

# Binance Futures Order Bot

## (TestNET)

### 1.1 Project Goal

The objective was to develop a comprehensive trading solution for the Binance USDT-M Futures Testnet, encompassing core trade execution like market order and limit order, advanced algorithmic strategies (TWAP, OCO), robust validation, and a user-friendly frontend interface.

### 1.2 Modular Architecture

```
Sulogno_Sarkar_trading_bot/
|
├── app.py          # 🎨 Streamlit dashboard (UI for trading actions)
├── .env            # 🔑 API keys and credentials (Binance Testnet)
├── bot.log          # 📋 Log file for executed trades and errors
├── README.md        # 💡 Project description and usage instructions
├── report.pdf       # 📄 Final project report
|
├── src/             # 🛠️ Core backend logic
|   ├── utils.py      # 🛠️ Utility functions (client setup, precision, helpers)
|   ├── market_orders.py # 💰 Market order execution
|   ├── limit_orders.py # ⚒ Limit order execution with tick/step handling
|   |
|   ├── advanced/      # 💭 Advanced trading strategies
|   |   ├── oco.py       # 💯 One-Cancels-the-Other (OCO) TP/SL logic
|   |   ├── twap.py      # ⏳ Time-Weighted Average Price (TWAP) execution
|   |   └── __pycache__/  # 🐍 Compiled Python bytecode (auto-generated)
|   |
|   └── __pycache__/    # 🐍 Compiled backend cache (auto-generated)
|
└── requirements.txt # 📦 Dependencies list for deployment
```

The solution uses a highly modular Python structure, facilitating independent development and easy maintenance:

- **Execution Logic (src/\*.py):** Dedicated files handle the placement of Market, Limit, OCO, and TWAP orders.
- **Utilities (src/utils.py):** Centralized module handling **API connectivity (Testnet)**, exchange info caching, and enforcing **Binance's critical precision rules** (tickSize, stepSize).
- **Frontend (app.py):** A **Lightweight Streamlit Web Application** that wraps the core functions, providing a dashboard for execution and monitoring.

## 2. Fulfillment of Mandatory Requirements

Requirement	Implementation Detail	Proof of Success
<b>Market &amp; Limit Orders</b>	Implemented in dedicated files using <code>client.futures_create_order</code> . Supports dynamic BUY and SELL sides.	Verified via Testnet execution; logs confirm NEW or FILLED status.
<b>Robust Validation</b>	The <code>validate_order</code> function in <code>utils.py</code> checks user inputs against exchange <code>PRICE_FILTER</code> and <code>LOT_SIZE</code> rules <b>before</b> sending the API request.	Testing with invalid price/quantity precision resulted in logged <code>ValueError</code> exceptions, proving client-side checks are effective.
<b>Logging &amp; Errors</b>	All actions (startup, API success/failure) are logged to <code>bot.log</code> with timestamps and structured messages.	Log analysis confirms successful <code>LIMIT_ORDER_SUCCESS</code> and capture of validation errors.

### **3. Bonus Features and Advanced Strategies**

The following features were implemented to exceed the core requirements and showcase advanced capabilities:

#### **3.1 TWAP (Time-Weighted Average Price) Strategy**

Implemented in `src/advanced/twap.py`, this strategy demonstrates an ability to execute complex algorithms.

- **Mechanism:** Divides a Total Quantity into equal chunks and places sequential **Market Orders** separated by precise time intervals using `time.sleep()`.
- **Benefit:** Minimizes the market impact typically caused by a single large order.

#### **3.2 OCO (One-Cancels-the-Other) Logic**

Implemented in `src/advanced/oco.py`, this fulfills a critical risk management need for Futures trading.

- **Mechanism:** Since native OCO is unavailable on Futures, the bot places two simultaneous conditional orders (`TAKE_PROFIT_MARKET` and `STOP_MARKET`) that both use the `reduceOnly=True` flag to ensure one cancels the other upon position closure.

### 3.3 Lightweight Streamlit Frontend (app.py)

The Streamlit UI provides a production-grade interface for the bot.

UI Component	Functionality
Order Tabs	Dedicated input forms for Market, Limit, OCO, and TWAP execution.
Live PNL Tracking	Displays the <b>Total Unrealized PNL</b> across all open positions using an st.metric with dynamic color coding (green/red).
Live Position Monitoring	Shows the active size, entry price, liquidation price, and individual PNL for all open positions in real-time.
Position Exit	Dedicated tab to safely close any selected open position using a targeted Market Order with reduceOnly=True.

## To run the ui:

streamlit run app.py

The screenshot shows a Streamlit application titled "Binance Futures Testnet Bot". The top navigation bar has a "Deploy" button and three dots. The main area is divided into four sections: "Market Snapshot", "Account", "Open Orders", and "Live Positions & PNL".

- Market Snapshot:** Shows "Symbol: BTCUSDT" and "BTCUSDT Price: 113002.50".
- Account:** Shows "Refresh Balances" and "USDT Balance: 4986.47".
- Open Orders:** Shows "Refresh Orders" and "Open orders: 0".
- Live Positions & PNL:** Shows "Refresh Positions" and "Total Unrealized PNL: \$0.00".

