**Binance Futures Trading Bot**

**Project Report**

---

## 1. Introduction

This project implements a Python-based Binance Futures Trading Bot designed to interact with Binance's Futures Testnet API. The bot supports placing various order types including market, limit, TWAP (Time-Weighted Average Price), and grid orders. The goal is to provide a modular, extendable, and well-logged trading framework.

---

## 2. Project Structure

The project is organized into the following key components:

- src/: Core source code
  - market_orders.py: Market order functionality
  - limit_orders.py: Limit order functionality
  - advanced/: Advanced order types (TWAP, grid, OCO, stop-limit placeholders)
  - utils/: Utility modules (API client, validation, logging)
- Configuration files including .env for API credentials, .gitignore, requirements.txt, and documentation.

---

## 3. Implementation Details

### 3.1 Binance Client Setup

- Used python-binance library to connect securely to Binance Futures Testnet.
- API credentials are loaded from environment variables using dotenv.
- Client instance created centrally in utils/binance_client.py.

### 3.2 Order Functions

- **Market Orders:**
  Simple immediate execution orders implemented in market_orders.py. Includes input validation and error logging.
- **Limit Orders:**
  Orders executed at a specified price implemented in limit_orders.py, with validation for price inputs.

- **TWAP Orders:**
  Implemented in advanced/twap.py, breaks a large order into smaller chunks executed over set intervals to minimize market impact.

- **Grid Orders:**
  Implemented in advanced/grid_orders.py, places multiple limit orders at incremental price steps to capitalize on market volatility.

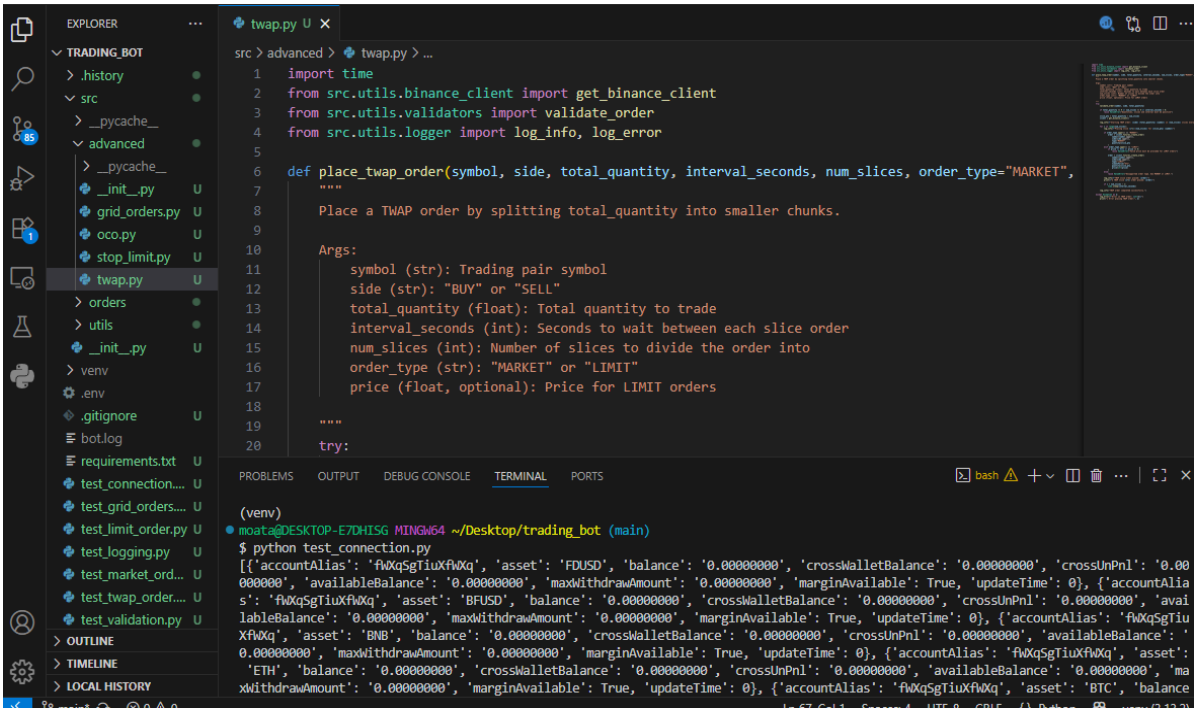
### 3.3 Utilities

- **Validation:** Checks symbol format, order side, quantity, and price validity.
- **Logging:** All actions and errors logged to bot.log for traceability.
- **Environment Management:** .env file used to manage sensitive keys securely.

