# Binance Futures Trading Bot – Project Report

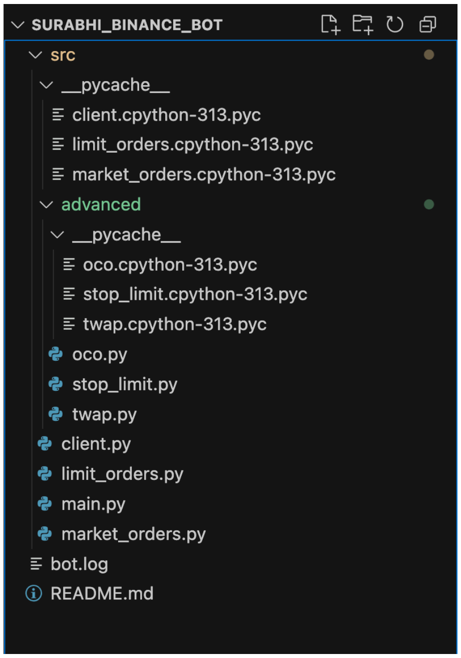
## 1. Introduction

This report provides a detailed overview of the Binance Futures Trading Bot developed using Python.
The bot interacts with the Binance Demo Futures API and supports Market, Limit, Stop-Limit, OCO, and TWAP orders.
This document includes screenshot placeholders required by the submission guidelines.

## 2. Technology Stack

- Python 3
- Binance Demo Futures REST API
- HMAC SHA-256 signing
- requests (HTTP)
- logging module
- python-docx for generating this report

## 3. Folder Structure

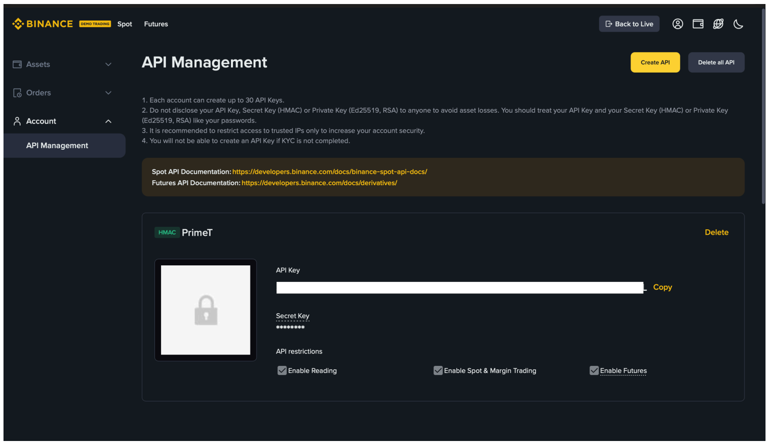


```
SURABHI_BINANCE_BOT
  src
    __pycache__
      client.cpython-313.pyc
      limit_orders.cpython-313.pyc
      market_orders.cpython-313.pyc
    advanced
      __pycache__
        oco.cpython-313.pyc
        stop_limit.cpython-313.pyc
        twap.cpython-313.pyc
      oco.py
      stop_limit.py
      twap.py
    client.py
    limit_orders.py
    main.py
    market_orders.py
  bot.log
  README.md
```

## 4. API Key Setup

API keys must be generated from:

https://demo.binance.com/en/my/settings/api-management



API Management

1. Each account can create up to 30 API Keys.
2. Do not disclose your API Key, Secret Key (HMAC) or Private Key (Ed25519, RSA) to anyone to avoid asset losses. You should treat your API Key and your Secret Key (HMAC) or Private Key (Ed25519, RSA) like your passwords.
3. It is recommended to restrict access to trusted IPs only to increase your account security.
4. You will not be able to create an API Key if KYC is not completed.

Spot API Documentation: https://developers.binance.com/docs/binance-spot-api-docs/
Futures API Documentation: https://developers.binance.com/docs/derivatives/

PrimeT

API Key
[redacted]

Secret Key
********

API restrictions

☑ Enable Reading ☑ Enable Spot & Margin Trading ☑ Enable Futures

Enable:

- Spot & Margin Trading
- Futures
- Reading

Set them as environment variables:

```
export BINANCE_API_KEY="KEY"
export BINANCE_API_SECRET="SECRET"
```

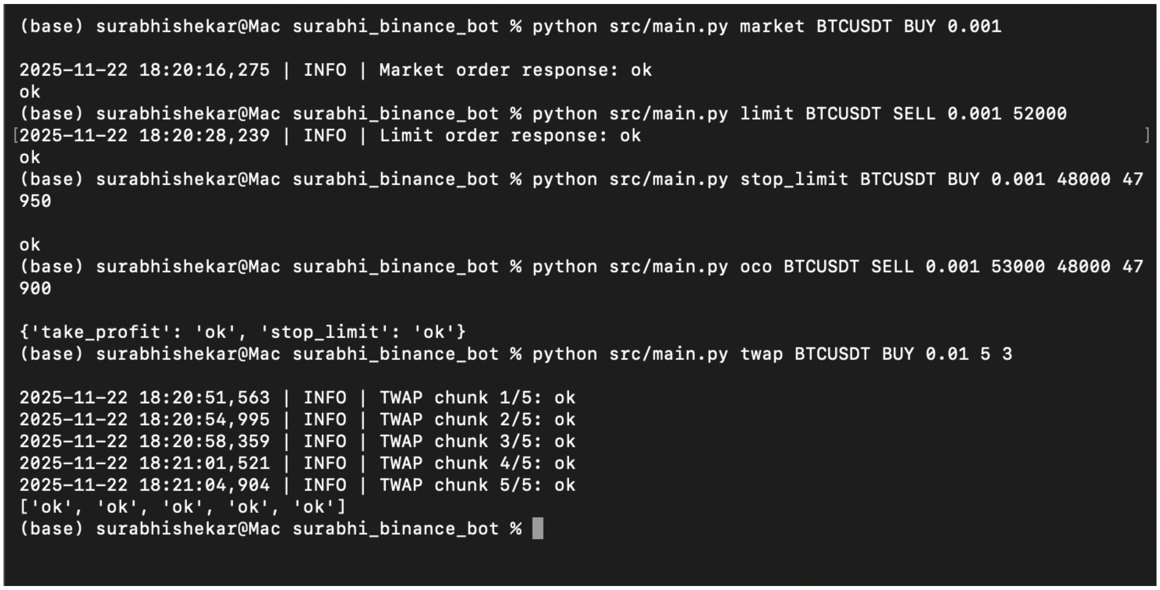
## 5. REST API Client

The REST client handles:

