Binance Futures Trading Bot — Assignment Report

Candidate Name: Sharayu Bodkhe
Role Applied: Full Stack Developer Intern — Trading Automation
Date: 24 Nov 2025

1. Introduction

This project implements a Python-based automated trading bot for the Binance USDT-M Futures Testnet. The bot supports multiple order types, automated strategies, error handling, and logging. The bot interacts only with the Binance Testnet, meaning no real money is used.

2. Objectives

- Implement a trading bot using the Binance Testnet API
- Support Market and Limit orders
- Add advanced order handling including Stop-Limit, OCO, TWAP, and Grid trading
- Ensure reliable error handling, input validation, and logging
- Provide a user-friendly CLI interface

3. Features Implemented

Feature	Status
Market Orders	✔ Implemented
Limit Orders	✔ Implemented

Stop-Limit Orders	✔ Implemented
OCO (One Cancels the Other)	✔ Implemented
TWAP Strategy (time-based algorithmic execution)	✔ Implemented
Grid Strategy (automated multi-level limit orders)	✔ Implemented
Input Validation	✔ Implemented
Logging	✔ Implemented
View Open Orders	✔ Implemented

All assignment requirements and advanced options were completed successfully.

4. Technology Stack

- **Python**
- **python-binance API**
- **dotenv for credentials**
- **Logging module**
- **Command Line Interface (CLI)**

5. System Architecture

The bot follows a modular and scalable structure:

```
src/
 ├── core.py
 ├── orders/
 |      ├── market_order.py
|       ├── limit_order.py
|       └── stop_limit_order.py
└── advanced/
        ├── oco_order.py
        ├── twap_order.py
        └── grid_order.py
```

This structure allows new strategies and features to be added easily without modifying core logic.

6. Execution Flow

1. User runs `python src/core.py`
2. Bot loads `.env` credentials
3. User selects order type from menu
4. Inputs are validated
5. Order request is sent to Binance Testnet
6. Result is printed and logged in `bot.log`

7. Testing Summary

To test functionality, symbols such as `BTCUSDT` and `ETHUSDT` were used.

Test outcomes included:

- Successful order placements
- API validation messages (minimum size, margin, trigger rules)
- Orders confirmed on Binance Testnet dashboard

These responses confirmed correct interaction with Binance API.

8. Future Improvements

Potential enhancements:

- Frontend dashboard (Flask or React)
- Live price streaming using WebSockets
- Advanced risk management (Stop Loss automation)
- Automated backtesting & trade analytics

9. Conclusion

This project fulfills all core and advanced technical assignment requirements.
It demonstrates strong understanding of:

- Automated trading logic
- Binance Testnet API usage
- Secure credential handling
- Modular software design principles