📊 Binance Futures Order Bot — Project Report

1. Objective

The goal of this project is to build a CLI-based Binance USDT-M Futures trading bot capable of executing both basic and advanced order types.
It demonstrates real-time interaction with Binance Futures API, includes structured logging, and follows secure coding practices (using .env for API keys).

2. Features Implemented

✅ Core Orders (Mandatory)

Order Type	Description	Example Command
Market Order	Executes immediately at current market price.	python src/market_orders.py BTCUSDT BUY 0.01
Limit Order	Executes when the market reaches a specified price.	python src/limit_orders.py BTCUSDT BUY 0.01 67000

3. Validation & Logging

- **Validation:**
    - Checks symbol format (e.g., BTCUSDT).
    - Verifies positive quantity and price values.
    - Ensures user inputs are within Binance-allowed ranges.
- **Logging:**
  All actions (order placement, errors, and confirmations) are written to bot.log in this format:
- [2025-10-24 14:25:30] INFO: Market order placed for BTCUSDT (BUY 0.01)
- [2025-10-24 14:25:35] ERROR: Invalid price input for limit order

4. Technology Stack

Component	Technology Used
Language	Python 3.10+
API Library	python-binance
Logging	Python logging module
CLI	argparse

5. Sample Outputs

Market Order Execution

Command: python src/market_orders.py BTCUSDT BUY 0.01

Output: ✅ Market order executed successfully!

Limit Order Example

Command: python src/limit_orders.py BTCUSDT BUY 0.01 67000

Output: ⏳ Limit order placed at $67000

OCO Order Example

Command: python src/advanced/oco.py BTCUSDT BUY 0.01 68000 66000

Output: Placed OCO — Take Profit @ 68000, Stop Loss @ 66000

TWAP Strategy Example

Command: python src/advanced/twap.py BTCUSDT BUY 0.05 --interval 10

Output: Executing 5 orders of 0.01 BTC every 10 seconds...

6. Results & Observations

- Market and limit orders executed successfully in test environment.
- TWAP strategy reduced slippage effectively by splitting orders.
- Logging provided full traceability of all actions.