Used Project Structure

```
[project_root]/
│
├── /src/                    # All source code
│   ├── __init__.py
│   ├── main.py              # Main CLI interface
│   ├── config.py            # Configuration management
│   ├── logger.py            # Structured logging
│   ├── validator.py         # Input validation
│   ├── binance_client.py    # Binance API client wrapper
│   ├── market_orders.py     # Market order logic
│   ├── limit_orders.py      # Limit order logic
│   │
│   └── /advanced/           # Advanced order types
│       ├── __init__.py
│       ├── stop_limit.py    # Stop-limit order logic
│       ├── oco.py           # OCO order logic
│       ├── twap.py          # TWAP strategy
│       └── grid_strategy.py # Grid trading strategy
│
├── bot.log                  # Logs (API calls, errors, executions)
├── requirements.txt        # Python dependencies
├── .gitignore              # Git ignore rules
├── env_example.txt         # Environment variables example
└── README.md               # This file
```

Setup Instructions

1. **Install dependencies**

```
pip install -r requirements.txt
```

2. **Set up API credentials**

```
# Edit config.py and add your credentials
BINANCE_API_KEY=your_api_key_here
BINANCE_API_SECRET=your_api_secret_here
BINANCE_USE_TESTNET=true  # Use testnet for testing
```

Usage/Examples

The bot provides a unified CLI interface through `src/main.py`:

```
python -m src.main <command> [arguments]
```

market Orders

Execute an order at the current market price:

```
python -m src.main market <SYMBOL> <SIDE> <QUANTITY> [--reduce-only]

# Examples:
python -m src.main market BTCUSDT BUY 0.01
python -m src.main market ETHUSDT SELL 0.1 --reduce-only
```

Limit Order

Place an order at a specific price:

```
python -m src.main limit <SYMBOL> <SIDE> <QUANTITY> <PRICE> [--time-in-
force GTC|IOC|FOK] [--reduce-only]

# Examples:
python -m src.main limit BTCUSDT BUY 0.01 50000
python -m src.main limit ETHUSDT SELL 0.1 3000 --time-in-force IOC
```

Stop-Limit Order

Trigger a limit order when stop price is reached:

```
python -m src.main stop-limit <SYMBOL> <SIDE> <QUANTITY> <STOP_PRICE>
<LIMIT_PRICE> [--time-in-force] [--reduce-only]

# Examples:
# BUY: Triggers when price rises to or above stop_price
python -m src.main stop-limit BTCUSDT BUY 0.01 51000 50900

# SELL: Triggers when price falls to or below stop_price
python -m src.main stop-limit BTCUSDT SELL 0.01 49000 49100
```

OCO (One-Cancels-the-Other) Order

Place take-profit and stop-loss orders simultaneously:

```
python -m src.main oco <SYMBOL> <SIDE> <QUANTITY> <TAKE_PROFIT_PRICE>
<STOP_LOSS_PRICE> [--stop-limit-price]
```

```
# Examples:
python -m src.main oco BTCUSDT BUY 0.01 52000 48000
python -m src.main oco ETHUSDT SELL 0.1 3200 2800
```

TWAP (Time-Weighted Average Price) Order

Split large orders into smaller chunks over time:

```
python -m src.main twap <SYMBOL> <SIDE> <TOTAL_QUANTITY>
<DURATION_MINUTES> [--num-slices]

# Examples:
# Execute 0.1 BTC over 60 minutes, split into 10 slices
python -m src.main twap BTCUSDT BUY 0.1 60

# Execute 1 ETH over 30 minutes, split into 20 slices
python -m src.main twap ETHUSDT SELL 1.0 30 --num-slices 20
```

Grid Trading Strategy

Create automated buy-low/sell-high orders within a price range:

```
# Create a grid
python -m src.main grid create <SYMBOL> <LOWER_PRICE> <UPPER_PRICE>
<NUM_GRIDS> <QUANTITY_PER_GRID> [--side BOTH|BUY|SELL]

# Check grid status
python -m src.main grid status <SYMBOL>

# Examples:
# Create a grid with 10 levels between 48000 and 52000
python -m src.main grid create BTCUSDT 48000 52000 10 0.01

# Create only BUY orders
python -m src.main grid create BTCUSDT 48000 52000 10 0.01 --side BUY

# Check status
python -m src.main grid status BTCUSDT
```

