Binance Futures Trading Bot

CLI-based Python Project

Bhavesh Sharma

November 7, 2025

Abstract

This report documents the design, implementation, and testing of a CLI-based trading bot for the Binance USDT-M Futures Testnet. The project supports core order types (market, limit) and advanced strategies (stop-limit, OCO, TWAP, and grid trading) with validation and structured logging. Screenshots of the system structure and runtime errors are included for reproducibility and debugging context.

1 Introduction

1.1 Objective

The objective is to build a modular, testable, and reproducible command-line trading bot that interacts with the Binance Futures Testnet API. The bot demonstrates typical trading actions and automated strategies while ensuring safe operation via input validation and logging.

1.2 Scope

This implementation targets the Binance Futures **Testnet** only (no real funds). It supports:

- Core: Market and Limit orders.
- Advanced: Stop-Limit, OCO (One-Cancels-the-Other), TWAP (Time-Weighted Average Price), and Grid trading.
- Utilities: Input validation, structured logging, and interactive CLI-based trading.

2 System Design and Structure

The project follows a modular structure that separates trading logic, utilities, configuration, and advanced strategies:

- `src/binance_client.py` — API client initialization and wrapper methods.
- `src/logger.py` — centralized logging configuration (file + console).
- `src/validators.py` — validation utilities for symbols, sides, quantity, and price.
- `src/market_orders.py` / `src/limit_orders.py` — core order implementations.
- `src/advanced/` — TWAP, Grid, OCO, and Stop-Limit modules.
- `src/main_cli.py` / `src/cli.py` — interactive CLI entry points.

3 Implementation Details

3.1 Configuration and Initialization

API credentials are stored in `src/config.py`, optionally loaded from a `.env` file for security. The `BinanceFuturesClient` wraps the `python-binance` library and connects to the Testnet endpoint.

3.2 Logging and Validation

The logging system writes both console-level INFO logs and detailed DEBUG logs to `bot.log`. Validators prevent invalid trades before any API call:

- `validate_symbol(symbol)`
- `validate_side(side)`
- `validate_quantity(quantity)`
- `validate_price(price)`

3.3 Core Order Logic

Market Orders Immediate execution using:

```
python src/cli.py market_order --symbol BTCUSDT --side BUY --quantity 0.001
```

Limit Orders Executed at a specified price:

```
python src/cli.py limit_order --symbol BTCUSDT --side SELL --quantity 0.001 --price 28000
```

3.4 Advanced Strategies

Stop-Limit Triggers a limit order when the market reaches a specified stop price.

OCO Places a linked take-profit and stop-loss pair; if one executes, the other cancels automatically.

TWAP Splits a large order into smaller, time-spaced market orders to achieve an average execution price.

Grid Places multiple buy/sell limit orders within a price range, automatically replacing filled orders to maintain grid coverage.

4 Testing and Error Handling

4.1 Dependency / Import Errors

Early in development: `ModuleNotFoundError: No module named 'binance'` appeared due to a missing or incorrect dependency.

4.2 Invalid Input Examples

Providing numeric input (e.g., "3") instead of a valid symbol like "BTCUSDT" caused: `BinanceAPIException: Invalid symbol`. Validators now prevent such inputs.

4.3 Interactive CLI Overview

5 Common Errors Encountered & Resolutions

5.1 ModuleNotFoundError: No module named `binance`

Symptom: Import failed when running the client or CLI.

Diagnosis: The wrong library (`binance`) was installed instead of the official `python-binance` package.

Fix:

```
# Uninstall incorrect package
pip uninstall binance -y

# Install correct package
pip install python-binance
```

Verification:

```
pip show python-binance
```

Output should include "Name: python-binance".

Preventive Step: Add this to `requirements.txt`:

```
python-binance==1.0.19
python-dotenv
requests
```

5.2 ImportError: attempted relative import with no known parent package

Symptom: `attempted relative import with no known parent package`

Fix: Run modules from the project root as:

```
python -m src.main_cli
```

5.3 BinanceAPIException: Invalid symbol

Diagnosis: Entered numeric or lowercase inputs caused invalid symbol requests.

