LASTENHEFT / PFLICHTENHEFT

Abschlussprojekt an der HBFSWI Saarbrücken

Thema: Orderbuch-Simulation
Name: Moritz Kolb
Klasse: WI24ZI
Abschlussjahr: 2026

Inhaltsverzeichnis

1. Einführung

In jedem Finanzmarkt auf der Welt wird ein sogenanntes Orderbuch verwendet. Dieses Orderbuch listet die Kauf- und Verkaufsangebote der Marktteilnehmer auf und verrechnet diese miteinander. Zu jedem Zeitpunkt befindet sich das Orderbuch auf dem aktuellen Stand der Interessen von Käufern und Verkäufern.

Jede Order besteht aus einer bestimmten Menge des gehandelten Wertes sowie aus einem festen Preis. Bei Verkäufern beschreibt dieser Preis den minimalen Preis, zu dem sie bereit sind zu verkaufen, während es sich bei Käufern um den maximalen Preis handelt, den sie bereit sind zu zahlen. Die im Orderbuch enthaltenen Ein- und Verkaufsanweisungen werden systematisch miteinander verrechnet. Die besten Angebote bestimmen dabei den Preis, zu dem das jeweilige Gut aktuell gehandelt wird.

Eine Software, welche diesen Prozess simuliert, ist besonders interessant für Lehrtätigkeiten sowie für Forschungs- und Testzwecke. Mit einer solchen Simulation lassen sich die zugrundeliegenden Prozesse der Preisentstehung besser verstehen, analysieren und erklären. Da es sich um ein internes Projekt handelt, wird eine fiktive, aber realitätsnahe Projektsituation angenommen.

2. Ist-Situation

Eine Bank, die die Geldanlagen ihrer Kunden möglichst effektiv anlegen möchte, nutzt eine Software, welche verschiedene Finanzmärkte auf Investitionsmöglichkeiten untersucht. Dabei investiert die Bank bevorzugt in Märkte, deren aktueller Preis über dem Durchschnittspreis der letzten 50 Tage liegt.

Um sicherzustellen, dass diese Software auch im späteren Live-Betrieb zuverlässig und fehlerfrei funktioniert, müssen umfangreiche Unit-Tests durchgeführt werden. Diese Tests benötigen eine große Menge an Marktdaten, um möglichst viele unterschiedliche Testszenarien abdecken zu können.

Reale digitale Finanzmarktdaten sind jedoch in ihrer Anzahl begrenzt und häufig kostenpflichtig. Außerdem lassen sie sich nur eingeschränkt anpassen. Aus diesem Grund besteht die Notwendigkeit, künstliche Preisdaten zu erzeugen. Eine Software, die einen Markt mithilfe eines Orderbuchs simuliert, stellt hierfür eine geeignete Lösung dar.

3. Soll-Situation

Aus Sicht des Auftraggebers soll eine Software entstehen, die es ermöglicht, einen Finanzmarkt realitätsnah zu simulieren. Dabei sollen die grundlegenden Mechanismen eines Orderbuchs abgebildet werden, um daraus Preisverläufe zu generieren, die für Tests, Analysen und Lehrzwecke genutzt werden können.

3.1. Soll-Zustand

Nach Abschluss des Projekts steht eine lauffähige Anwendung zur Verfügung, die einen Markt mithilfe eines Orderbuchs simuliert. Der Benutzer kann verschiedene Parameter des Marktes konfigurieren und den daraus entstehenden Preisverlauf beobachten.

Die Software erzeugt dabei Preisdaten, die gespeichert und später weiterverwendet werden können. Der entstehende Mehrwert liegt darin, dass Test- und Schulungsdaten unabhängig von realen Finanzmärkten generiert werden können und somit keine externen Marktdaten benötigt werden.

3.2 Funktionale Anforderungen

Die Software soll es ermöglichen, einen Markt anhand vorher festgelegter Parameter zu simulieren. Dazu gehört die Möglichkeit, die Anzahl der Käufer und Verkäufer festzulegen sowie die Risikobereitschaft der Marktteilnehmer zu konfigurieren. Außerdem soll ein minimaler und maximaler Preis definiert werden können, zu dem gehandelt wird. Die Geschwindigkeit, mit der sich der Markt entwickelt, soll ebenfalls einstellbar sein.

