# 🤖 AI Trading Bot with Binance Testnet + TradingView

This is an automated trading bot powered by AI that places orders on the \*\*Binance Testnet\*\*. It listens to \*\*webhooks from TradingView\*\* and reacts to technical signals like RSI, MACD, EMA, and more.

---

## 📁 Project Structure

ai\_trading\_bot/

├── .env # Environment variables (not committed)

├── .env.example # Template for .env setup

├── requirements.txt # Python dependencies

├── Dockerfile # Docker image config

├── docker-compose.yml # Optional container orchestration

├── webhook\_server/ # Receives TradingView alerts (FastAPI)

├── trading\_bot/ # AI logic and Binance interaction

├── trades.db # SQLite DB (autogenerated)

└── strategy.pine # Pine script for TradingView

## 📁 Project Structure

ai\_trading\_bot/

├── .env # Secrets and keys (ignored in Git)

├── .env.example # Template example for .env

├── requirements.txt # Python dependencies

├── Dockerfile # Container build config

├── docker-compose.yml # (Optional) Docker Compose

├── .dockerignore # Exclude files from Docker image

├── webhook\_server/ # Receives TradingView webhook alerts

│ ├── main.py # FastAPI server + HMAC verification

│ ├── utils.py # HMAC signature logic

│ └── config.py # Load environment variables

├── trading\_bot/ # Core AI + trade logic

│ ├── main.py # Signal handler and execution

│ ├── ai\_model.py # LSTM model and RL stub

│ ├── binance\_client.py # Connects to Binance API (to be filled)

│ ├── logger.py # Logging and SQLite trade tracking

├── tests/ # Unit tests

│ ├── test\_ai\_model.py # Tests for LSTM decision logic

├── trades.db # SQLite database (generated at runtime)

├── trading\_bot.log # Log file (generated at runtime)

└── strategy.pine # Pine Script for TradingView strategy

## 🔐 Setting Up API Keys

### 🔑 1. TradingView Webhook API Key

This is used to verify that incoming webhook requests are valid.

- Go to: [uuidgenerator.net](https://www.uuidgenerator.net/)

- Copy a UUID (e.g., `d4f0c532-3905-449f-b7da-69ee07125da7`)