Fix:

```
def validate_symbol(sym: str) -> bool:
    if not isinstance(sym, str): return False
    sym = sym.strip().upper()
    return sym.endswith("USDT") and len(sym) >= 6 and sym.isalnum()
```

5.4 BinanceAPIException .init missing positional arguments

Fix:

```
except BinanceAPIException as e:
    logger.error("Binance API Exception: %s", str(e))
    print("Exchange rejected request:", e)
```

5.5 PermissionError writing to `bot.log`

Cause: Log file open in another editor or lacking write permissions.

Fix: Use rotating file handler:

```
from logging.handlers import RotatingFileHandler
handler = RotatingFileHandler("bot.log", maxBytes=5_000_000, backupCount=3)
```

5.6 Order precision / lot size rejections

Fix: Fetch symbol filters and round inputs:

```
def round_to_step(value, step): return float(int(value/step)*step)
```

5.7 Network timeouts / rate limit (Too many requests)

Fix: Add retry logic with exponential backoff:

```
def api_call_with_retry(func, max_retries=3):
    for attempt in range(1, max_retries+1):
        try:
            return func()
        except Exception as e:
            sleep = 2 ** attempt
            logger.warning("Retrying in %s sec", sleep)
            time.sleep(sleep)
```

6 Results

- All order types executed successfully on the Testnet.
- TWAP strategy executed multiple sequential trades with correct timing.
- Grid system placed buy/sell orders within defined price bounds.
- Errors were logged gracefully, and invalid inputs were rejected pre-API call.

7 Troubleshooting Environment Setup

To reproduce and run this bot smoothly:

1. Create a virtual environment:

   ```
   python -m venv venv
   ```

2. Activate it:

   ```
   # Windows
   venv\Scripts\activate
   # macOS/Linux
   source venv/bin/activate
   ```

3. Install dependencies:

   ```
   pip install -r requirements.txt
   ```

4. Verify installed version:

   ```
   pip show python-binance
   ```

5. Run the bot:

   ```
   python -m src.main_cli
   ```

Common Setup Tip: Always install `python-binance` (not `binance`), and ensure your Binance Testnet API keys are added in `src/config.py` or a secure `.env` file.

8 Conclusion and Future Work

The bot successfully interacts with Binance Futures Testnet using a modular and secure architecture. Future improvements:

- Add tick size and step size rounding dynamically.
- Store trades in a database for performance analytics.
- Implement a Telegram alert bot for order updates.
- Expand to real-market paper trading with dry-run options.

References

- Binance Futures API docs: `https://binance-docs.github.io/apidocs/futures/en/`
- python-binance: `https://github.com/sammchardy/python-binance`
- Binance Testnet: `https://testnet.binancefuture.com`