Die Software soll in der Lage sein, den aktuellen Systemzustand zu speichern und diesen nach einem Absturz oder Neustart wiederherzustellen.

Die Anwendung soll über eine grafische Benutzeroberfläche verfügen, über die alle relevanten Parameter benutzerfreundlich eingestellt werden können. Zusätzlich soll der aktuelle Zustand des Orderbuchs visualisiert werden. Der aus dem Orderbuch entstehende Preis soll in Form eines Graphen dargestellt werden. Dieser Graph soll sowohl auf der X-Achse als auch auf der Y-Achse verschoben werden können und sich mit dem Mausrad skalieren lassen.

Der geschätzte Gesamtaufwand für die Umsetzung der funktionalen Anforderungen liegt bei etwa 110 bis 120 Stunden und ist damit auf einen Projektzeitraum von ungefähr einem Monat ausgelegt.

3.3 Nicht-funktionale Anforderungen

Alle im Rahmen der Simulation ausgeführten Transaktionen sollen dauerhaft in einer zentralen Datenbank gespeichert werden. Die Software soll stabil und performant arbeiten, sodass auch bei einer größeren Anzahl von Marktteilnehmern eine flüssige Simulation möglich ist.

Die Benutzeroberfläche soll leicht verständlich und intuitiv bedienbar sein. Zusätzlich soll sie visuell ansprechend und dynamisch gestaltet werden, um das Interesse des Benutzers zu wecken. Der geschätzte Aufwand für die Umsetzung der nicht-funktionalen Anforderungen beträgt etwa 40 Stunden.

3.4 Schnittstellen

Zur Speicherung und zum Laden der Transaktionen sowie relevanter Simulationsdaten wird eine PostgreSQL-Datenbank eingesetzt. Diese dient als zentrale Schnittstelle zwischen der Anwendung und der persistenten Datenhaltung.

3.5 Risiken

Ein mögliches Risiko besteht darin, dass ein simuliertes Orderbuch allein nicht ausreicht, um alle für reale Finanzmärkte typischen Muster und Preisbewegungen abzubilden, da nicht alle externen Einflussfaktoren berücksichtigt werden können. Dieses Risiko ist vom Auftraggeber zu tragen.

Als alternative Vorgehensweise besteht die Möglichkeit, ein reales Orderbuch über eine entsprechende API-Schnittstelle eines Marktes einzubinden, um die Simulation mit echten Marktdaten zu vergleichen oder zu ergänzen.

Ein weiteres Risiko liegt in der technischen Komplexität der Simulation, insbesondere in Bezug auf Performance und Stabilität. Dieses Risiko ist vom Dienstleister zu tragen und kann durch Optimierungen oder eine Reduzierung der Modellkomplexität minimiert werden.

