# How My Simulator Mimics a Real Exchange

## 1. Introduction

A financial exchange is a system that allows buyers and sellers to trade assets by submitting orders. At the core of any modern electronic exchange lies the **Limit Order Book (LOB)**, which stores outstanding buy and sell orders and matches them according to predefined rules. The goal of this project was to build a simplified but functionally correct version of a real exchange's order matching system.

In this simulator, I implemented a working limit order book that supports limit orders, market orders, price–time priority, partial fills, and trade generation. Although the simulator is simplified and does not focus on performance or real-world scale, it closely follows the fundamental principles used by real exchanges.

## 2. Orders and Order Types

In real exchanges, participants interact with the market by submitting **orders**. My simulator models this using an Order object that contains the essential attributes needed for trading:

- Side (buy or sell)
- Price (for limit orders)
- Quantity
- Timestamp

The simulator supports two main types of orders:

### Limit Orders

A limit order specifies the maximum price a buyer is willing to pay or the minimum price a seller is willing to accept. If the order cannot be executed immediately, it is stored in the order book. Limit orders provide liquidity to the market.

### Market Orders

A market order does not specify a price and instead executes immediately against the best available prices in the order book. Market orders consume liquidity and may move the market price.

These two order types capture the core interaction between liquidity providers and liquidity takers in real markets.

## 3. Structure of the Limit Order Book

The order book is divided into two sides:

- **Bid side (buy orders)**
- **Ask side (sell orders)**

Each side is organized by price levels. Buy orders are sorted from highest price to lowest price, while sell orders are sorted from lowest price to highest price. This sorting ensures that the best available prices are always accessed first.

At each price level, orders are stored in a FIFO (first-in-first-out) queue. This reflects **time priority**, meaning that among orders with the same price, the one that arrived earlier is executed first. This price–time priority is the same rule used by real-world exchanges.

## 4. Matching Engine and Trade Execution

The matching engine is the core component of the simulator. Whenever a new order arrives, the order book checks whether it can be matched against existing orders on the opposite side.

For a **buy order**, matching occurs when the best ask price is less than or equal to the buy price.

For a **sell order**, matching occurs when the best bid price is greater than or equal to the sell price.

Trades are executed using the following logic:

- Orders match at the best available price
- The traded quantity is the minimum of the incoming order quantity and the resting order quantity

• Partial fills are allowed
• Any remaining quantity is either matched further or stored back in the book

Each successful match generates a **trade record**, which contains the trade price, quantity, and aggressor side. This process closely mimics how trades are executed in real electronic exchanges.

# 5. Liquidity, Spread, and Market Behaviour

The simulator naturally exhibits important market concepts:

• **Liquidity**: Provided by limit orders resting in the book
• **Market impact**: Market orders consume liquidity and may move prices
• **Bid–ask spread**: Formed by the difference between the best bid and best ask

As orders arrive and trades occur, the spread can widen or narrow depending on available liquidity. This behavior reflects real market dynamics, even though the simulator operates at a simplified level.

# 6. Simplifications and Assumptions

To keep the system simple and focused on core concepts, several real-world features were intentionally omitted:

• No order cancellations or modifications
• No hidden orders or advanced order types
• No latency, networking, or concurrency
• No real market data feed

Despite these simplifications, the simulator remains conceptually accurate. The core mechanisms of order submission, matching, and execution are the same as those used by real exchanges.

# 7. Conclusion

This simulator demonstrates how a real exchange operates at its core. By implementing a limit order book with price–time priority, limit and market orders, and a matching engine, the project captures the essential behavior of electronic markets. While simplified, the design is modular and extensible, making it suitable for future extensions such as multiple trading agents, visualisation tools, or reinforcement learning environments.

Overall, this project helped solidify my understanding of market microstructure and the role of the limit order book in price discovery and liquidity formation.