At the bottom, there is a "Market Order" section with tabs for "Market", "Limit", "OCO (Conditional)", "TWAP", and "Exit Position". The "Market" tab is selected. Below the tabs, there is a "Market Order" heading and a form for placing a market order. The form fields include "Symbol: BTCUSDT", "Side: BUY", "Quantity: 0.000001", and a checked "Show validation preview" checkbox.

## Marker Order:

The screenshot shows a "Market Order" form. At the top, there is a horizontal navigation bar with tabs: "Market", "Limit", "OCO (Conditional)", "TWAP", and "Exit Position". The "Market" tab is highlighted with a red underline.

The main form area has the following fields:

- Symbol:** BTCUSDT
- Side:** BUY (radio button selected)
- Quantity:** 0.000001
- Show validation preview:**

At the bottom of the form is a "Place Market Order" button.

## Limit Order:

Limit Order

Symbol: BTCUSDT

Side: BUY

Quantity: 0.000001

Price: 0.00

Auto adjust price/qty to tick/lot step

Place Limit Order

## OCO:

Market Limit OCO (Conditional) TWAP Exit Position Deploy :

OCO (Take-Profit & Stop-Loss)

This places two conditional orders that attempt to close an existing position (reduceOnly).

Symbol: BTCUSDT

Closing Side (SELL if you are LONG): SELL

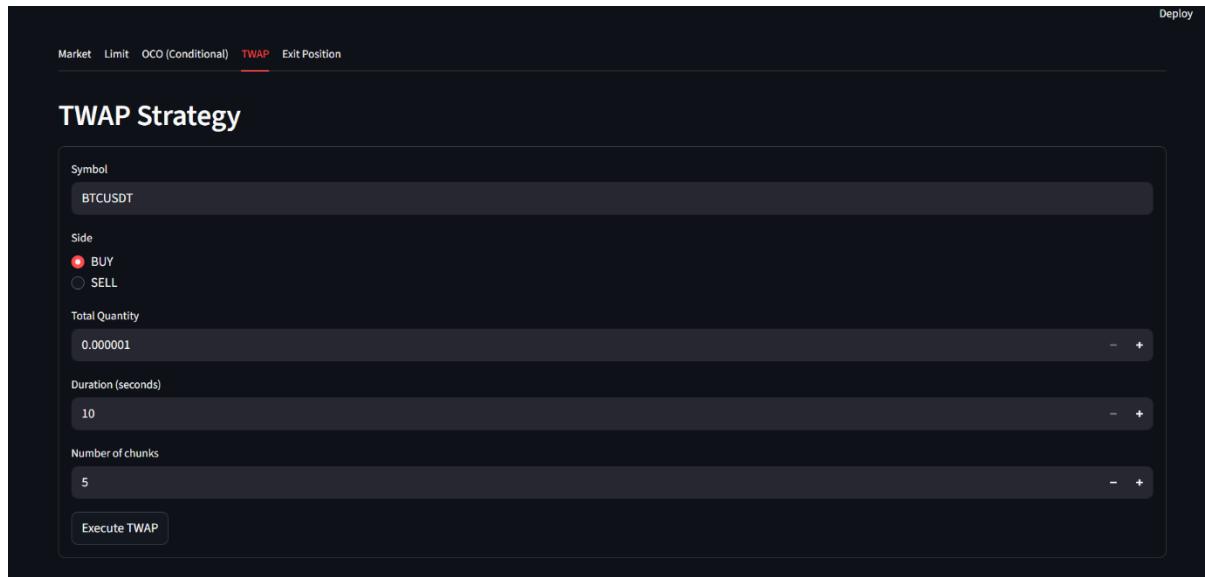
Quantity: 0.000001

Take Profit Trigger Price: 0.00

Stop Loss Trigger Price: 0.00

Place OCO (TP & SL)