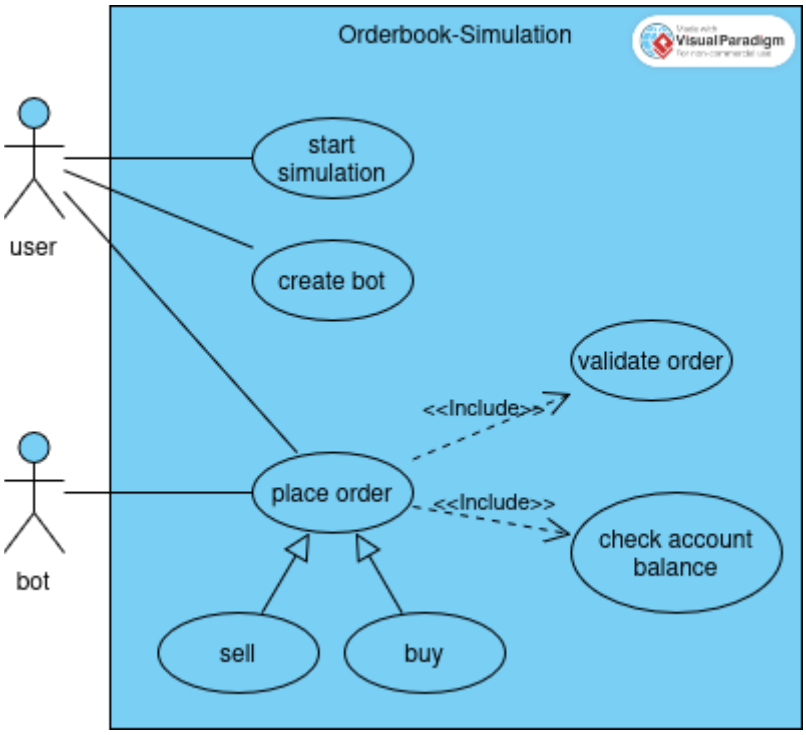
4. Abnahmekriterien

Die Software gilt als abgenommen, wenn das Orderbuch funktional korrekt und realitätsnah berechnet wird und sowohl das Orderbuch als auch der aktuelle Marktpreis visuell dargestellt werden. Alle relevanten Parameter zur Preisgenerierung müssen vom Benutzer einstellbar sein. Sämtliche Transaktionen müssen automatisch in der Datenbank gespeichert werden.

Alle in den funktionalen Anforderungen beschriebenen Anwendungsfälle müssen mindestens einen erfolgreichen Testlauf abgeschlossen haben.

Zu den Kann-Kriterien zählen die Möglichkeit für Benutzer, selbst Kauf- und Verkaufsorders im simulierten Markt zu platzieren, sowie ein Werkzeug, mit dem Markierungen oder Linien im Preisgraphen eingefügt werden können. Diese Funktionen sind optional und nicht zwingend für die Abnahme erforderlich.

5. Use-Case-Diagramm:



USE-CASE 1: Die Simulation starten

Use Case Name	Simulation starten
Akteur(e)	Benutzer
Ziel	Benutzer startet eine Handelssimulation
Vorbedingungen	Benutzer ist angemeldet und die Simulation kann gestartet werden
Nachbedingungen	Simulation läuft, Bot kann erstellt und Aktionen ausgeführt werden
Hauptablauf	1. Benutzer klickt „Simulation starten“ 2. System initialisiert die Simulation
Alternativabläufe / Ausnahmen	1. System kann Simulation nicht starten → Fehlermeldung

USE-CASE 2: Einen Bot erstellen

Use Case Name	Bot erstellen
Akteur(e)	Benutzer
Ziel	Benutzer erstellt einen Handelsbot
Vorbedingungen	Simulation läuft
Nachbedingungen	Bot ist erstellt und kann Aufträge platzieren
Hauptablauf	1. Benutzer wählt „Bot erstellen“2. System legt Bot an
Alternativabläufe / Ausnahmen	1. Bot kann nicht erstellt werden → Fehlermeldung

USE-CASE 3: Eine Verkaufs oder Einkaufsorder erstellen

Use Case Name	Order platzieren
Akteur(e)	Bot (oder Benutzer)
Ziel	Bot platziert einen Kauf- oder Verkaufsauftrag
Vorbedingungen	Bot existiert, Simulation läuft, Benutzerkonto ist aktiv
Nachbedingungen	Order wird erstellt, validiert und (falls möglich) ausgeführt
Hauptablauf	1. Bot wählt „Order platzieren“ 2. System **prüft Kontostand** (`include`) 3. System **validiert Order** (`include`) 4. System führt Order aus
Alternativabläufe / Ausnahmen	1. Kontostand unzureichend → Order abgelehnt2. Order ungültig → Fehlermeldung
Include / Extend	Include: Kontostand prüfen, Order validierenExtend: Kauf oder Verkauf (Buy/Sell)