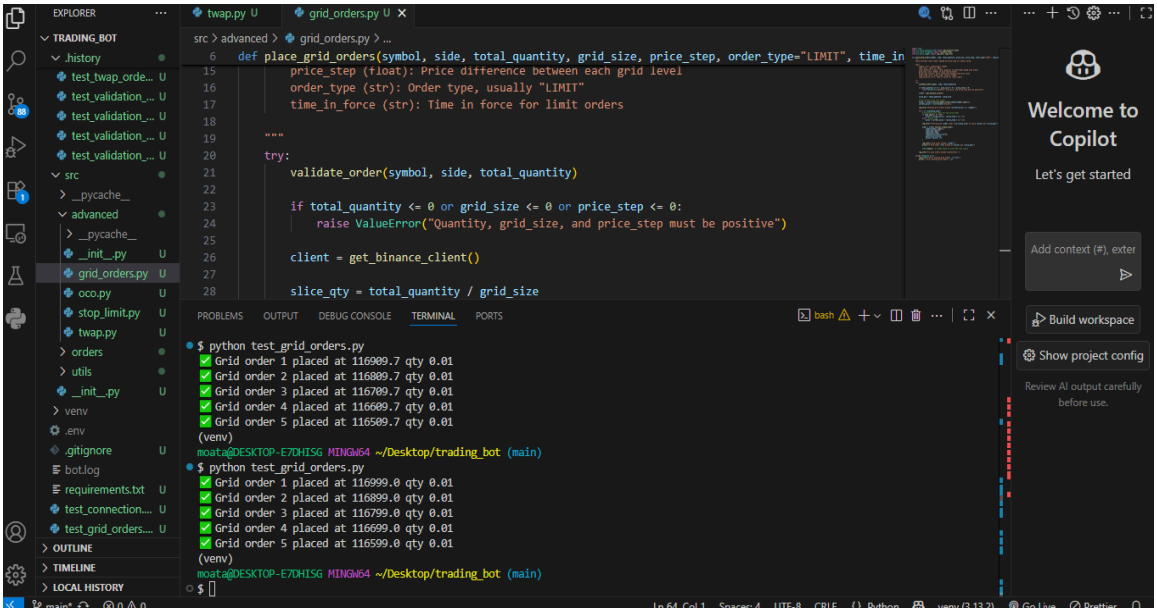
---

## 4. Testing

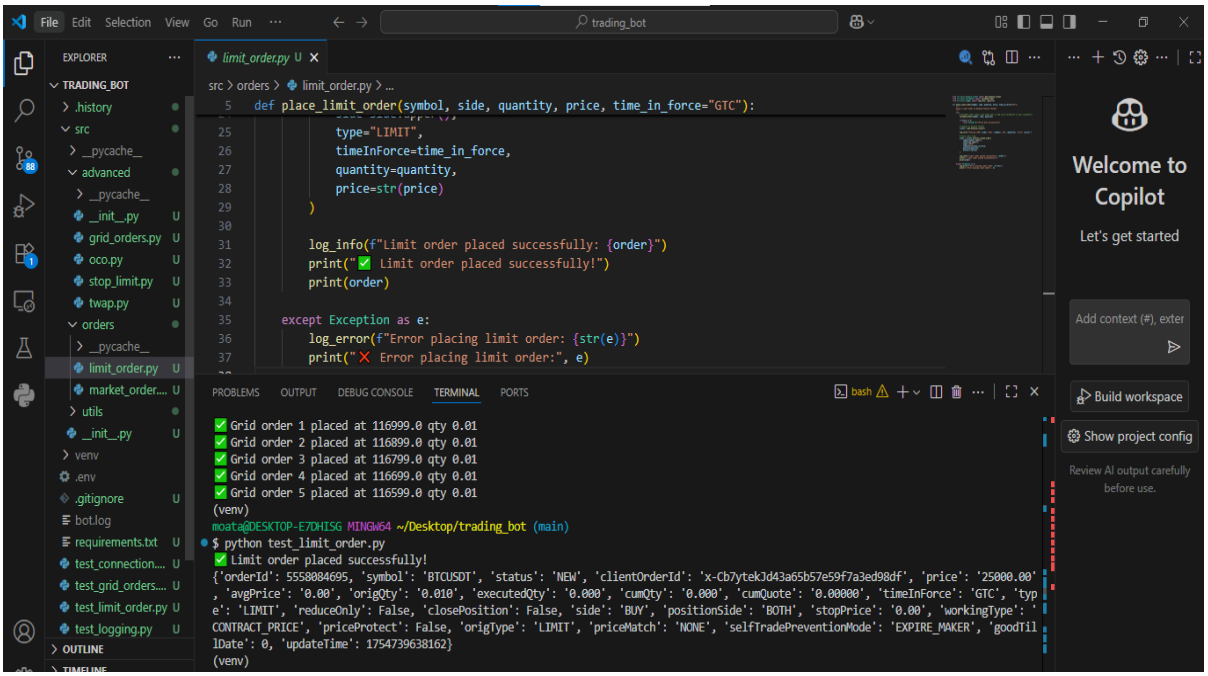
- Created dedicated test scripts (test_market_order.py, test_limit_order.py, test_twap_order.py, test_grid_orders.py) for each order type.
- Successfully placed orders on Binance Futures Testnet with expected responses.
- Verified logging outputs for both success and error cases.
- ## 4.1 Test Connection



