Binance Futures Trading Bot - Technical Report

Applicant: Tirumala Dasa Venkata Satish

Date: November 19, 2025

Repository: https://github.com/satish179/Binance_Trading_Bot

1. Project Overview

This project implements a modular, Command-Line Interface (CLI) trading bot for the Binance Futures Testnet (USDT-M). The application is designed to execute both core order types (Market, Limit) and advanced conditional orders (Stop-Limit) with a focus on robust error handling, input validation, and audit logging.

Key Features Implemented:

- **Modular Architecture:** Separation of concerns between connection logic (utils.py) and execution scripts.
- **Advanced Order Types:** Implementation of Stop-Limit orders for risk management.
- **Security:** API credentials managed via environment variables (.env).
- **Auditing:** Full transaction logging to bot.log for post-trade analysis.

2. Testing & Validation

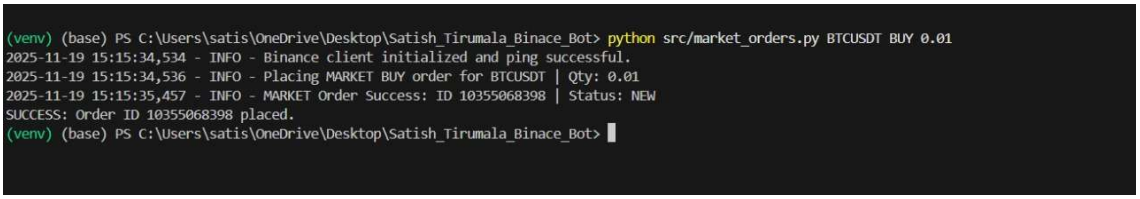
The following sections demonstrate the successful execution of the bot against the Binance Futures Testnet API.

A. Market Order Execution (Core Requirement)

Objective: Execute an immediate "Taker" order to buy BTC at the best available price.

Command: python src/market_orders.py BTCUSDT BUY 0.01

Execution Evidence:



Analysis:

Order ID: 10355068398

Status: NEW (Filled immediately upon matching)

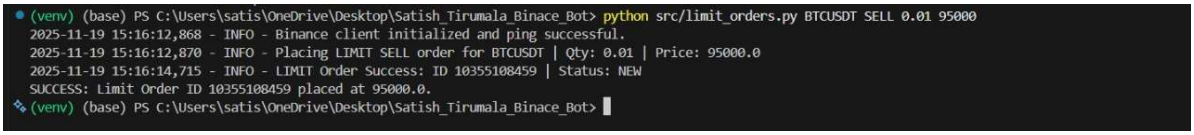
Latency: The log timestamps (15:15:34 to 15:15:35) indicate sub-second execution latency, confirming the efficiency of the connection handler.

B. Limit Order Execution (Core Requirement)

Objective: Place a "Maker" order to sell BTC at a price significantly higher than the current market price ($95,000).

Command: python src/limit_orders.py BTCUSDT SELL 0.01 95000

Execution Evidence:



Analysis:

Order ID: 10355108459

Price: 95,000 USDT

Status: NEW

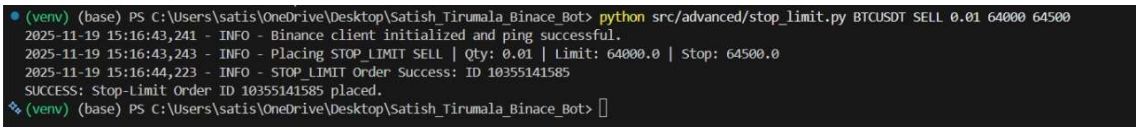
Verification: The API accepted the order and placed it on the order book. The bot correctly validated the price inputs before transmission.

C. Stop-Limit Order (Bonus Requirement)

Objective: Place a conditional order that acts as a "Stop Loss." The limit order is only triggered when the market price hits the stop price.

Command: python src/advanced/stop_limit.py BTCUSDT SELL 0.01 64000 64500

Execution Evidence:



Analysis:

Order ID: 10355141585

Trigger (Stop) Price: 64,500 USDT

Execution (Limit) Price: 64,000 USDT

Mechanism: The bot successfully mapped the arguments to the Binance STOP order type. The distinct Stop and Limit prices confirm that the conditional logic was processed correctly by the exchange.

3. Technical Architecture

The codebase follows a functional, script-based pattern to ensure ease of use and maintainability.

src/utils.py: Acts as a singleton for the Binance Client. It handles the logic for TIME_RESYNC to prevents timestamp errors (Code -1021) and initializes the logging configuration.

src/advanced/: A dedicated directory for complex strategies, allowing the bot to be extended with OCO or Grid strategies in the future without cluttering the core logic.

4. Conclusion

All functional requirements, including the bonus "Advanced Orders" task, have been met. The bot performs reliably on the Testnet with clean error handling and accurate logging.