Example `.env` config:

```env

API\_KEY=tradingview\_webhook\_uk\_bot # Your TradingView API Key Create your own any text

SECRET\_KEY=d4f0c532-3905-449f-b7da-69ee07125da7 # HMAC secret key

🧪 2. Get Binance Testnet API Key

Visit: https://testnet.binance.vision

Log in with your Binance account

Click "Create API Key"

Copy your API Key and Secret

# Your Binance Testnet API credentials

BINANCE\_API\_KEY=your\_testnet\_api\_key

BINANCE\_API\_SECRET=your\_testnet\_api\_secret

USE\_TESTNET=true

⚙️ .env Example (Full Config)

# === Webhook Auth ===

API\_KEY=tradingview\_webhook\_uk\_bot

SECRET\_KEY=d4f0c532-3905-449f-b7da-69ee07125da7

# === Binance Testnet Keys ===

BINANCE\_API\_KEY=3Mq4UvD1ObhsGFVphr9hi4zv5dFxZ6GIWT64G41E0X6aNXwNEnPM0NCHSV3MU8Wq

BINANCE\_API\_SECRET=1cLewWfB6mxy98lda7gDVlt4ytW1n7uppebEjfpCbdZg149EaaEiqY6iPtPFcvXo

USE\_TESTNET=true

# === Binance Testnet Symbols Supported ===

# BNBUSDT, BTCUSDT, ETHUSDT, LTCUSDT, TRXUSDT, XRPUSDT, USDT, BUSD

# === Trade Symbols ===

SYMBOLS=BNBUSDT,BTCUSDT,ETHUSDT

# === Default Trade Settings ===

MAX\_POSITION\_SIZE=0.01

COOLDOWN\_SECONDS=60