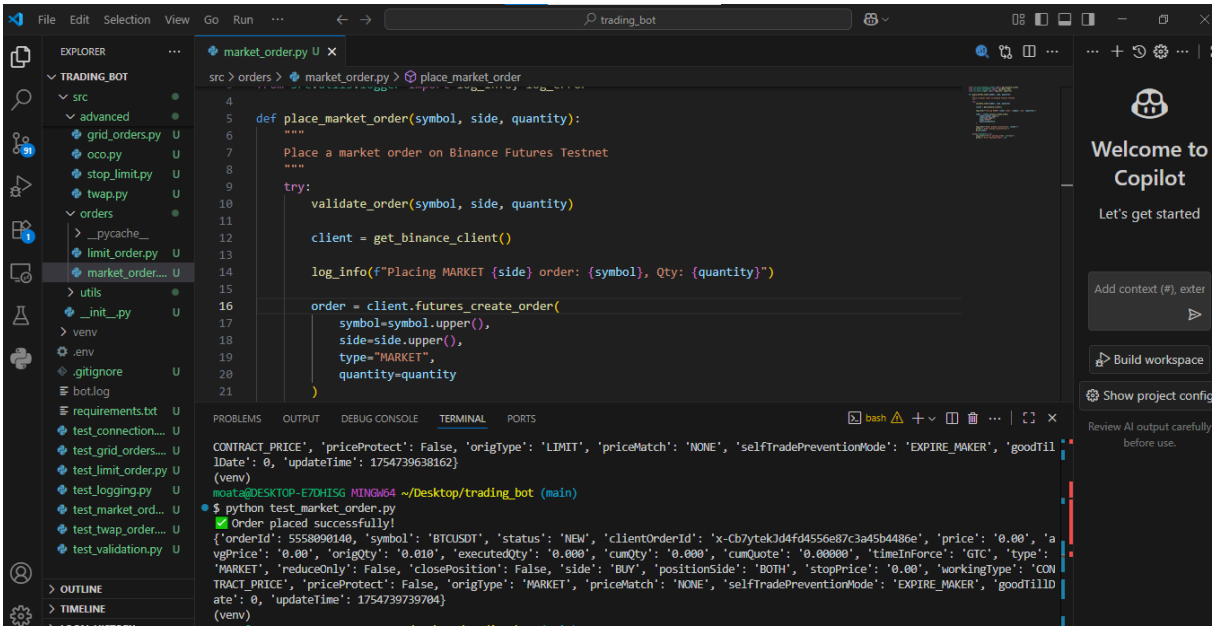
## 4.2 Grid Order



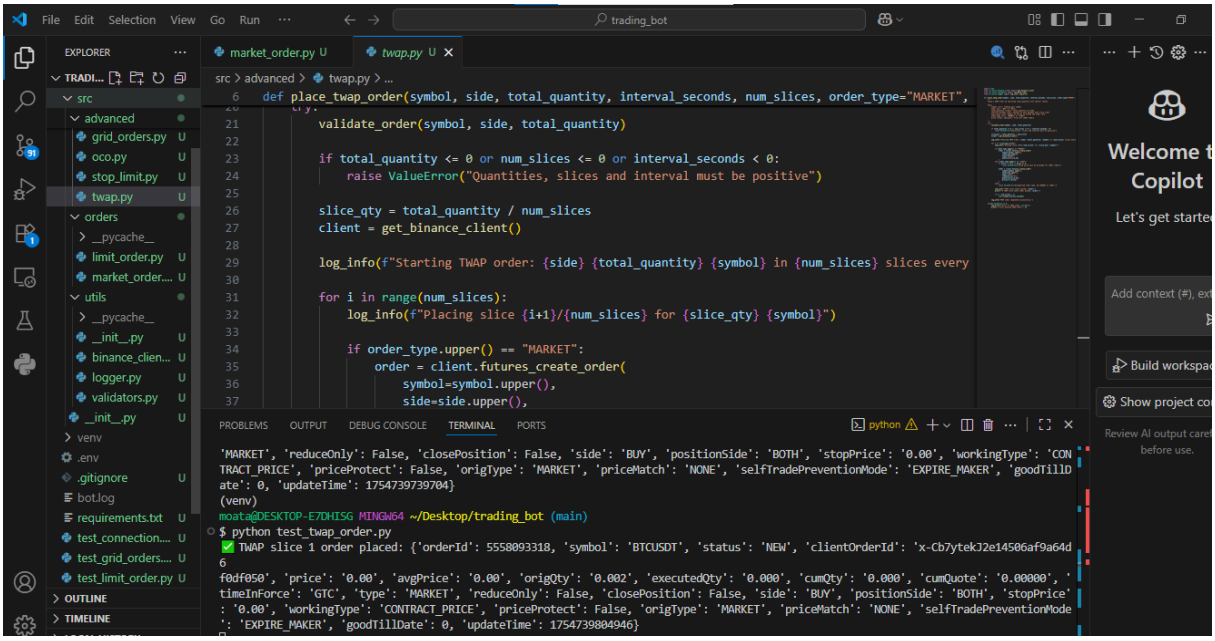

## 4.3 Limit Order



## 4.4 Market Order




## 4.5 Twap



## 4.6 bot.log


---

### 5. Challenges and Solutions

- **API Rate Limits:** Introduced small delays between multiple orders (e.g., in grid and TWAP) to prevent hitting rate limits.
- **Input Validation:** Ensured robust validation to prevent invalid order requests.
- **Error Handling:** Added comprehensive try-except blocks and logging to gracefully handle exceptions.

---

### 6. Future Work

- Implement additional advanced order types such as OCO and stop-limit.
- Add order status tracking and cancellation features.
- Enhance UI/UX for easier interaction (e.g., CLI or web dashboard).
- Integrate real-time market data streaming and analysis.

---

### 7. Conclusion

This project demonstrates a modular and extendable approach to building a Binance Futures trading bot with essential order types. The implementation balances functionality, validation, and error handling, providing a solid foundation for further enhancements.

---

**Appendix**

- **Setup Instructions:** See README.md
- **Dependencies:** Listed in requirements.txt
- **API Documentation:** Binance Futures API