REPORT ON CLI BASED PYTHON BOT

WHAT I USED?

Libraries

1. **logging** – For maintaining logs of actions, orders, and errors.
2. **binance** – Official Python wrapper for Binance API to place orders and fetch account/trade information.
3. **datetime** – For handling timestamps and synchronizing with Binance server time.
4. **dotenv** – For loading API keys from a .env file, keeping sensitive information secure and outside the code.

1.

WORKFLOW & IMPLEMENTATION

The bot was designed as a **Command Line Interface (CLI)** application where the user can select order types, enter order details, and execute trades on **Binance Futures Testnet**.

Core Steps:

1. Load API keys securely from .env file.
2. Initialize Binance API client with testnet enabled.
3. Get user inputs for trade type, symbol, side, and quantity.
4. Place orders using Binance Futures API functions.
5. Log all actions in a bot.log file for debugging and record-keeping.

FEATURES IMPLEMENTED

1. Market Order

Ease of Implementation: Easiest to implement.

Functionality: Executes immediately at the best available market price.

Logging: Successful orders and errors are recorded in the log file.

---

2. Limit Order

Difficulty Level: Moderate – required understanding how to set price and timeInForce.

Functionality: Places an order at a specific price, which is executed only if the market reaches that price.

Logging: Every order attempt is logged.

Key Parameters Used:

price – Limit price at which the trade is executed.

timeInForce – “GTC” (Good Till Cancelled) to keep the order open until it’s filled or manually cancelled.

3. Stop Limit Order

Difficulty Level: Higher – needed understanding of stopPrice vs price and trigger behavior.

Functionality: Combines a stop trigger and limit order. The order is activated only when the stop price is reached.

Logging: Both trigger and limit execution logs recorded.

Key Parameters Used:

stopPrice – Price at which the stop order is activated.

price – Limit price for the order after activation.

CHALLENGES FACED

Late Start – Project began two days after receiving the assignment due to PC repairs.

Understanding Binance Testnet – Needed time to grasp Futures account setup and API usage.

Order Type Logic –

Market order was straightforward.

Limit order required more study to fully understand execution conditions.

Stop limit order was initially confusing due to differences between stopPrice and price, but I was able to implement it successfully.

Environment Setup – Ensuring .env keys were correctly loaded without hardcoding sensitive data.

LEARNINGS & TAKEAWAYS

Learned how to **securely store API keys** using .env and dotenv.

Gained hands-on experience with **Binance Futures API**.

Understood the differences between **market, limit, and stop limit orders**.

Improved knowledge of **logging and error handling** in Python projects.

Realized the importance of **server time synchronization** to prevent recvWindow and timestamp errors.

FUTURE IMPROVEMENTS

Implement **automated OCO (One Cancels the Other) logic** for Futures by combining stop-loss and take-profit orders.

Add **order status tracking** and automatic cancellation of unfilled orders after a timeout.

Integrate **risk management features** such as maximum loss per trade and leverage adjustments.

Improve CLI interface for a more user-friendly experience.

-

CONCLUSION

This project gave me a strong foundation in working with trading APIs, handling order types, and building a CLI-based trading tool. Despite a delayed start, I was able to implement multiple order types, handle logging, and securely manage API credentials. The knowledge gained will be valuable for more complex trading bot development in the future.