Direct Module Execution

```
# Market orders
python -m src.market_orders BTCUSDT BUY 0.01

# Limit orders
```

```
python -m src.limit_orders BTCUSDT BUY 0.01 50000

# Stop-limit orders
python -m src.advanced.stop_limit BTCUSDT BUY 0.01 51000 50900

# OCO orders
python -m src.advanced.oco BTCUSDT BUY 0.01 52000 48000

# TWAP orders
python -m src.advanced.twap BTCUSDT BUY 0.1 60

# Grid strategy
python -m src.advanced.grid_strategy BTCUSDT 48000 52000 10 0.01
python -m src.advanced.grid_strategy status BTCUSDT
```

Examples

Example 1: Simple Market Buy

```
python -m src.main market BTCUSDT BUY 0.01
```

Example 2: Limit Order with Take-Profit and Stop-Loss

```
# Place entry order
python -m src.main limit BTCUSDT BUY 0.01 50000

# Place OCO for exit
python -m src.main oco BTCUSDT SELL 0.01 52000 48000
```

Example 3: TWAP Execution

```
# Execute large order over 1 hour
python -m src.main twap BTCUSDT BUY 1.0 60 --num-slices 20
```

Example 4: Grid Trading

```
# Create grid
python -m src.main grid create BTCUSDT 48000 52000 20 0.01

# Check status later
python -m src.main grid status BTCUSDT
```

Run Example Log:

```
2025-11-26 09:39:29,957 - twap - INFO - TWAP slice 1/10
2025-11-26 09:39:29,957 - binance_client - INFO - HTTP GET
/fapi/v1/exchangeInfo
2025-11-26 09:39:30,708 - binance_client - INFO - HTTP response OK
2025-11-26 09:39:30,709 - binance_client - INFO - Placing MARKET order
2025-11-26 09:39:30,709 - binance_client - INFO - HTTP POST /fapi/v1/order
2025-11-26 09:39:31,257 - binance_client - ERROR - API error 401: {'code':
-2014, 'msg': 'API-key format invalid.'}
2025-11-26 09:39:31,257 - main - ERROR - CLI command failed: API error
401: {'code': -2014, 'msg': 'API-key format invalid.'}
2025-11-26 09:39:32,332 - twap - INFO - TWAP slice 1/10
2025-11-26 09:39:32,332 - binance_client - INFO - HTTP GET
/fapi/v1/exchangeInfo
2025-11-26 09:39:33,070 - binance_client - INFO - HTTP response OK
2025-11-26 09:39:33,073 - binance_client - INFO - Placing MARKET order
2025-11-26 09:39:33,073 - binance_client - INFO - HTTP POST /fapi/v1/order
2025-11-26 09:39:33,601 - binance_client - ERROR - API error 401: {'code':
-2014, 'msg': 'API-key format invalid.'}
2025-11-26 09:39:33,601 - main - ERROR - CLI command failed: API error
401: {'code': -2014, 'msg': 'API-key format invalid.'}
2025-11-26 09:39:34,766 - twap - INFO - TWAP slice 1/10
2025-11-26 09:39:34,767 - binance_client - INFO - HTTP GET
/fapi/v1/exchangeInfo
2025-11-26 09:39:35,569 - binance_client - INFO - HTTP response OK
2025-11-26 09:39:35,572 - binance_client - INFO - Placing MARKET order
2025-11-26 09:39:35,572 - binance_client - INFO - HTTP POST /fapi/v1/order
2025-11-26 09:39:36,112 - binance_client - ERROR - API error 401: {'code':
-2014, 'msg': 'API-key format invalid.'}
2025-11-26 09:39:36,112 - main - ERROR - CLI command failed: API error
401: {'code': -2014, 'msg': 'API-key format invalid.'}
```