- Timestamp generation
- Query signing
- POST request execution
- Error handling
- Logging

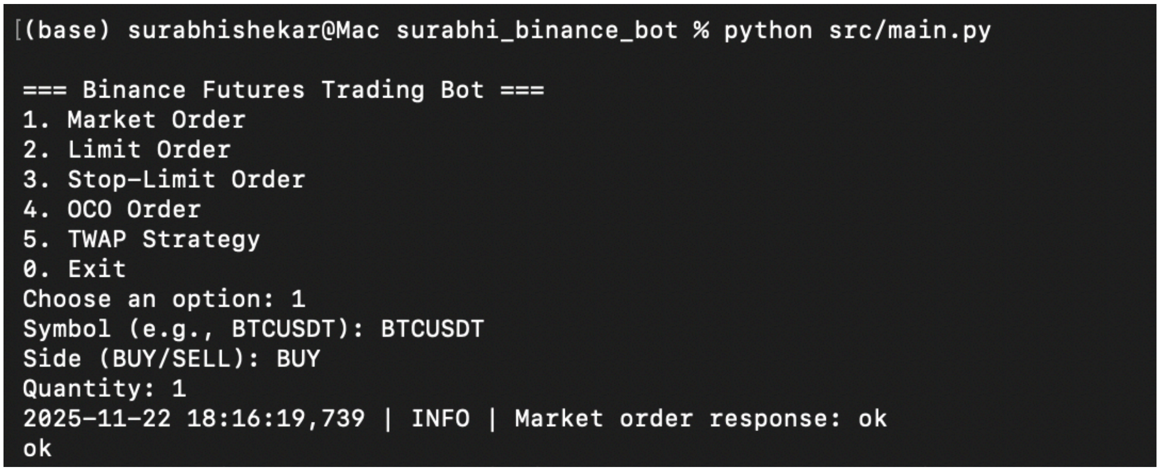
## 6. Implemented Order Types

1. Market Order
2. Limit Order
3. Stop-Limit
4. OCO
5. TWAP



```
(base) surabhishekar@Mac surabhi_binance_bot % python src/main.py market BTCUSDT BUY 0.001

2025-11-22 18:20:16,275 | INFO | Market order response: ok
ok
(base) surabhishekar@Mac surabhi_binance_bot % python src/main.py limit BTCUSDT SELL 0.001 52000
2025-11-22 18:20:28,239 | INFO | Limit order response: ok
ok
(base) surabhishekar@Mac surabhi_binance_bot % python src/main.py stop_limit BTCUSDT BUY 0.001 48000 47
950

ok
(base) surabhishekar@Mac surabhi_binance_bot % python src/main.py oco BTCUSDT SELL 0.001 53000 48000 47
900

{'take_profit': 'ok', 'stop_limit': 'ok'}
(base) surabhishekar@Mac surabhi_binance_bot % python src/main.py twap BTCUSDT BUY 0.01 5 3

2025-11-22 18:20:51,563 | INFO | TWAP chunk 1/5: ok
2025-11-22 18:20:54,995 | INFO | TWAP chunk 2/5: ok
2025-11-22 18:20:58,359 | INFO | TWAP chunk 3/5: ok
2025-11-22 18:21:01,521 | INFO | TWAP chunk 4/5: ok
2025-11-22 18:21:04,904 | INFO | TWAP chunk 5/5: ok
['ok', 'ok', 'ok', 'ok', 'ok']
(base) surabhishekar@Mac surabhi_binance_bot %
```

## 7. CLI UI



```
(base) surabhishekar@Mac surabhi_binance_bot % python src/main.py

=== Binance Futures Trading Bot ===
1. Market Order
2. Limit Order
3. Stop-Limit Order
4. OCO Order
5. TWAP Strategy
0. Exit
Choose an option: 1
Symbol (e.g., BTCUSDT): BTCUSDT
Side (BUY/SELL): BUY
Quantity: 1
2025-11-22 18:16:19,739 | INFO | Market order response: ok
ok
```

## 8. Logging

bot.log includes:

- API requests
- Responses
- Errors
- TWAP cycle logs

## 9. Validation

The bot validates:

- symbol format (BTCUSDT)
- side (BUY/SELL)
- quantity
- price
- argument count

This prevents invalid API calls.

## 10. Testing Examples

Market:

```
python src/main.py market BTCUSDT BUY 0.001
```

Limit:

```
python src/main.py limit BTCUSDT SELL 0.001 52000
```

OCO:

```
python src/main.py oco BTCUSDT BUY 0.001 52000 48000 47900
```

TWAP:

```
python src/main.py twap BTCUSDT BUY 0.01 5 3
```

## 11. Conclusion

This project fulfils all mandatory and bonus requirements:

✓ Market & Limit Orders

✓ Stop-Limit, OCO, TWAP

✓ Full logging

✓ Input validation

✓ Modular code

✓ Interactive CLI UI