## TWAP:

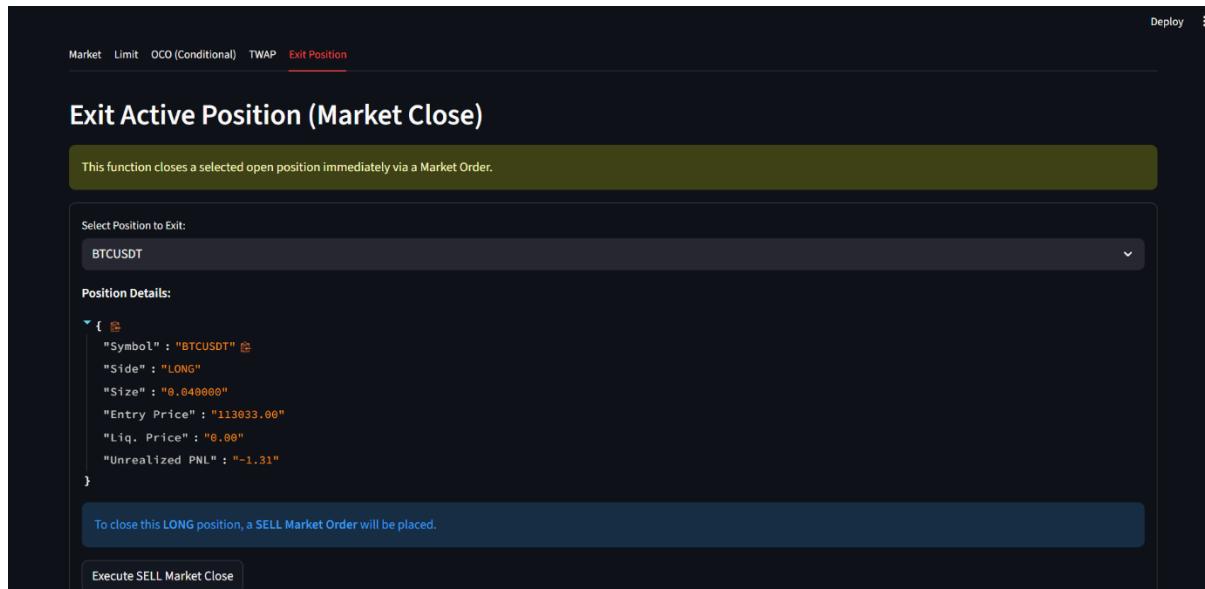


The screenshot shows a dark-themed user interface for a TWAP strategy. At the top, there are tabs: Market, Limit, OCO (Conditional), TWAP (which is selected and highlighted in red), and Exit Position. In the top right corner, there is a "Deploy" button. Below the tabs, the title "TWAP Strategy" is displayed. The form contains the following fields:

- Symbol: BTCUSDT
- Side: BUY (radio button selected)
- Total Quantity: 0.000001
- Duration (seconds): 10
- Number of chunks: 5

A large "Execute TWAP" button is located at the bottom of the form.

## Exit Active Position:



The screenshot shows a dark-themed user interface for exiting an active position. At the top, there are tabs: Market, Limit, OCO (Conditional), TWAP, and Exit Position (which is selected and highlighted in red). In the top right corner, there is a "Deploy" button. Below the tabs, the title "Exit Active Position (Market Close)" is displayed. A green banner below the title states: "This function closes a selected open position immediately via a Market Order." The form contains the following fields:

- Select Position to Exit: BTCUSDT
- Position Details: (A JSON object is shown, indicating a long position of 0.040000 units at an entry price of 113033.00)

```
{
  "Symbol": "BTCUSDT",
  "Side": "LONG",
  "Size": "0.040000",
  "Entry Price": "113033.00",
  "Liq. Price": "0.00",
  "Unrealized PNL": "-1.31"
}
```
- To close this LONG position, a SELL Market Order will be placed.
- Execute SELL Market Close

# LIVE POSITION AND PNL:

The screenshot shows the Binance Futures Testnet Bot interface. It includes sections for Market Snapshot (Symbol: BTCUSDT, Price: 113000.40), Account (USDT Balance: 4984.66), Open Orders (0), and Live Positions & PNL (Total Unrealized PNL: -\$1.72). A table details the active position for BTCUSDT.

Symbol	Side	Size	Entry Price	Liq. Price	Unrealized
BTCUSDT	LONG	0.040000	113033.00	0.00	-1.72

## ⚙️ 2. Execution Commands (CLI Mode)

These commands run the individual Python files directly from the command line. Run these from your project root (the directory containing the src/ folder).

File	Order Type	Format	Example (Run this)
market_orders.py	<b>Market</b> (Mandatory)	python src/market_orders.py <SYMBOL> <SIDE> <QTY>	python src/market_orders.py BTCUSDT BUY 0.001
limit_orders.py	<b>Limit</b> (Mandatory)	python src/limit_orders.py <SYMBOL> <SIDE> <QTY> <PRICE>	python src/limit_orders.py ETHUSDT SELL 0.05 3500.00
advanced/oco.py	<b>OCO-like</b> (Bonus)	python src/advanced/oco.py <SYMBOL> <SIDE> <QTY> <TP_PRICE> <SL_PRICE>	python src/advanced/oco.py BNBUSDT SELL 1.0 250.0 240.0
advanced/twap.py	<b>TWAP</b> (Bonus)	python src/advanced/twap.py <SYMBOL> <SIDE> <TOTAL_QTY> <DURATION_SECONDS>	python src/advanced/twap.py BTCUSDT BUY 0.005 60

**OCO:** The SIDE argument must be the side needed to **close** an existing position (e.g., SELL if you are currently **LONG**).



### 3. Execution Command (Streamlit UI Mode)

This command runs your comprehensive web dashboard, enabling graphical interaction with all orders and displaying the live PNL and positions.

Command	Purpose
<code>streamlit run app.py</code>	Launches the Streamlit server and opens the full application frontend in your web browser.

## 5. Conclusion

The project is complete and fully validated. By integrating robust core functions with advanced strategies and presenting them via an intuitive Streamlit application, the solution demonstrates proficiency in **Python development, financial API interaction, algorithmic logic, and modern application presentation**.

