Crypto Trading Engine - Simple Documentation

What This Project Does

This is a crypto trading system that helps you:

- Show historical prices for a given time interval for the currency asked for
- Show real time socket data
- Switch off the socket for 10 seconds during live demo, and restart, the missed data should be filled too while tracking the real time data again
- Handle the case of missing data or heterogenous data across exchanges

It uses Binance for market data and MongoDB to store prices. It can recover from connection drops and can be expanded to support more exchanges like Coinbase or Kraken.

Main Parts of the Project

1. `main.py`

This is where the system starts running.

- Found in: `src/main.py`
- **Steps:**
    1. Read any command line settings.
    2. Load configuration from a file if it exists.
    3. Set up everything the system needs.
    4. Start the trading system.
    5. Stop everything safely if user presses Ctrl+C or system shuts down.

2. `price_engine/` folder

This is the core folder for price tracking and trading logic.

a. `data_manager.py`

- **Does:** Handle live and past price data.
- **Steps:**
    1. Start Binance WebSocket for live prices.

    2. Every time a price is received:
        - Save it in MongoDB.
        - Notify any connected tools.
    3. For old data:
        - Try to get it from MongoDB first.
        - If not there, fetch from Binance.

b. `binance_ws.py`

- **Does:** Connect to Binance for live trade prices.
- **Steps:**
    1. Open connection.
    2. When new data arrives:
        - Extract symbol, price, and time.
        - Send it to interested tools.
    3. If connection drops, try reconnecting.

c. `fetch_data.py`

- **Does:** Lets you manually get old price data.
- **Steps:**
    1. Get a list of symbols and times.
    2. For each symbol:
        - Ask for historical data.
        - Show it or save it.

d. `fetch.py`

- **Does:** Helper file used by `fetch_data.py` to reuse logic.
- It is used for finding out missing data or heterogenous data across exchanges

e. `processor.py`

- **Does:** Analyze prices and decide when to buy or sell.
- **Steps:**
    1. Keep a rolling list of past prices.
    2. Every time a new price comes:
        - Calculate indicators like SMA, RSI, MACD, etc.
        - Decide if the signal is BUY, SELL, or HOLD.
        - Notify any connected tools.

f. `risk_management.py`

- **Does:** Makes sure we don't take too much risk.
- **Steps:**
  1. Check your balance and the trade signal.
  2. Calculate safe trade size.
  3. Make sure:
     - We don't risk too much on one trade
     - We don't have too many trades open
     - Stop-loss and take-profit are set
  4. Approve or reject the trade.

g. `order_executor.py`

- **Does:** Places the actual trades.
- **Steps:**
  1. Accepts trade signals.
  2. Sends order to Binance (or simulates if no API keys).
  3. If the order is filled:
     - Log the trade
     - Register stop-loss/take-profit if needed
     - Save it to MongoDB

h. `trading_engine.log`

- Keeps track of everything that happens for debugging.

i. Other folders (`config/`, `data_fetchers/`, etc.)

- Help organize configs, DB files, and WebSocket connections.

3. `utils/`

Extra tools used by the system.

a. `config_loader.py`

- Loads settings from a config file.

b. `logger.py`

- Sets up how logs are shown and saved.

c. `data_fetcher.py`

- Helps with web requests like GET and POST with retry options.

4. `observability/`

- Will be used later to monitor logs and metrics.

5. `api/`, `execution/`, `llm_agents/`, `models/`

- These folders are placeholders for future features.

6. `tests/`

- Has test files.
- Used to check if signal and risk logic works properly.

What This System Can Do

1. Get Historical Prices

- Use `get_price_history(symbol, start_time, end_time)`
- Looks in MongoDB first, else asks Binance.

2. Live Price Streaming

- Uses WebSocket to get live prices.
- Saves each price in MongoDB.

3. Recover Lost Data

- If connection is lost:
    - System finds missing time
    - Fills missing prices from Binance

4. Smart Risk Control

- Calculates how much to trade based on risk.
- Automatically sets stop loss & take profit.

5. Ready for Multiple Exchanges

- Right now supports Binance.
- Easily expandable to Coinbase, Kraken, etc.

Testing the Project

Test 1: Get Historical Prices

python src/price_engine/fetch_data.py

Test 2: Live Price Tracking

python src/main.py

Test 3: Handle Socket Drop

1. Start system
2. Manually disconnect WebSocket
3. Wait 10 seconds
4. Restart socket and check it fills the gap

Test 4: Missing Data from Other Exchanges

- Delete data in MongoDB.
- System should auto-fill from Binance.
- Future: Try Coinbase/Kraken if added.

Folder Structure (Simplified)

```
src/
├── main.py               # Starts everything
├── price_engine/         # Core logic
│   ├── data_manager.py
│   ├── order_executor.py
│   ├── processor.py
│   ├── risk_management.py
│   ├── fetch_data.py
│   ├── main.py
│   ├── config/, db/, etc.
├── utils/                # Helpers
├── tests/                # Test cases
├── api/, models/, etc.   # Future use
```

Github Repository:

https://github.com/shashankmutyala/Automated-Trading-Agent