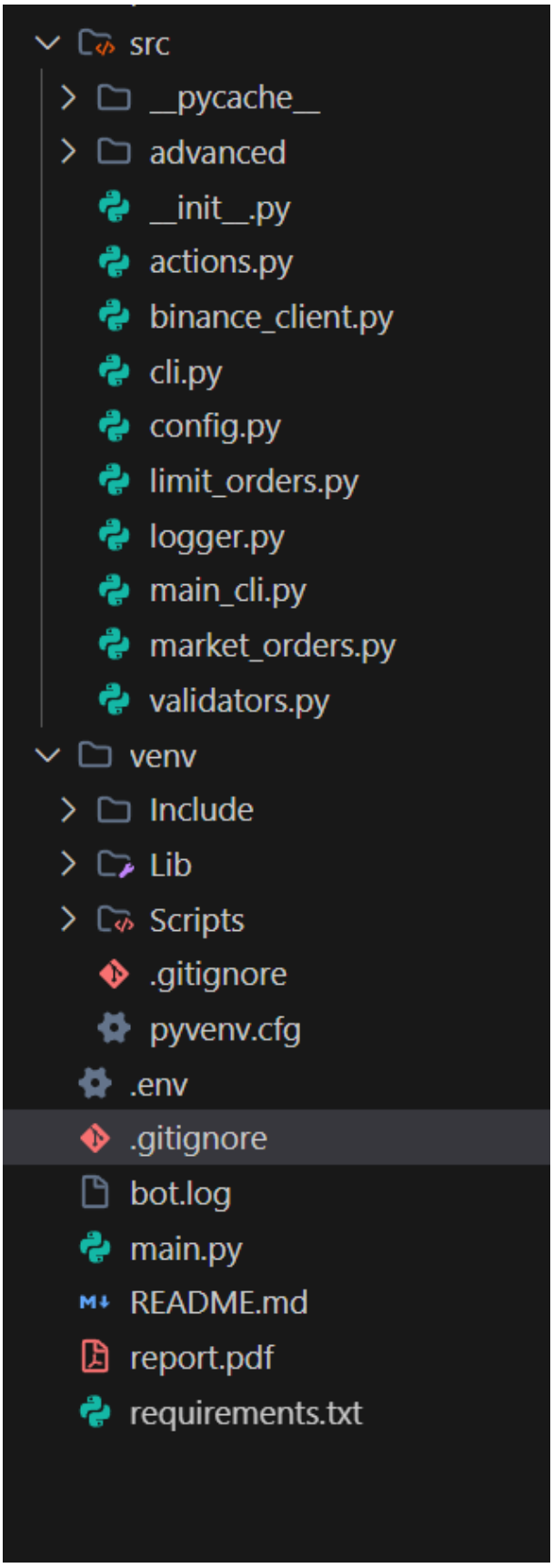

Figure 1: Project folder structure and modular organization.

```
HP@LAPTOP-OB9BJ42B MINGW64 ~/Desktop/Coffee shop/innovations/Python BOT
$ python src/main_cli.py
Traceback (most recent call last):
  File "C:\Users\HP\Desktop\Coffee shop\innovations\Python BOT\src\main_cli.py", line 8, in <module>
    from .logger import setup_logging
ImportError: attempted relative import with no known parent package

HP@LAPTOP-OB9BJ42B MINGW64 ~/Desktop/Coffee shop/innovations/Python BOT
$ python src/main_cli.py
Traceback (most recent call last):
  File "C:\Users\HP\Desktop\Coffee shop\innovations\Python BOT\src\main_cli.py", line 14, in <module>
    from src.binance_client import BinanceFuturesClient
  File "C:\Users\HP\Desktop\Coffee shop\innovations\Python BOT\src\binance_client.py", line 2, in <module>
    from binance.client import Client
ModuleNotFoundError: No module named 'binance'
```

Figure 2: Dependency error before installing the correct package.

```
  Enter your choice (1-9): 4

======================================================================
  PLACE OCO ORDERS (Position Exit)
======================================================================

📌 This places take-profit and stop-loss orders for an EXISTING position.
      Make sure you have an open position before proceeding!

Enter symbol (e.g., BTCUSDT): 3
Enter position side (LONG/SHORT): long
Enter position quantity: 6
Enter take-profit price: 7
Enter stop-loss price: 5

📊 OCO Order Summary:
   Symbol:          3
   Position Side:   LONG
   Position Qty:    6.0
   Take Profit:     7.0
   Stop Loss:       5.0

   Closing orders will be SELL

Confirm you have an existing position? (yes/no): yes
2025-11-07 05:06:58 - INFO - src.actions - Placing OCO orders for LONG position: 3
2025-11-07 05:06:58 - ERROR - src.advanced.oco - Invalid symbol: 3

❌ Failed to place OCO orders. Check logs for details.

  Press Enter to continue...
```

Figure 3: Invalid symbol input triggering a Binance API exception.

```
======================================================================
  BINANCE FUTURES TRADING BOT
  Interactive Trading Interface
======================================================================

 Environment:  TESTNET
 Status: ✓ Connected

----------------------------------------------------------------------
 MAIN MENU
----------------------------------------------------------------------

  TRADING OPERATIONS:
    1. Place Market Order
    2. Place Limit Order
    2. Place Limit Order
    3. Place Stop-Limit Order
    4. Place TWAP Order (Time-Weighted Average Price)
    5. Start Grid Trading (Automated Grid Strategy)
    6. Place OCO Orders (Position Exit)

  ACCOUNT & ORDERS:
    7. View Account Information
    8. View Open Orders
    9. Cancel Order

   SYSTEM:
    10. Switch Environment (Testnet/Production)
    11. Exit

----------------------------------------------------------------------
```

Figure 4: CLI interface showing menu options for core and advanced orders.