# === Per-Symbol Overrides (optional) ===

MAX\_POSITION\_SIZE\_BNB=0.1

MAX\_POSITION\_SIZE\_ETH=0.02

MAX\_POSITION\_SIZE\_BTC=0.005

🖥️ Run Locally (Virtual Environment)

1. Clone the Project

git clone https://github.com/your-username/ai\_trading\_bot.git

cd ai\_trading\_bot

2. Create Virtual Environment

python -m venv venv

# Activate:

# Windows:

venv\Scripts\activate

# macOS/Linux:

source venv/bin/activate

3. Install Dependencies

pip install -r requirements.txt

4. Add Your .env File

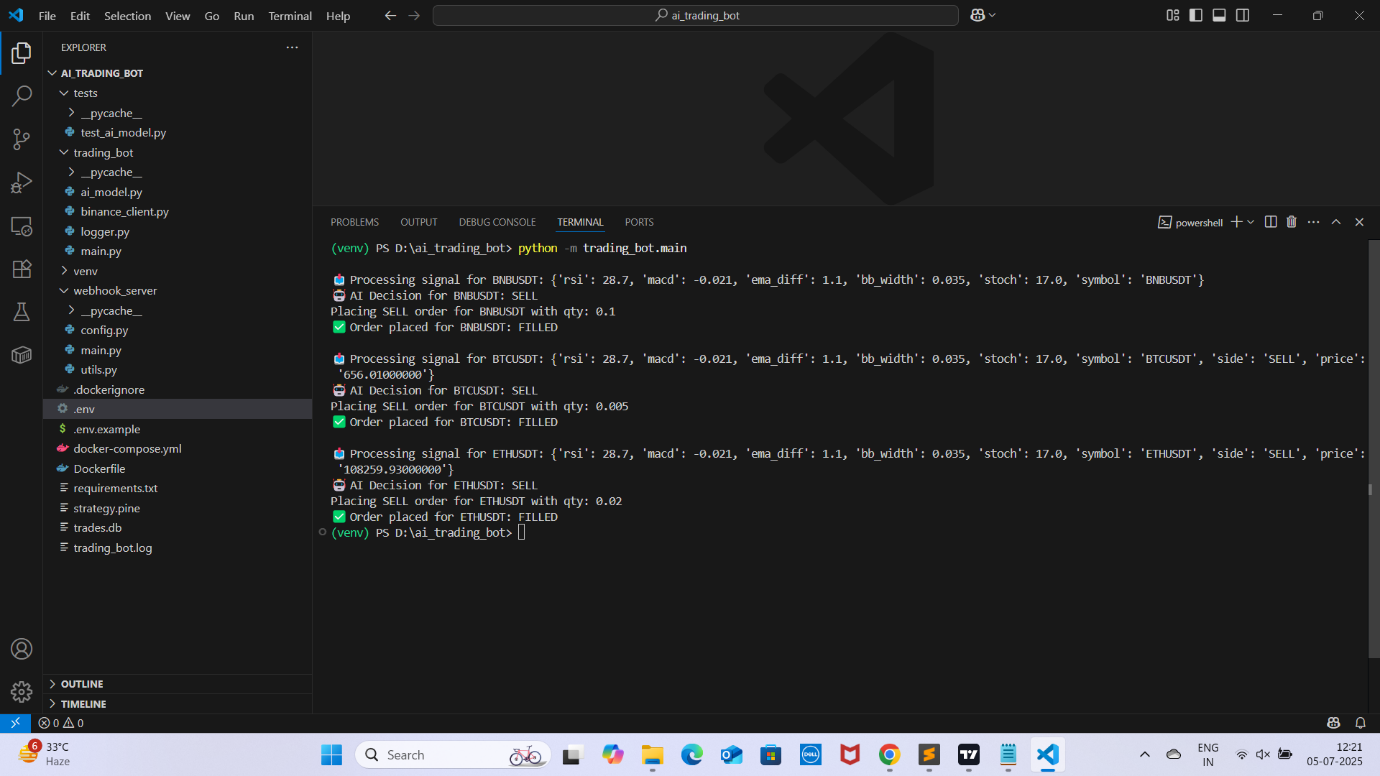
Copy .env.example to .env and fill in values.

5. Run the Webhook Server

uvicorn webhook\_server.main:app --reload --port 8000

6. Test a Signal

python -m trading\_bot.main



🐳 Run on Docker

1. Build & Start

docker-compose up --build

2. Open Web UI

<http://localhost:8000/docs>

3. Send a Webhook Manually

curl -X POST http://localhost:8000/webhook \

-H "Content-Type: application/json" \

-d '{"symbol": "BTCUSDT", "rsi": 30, "macd": -0.1, "side": "buy"}'

🔄 Connect with TradingView

1. Paste Pine Script into TradingView

Use the provided strategy.pine from this repo. Example webhook:

alert('{' +

