Binance USDT-M Futures Order Bot

Author: Supriya-Binance-Bot
Date: 9-8-25

1. Overview

This project implements a Command-Line Interface (CLI) bot for Binance USDT-M Futures trading. It supports both mandatory and advanced order types, with input validation, error handling, and structured JSON logging. The bot can operate in two modes: real trading with API keys or safe dry-run mode for simulation.

The goal of this bot is to give the user flexibility and safety when placing trades, as well as to demonstrate understanding of the Binance Futures API, Python CLI development, and logging pr

2. Architecture

The project is organized as follows:

- src/cli.py: Main entry point; parses arguments and calls respective order functions.
- src/logger.py: Structured JSON logger that writes to both console and bot.log.
- src/binance_client.py: Wrapper for Binance Futures API with dry-run fallback.
- src/market_orders.py and src/limit_orders.py: Core mandatory order logic.
- src/advanced/: Contains stop_limit.py, oco.py, twap.py, and grid.py for advanced features.
- README.md: Setup and usage guide.
- bot.log: Runtime log file in JSON format.

3. Usage

Example dry-run commands:
python src/cli.py --dry-run market BTCUSDT BUY 0.001
python src/cli.py --dry-run limit BTCUSDT SELL 0.001 56000

Run with API keys for live trading (caution advised):
python src/cli.py --api-key <KEY> --api-secret <SECRET> market BTCUSDT BUY 0.001

4. Challenges & Future Work

Challenges:
- Ensuring correct argument parsing for both global and subcommand options.
- Handling API unavailability or rate limits gracefully.

Future Work:
- Add native OCO support if Binance Futures API provides it in future.
- Integrate optional market indicators (e.g., Fear & Greed Index).
- Enhance grid strategy to run continuously and manage open orders automatically.

5. Screenshots

[Place your dry-run or testnet run screenshots here before submission.]