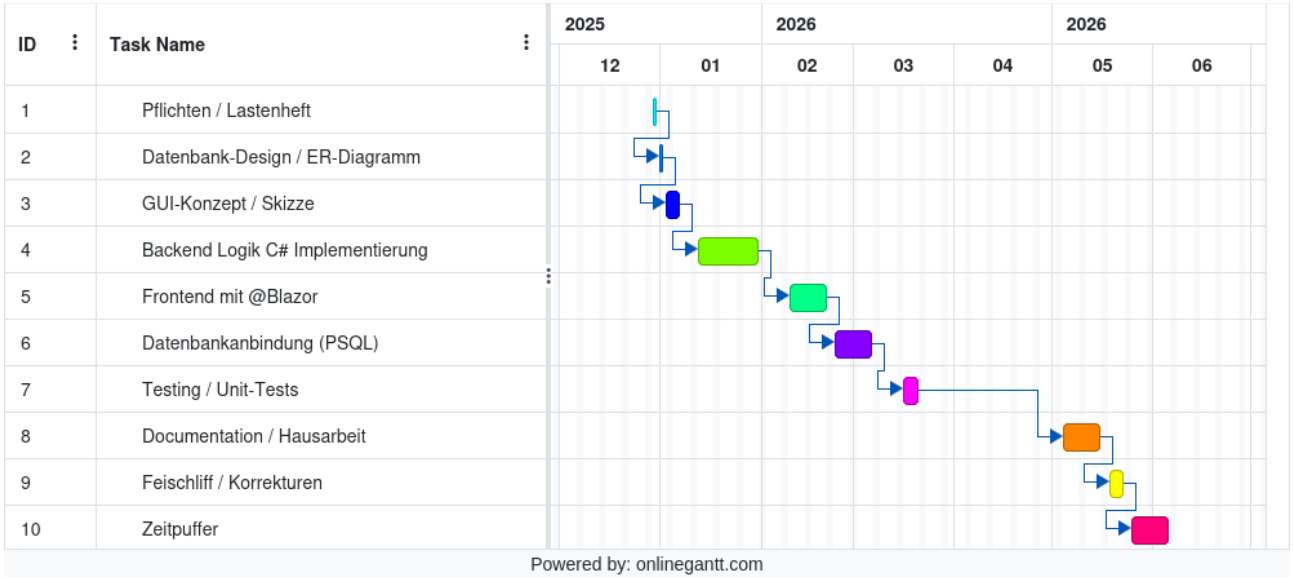
USE-CASE 4: Kaufen

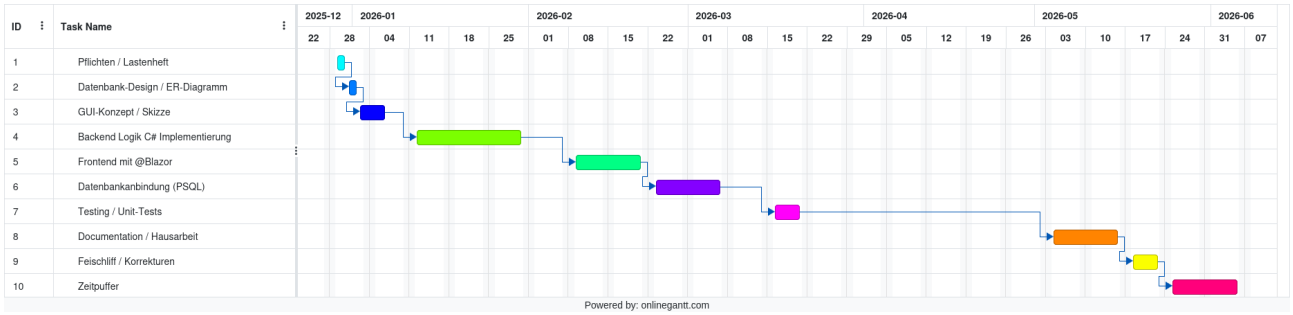
Use Case Name	Buy (Kaufauftrag)
Akteur(e)	Bot
Ziel	Bot kauft einen Vermögenswert
Vorbedingungen	Order platzieren ist möglich
Nachbedingungen	Kauforder wird ausgeführt
Hauptablauf	1. Bot wählt Kaufoption 2. System prüft Kontostand und validiert Order 3. Kauf wird ausgeführt
Alternativabläufe / Ausnahmen	Siehe Order platzieren
Include / Extend	Generalisierung von „Order platzieren“

USE-CASE 5: Verkaufen

Use Case Name	Sell (Verkaufsauftrag)
Akteur(e)	Bot
Ziel	Bot verkauft einen Vermögenswert
Vorbedingungen	Order platzieren ist möglich
Nachbedingungen	Verkaufsorder wird ausgeführt
Hauptablauf	1. Bot wählt Verkaufoption 2. System prüft Kontostand und validiert Order 3. Verkauf wird ausgeführt
Alternativabläufe / Ausnahmen	Siehe Order platzieren
Include / Extend	Generalisierung von „Order platzieren“

6. Projektplan





7. Produktumgebung

1) Blazor Server

Blazor Server ist ein **Web-Framework von Microsoft**, mit dem man **interaktive Webanwendungen komplett in C#** erstellen kann, ohne JavaScript zu verwenden.