'"api\_key": "tradingview\_webhook\_uk\_bot",' +

'"symbol": "' + syminfo.ticker + '",' +

'"price": "' + str.tostring(close) + '",' +

'"side": "buy",' +

'"rsi": "' + str.tostring(rsi) + '",' +

'"macd": "' + str.tostring(macdLine) + '"' +

'}', freq=alert.freq\_once\_per\_bar)

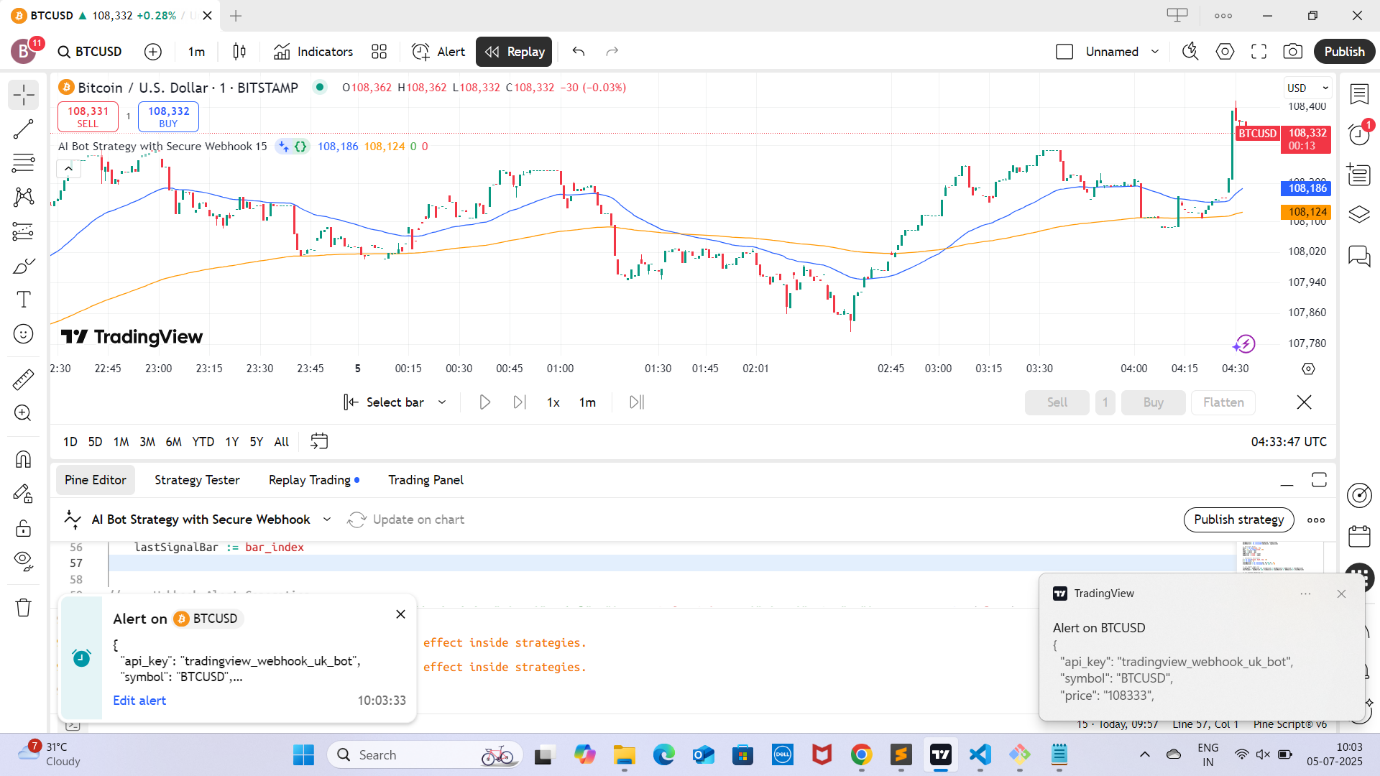
2. Set Webhook URL

http://<your-public-ip>:8000/webhook

If deploying to cloud (e.g., AWS/VPS), ensure port 8000 is open.

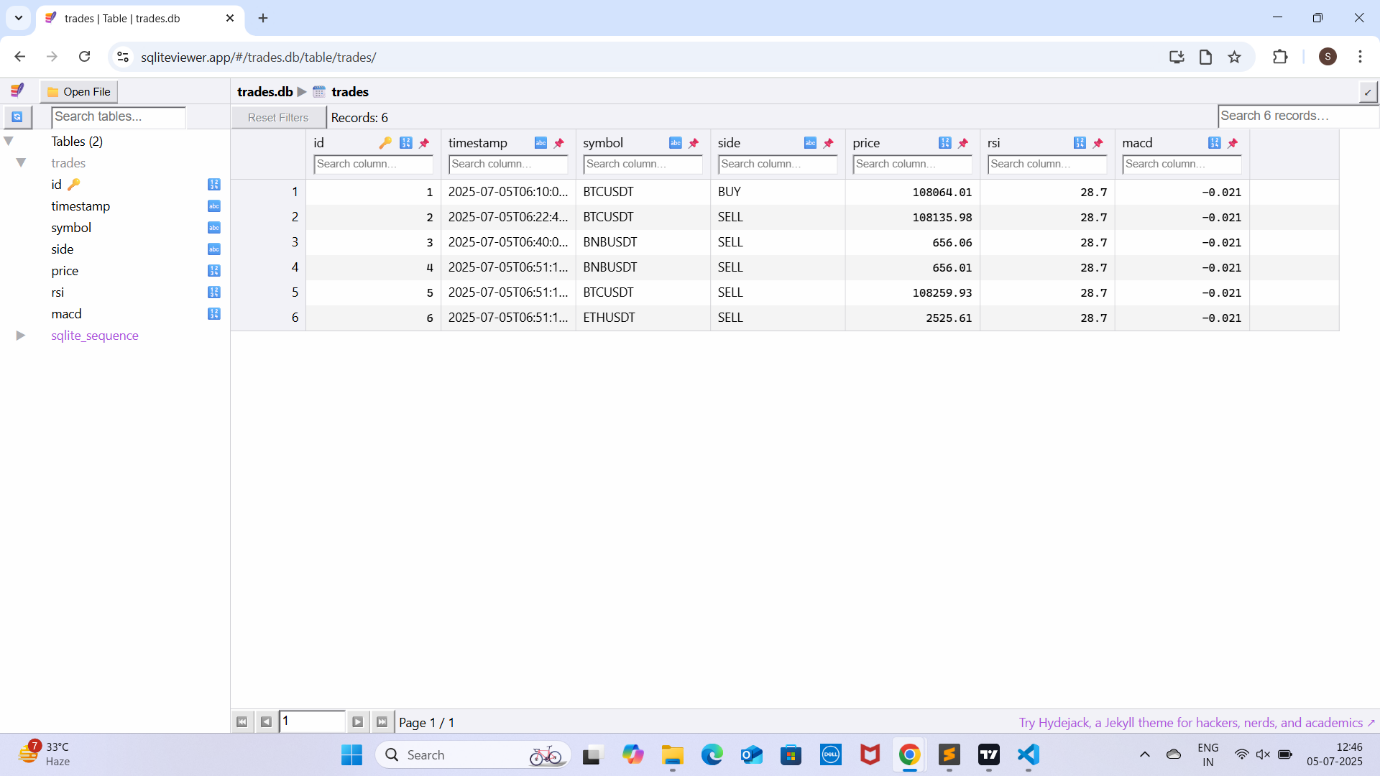
A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.



✅ Features

AI-based signal filtering (LSTM + indicators)

Binance Testnet trading

Per-symbol cooldown and position sizing

FastAPI webhook server

SQLite trade logging

Docker support

🧪 Run Unit Tests

pytest tests/

🔐 Disclaimer

This project is for educational and testing purposes only. Do not use real funds without fully understanding the risks.

📝 License

MIT License

Let me know if you want me to also generate:

- `.env.example`

- Dockerfile

- TradingView-compatible `strategy.pine`

- `docker-compose.yml`

- Or deploy instructions for a VPS like AWS/EC2 or DigitalOcean