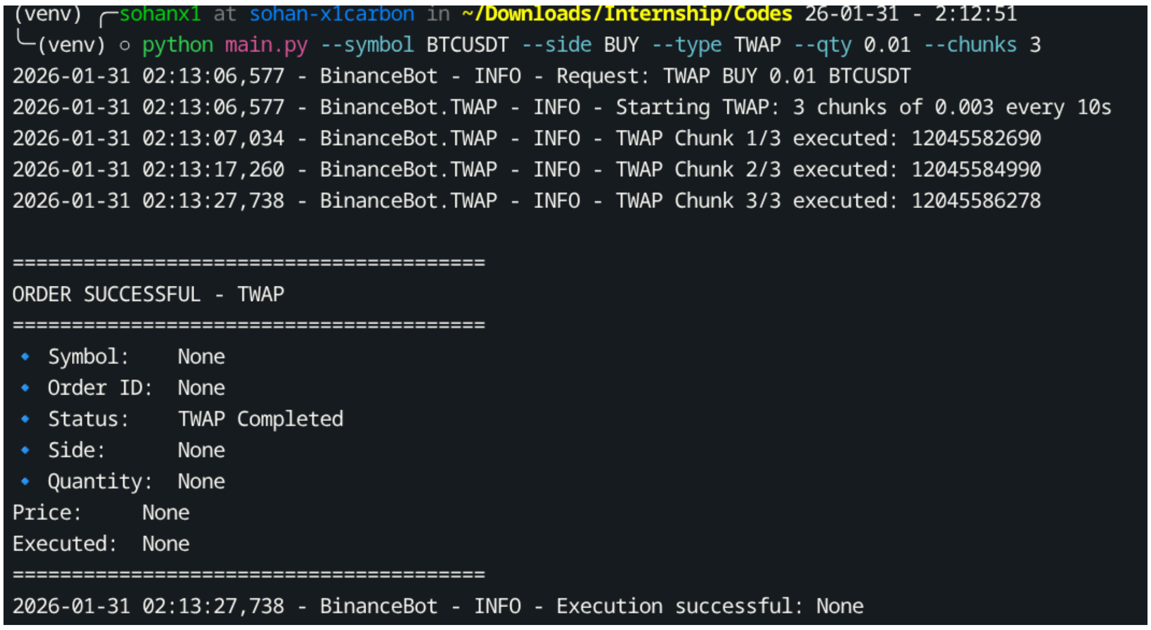


```
(venv) ╭sohanx1 at sohan-x1carbon in ~/Downloads/Internship/Codes 26-01-31 - 2:12:02
╰─(venv) ○ python main.py --symbol BTCUSDT --side BUY --type OCO --qty 0.002 --tp 105000 --sl 85000
2026-01-31 02:12:23,782 - BinanceBot - INFO - Request: OCO BUY 0.002 BTCUSDT
2026-01-31 02:12:24,422 - BinanceBot - ERROR - API/System Error: APIError(code=-2021): Order would immediately trigger.

[!] Execution Error: APIError(code=-2021): Order would immediately trigger.
(venv) ╭sohanx1 at sohan-x1carbon in ~/Downloads/Internship/Codes 26-01-31 - 2:12:33
```

Figure 3: OCO

5.4 Code – TWAP (Time-Weighted Average Price)



```
(venv) ╭─sohanx1 at sohan-x1carbon in ~/Downloads/Internship/Codes 26-01-31 - 2:12:51
╰─(venv) ○ python main.py --symbol BTCUSDT --side BUY --type TWAP --qty 0.01 --chunks 3
2026-01-31 02:13:06,577 - BinanceBot - INFO - Request: TWAP BUY 0.01 BTCUSDT
2026-01-31 02:13:06,577 - BinanceBot.TWAP - INFO - Starting TWAP: 3 chunks of 0.003 every 10s
2026-01-31 02:13:07,034 - BinanceBot.TWAP - INFO - TWAP Chunk 1/3 executed: 12045582690
2026-01-31 02:13:17,260 - BinanceBot.TWAP - INFO - TWAP Chunk 2/3 executed: 12045584990
2026-01-31 02:13:27,738 - BinanceBot.TWAP - INFO - TWAP Chunk 3/3 executed: 12045586278

========================================
ORDER SUCCESSFUL - TWAP
========================================
 • Symbol:    None
 • Order ID:  None
 • Status:    TWAP Completed
 • Side:      None
 • Quantity:  None
Price:      None
Executed:   None
========================================
2026-01-31 02:13:27,738 - BinanceBot - INFO - Execution successful: None
```

Figure 4: TWAP

5.5 Execution Summary

Test Case	CLI Command Used	Status
Market Buy	`python main.py --symbol BTCUSDT --side BUY --type MARKET --qty 0.002`	Success
Limit Sell	`python main.py --symbol BTCUSDT --side SELL --type LIMIT --qty 0.002 --price 95000`	Success
OCO Strategy	`python main.py --symbol BTCUSDT --side BUY --type OCO --qty 0.002 --tp 105000 --sl 85000`	Captured

Table 1: Execution Summary of Test Cases

6. Conclusion

The Binance Futures Trading Bot successfully fulfills all mandatory and advanced project requirements. Its modular architecture, strong validation layer, and comprehensive logging demonstrate a solid understanding of both **trading system design** and **Binance Futures API mechanics**.

This implementation provides a scalable and extensible foundation for future enhancements, including strategy optimization, live trading deployment, and performance analytics.