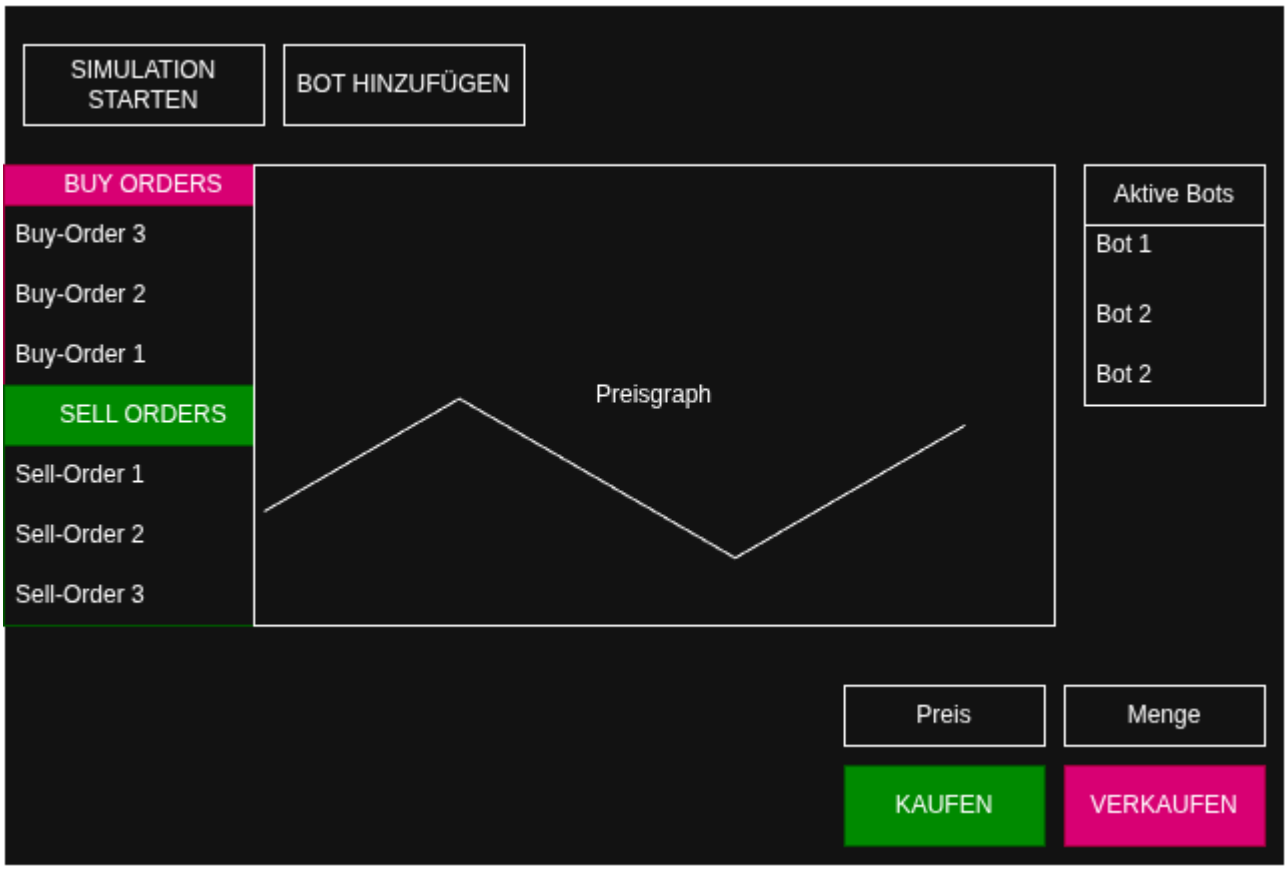
- **Frontend (UI)**: Geschrieben in C# über `.razor`-Komponenten.
- **Backend (Logik & Daten)**: Ebenfalls C# mit ASP.NET Core.
- **Kommunikation**: Die Anwendung läuft auf dem **Server**, während der Browser nur die UI rendert. Änderungen werden in **Echtzeit über SignalR** zwischen Server und Client synchronisiert.

2) PostgreSQL (PSQL)

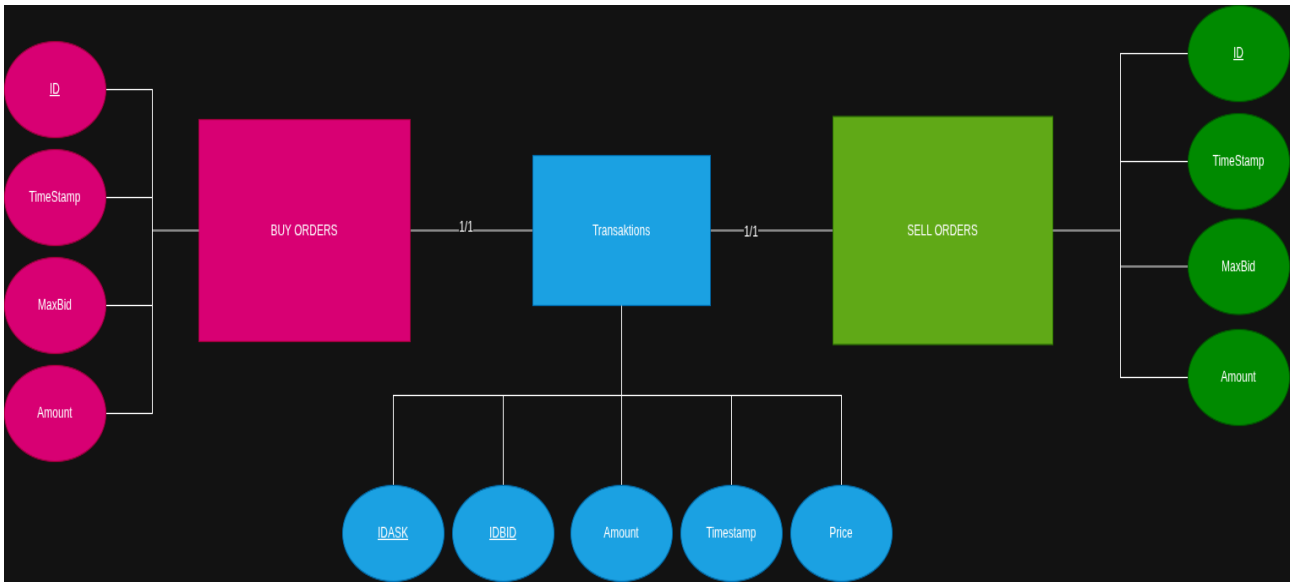
PostgreSQL (häufig PSQL genannt) ist ein **objekt-relationales Open-Source-Datenbanksystem**. Es wird für die Verwaltung von strukturierten Daten in relationalen Datenbanken verwendet und bietet viele moderne Features wie Transaktionen, Erweiterbarkeit und SQL-Konformität.

- **Relational**: Daten werden in Tabellen mit Zeilen und Spalten gespeichert.
- **Open-Source**: Kostenlos nutzbar und erweiterbar.

8. GUI-Skizze



9. DB-Entwurf



10. Testplan

Testplan für das Programm Orderbook-Simulation:

Testfall	Beschreibung	Vorbedingungen	Test-Schritte	Erwartetes Resultat
Test 1	Korrekte Verrechnung des Orderbooks	Orderbook initialisiert, mindestens 2 Orders vorhanden	Neue Orders anlegen, Verrechnung starten	Orders korrekt verrechnet, Mengen und Preise stimmen
Test 2	Richtige Ausführung der Orders	Orderbook korrekt, Testorders vorhanden	Orders eingeben, Ausführung auslösen	Orders werden in richtiger Reihenfolge und zu korrekten Preisen ausgeführt
Test 3	Speicherung in Datenbank	Datenbank erreichbar, Testorders angelegt	Orders ausführen, Datenbank prüfen	Alle Orders korrekt in DB gespeichert
Test 4	Korrekte visuelle Darstellung des Preises	Frontend aktiv, Orderbook aktuell	Preisänderungen erzeugen, Frontend prüfen	Preise korrekt und aktuell angezeigt
Test 5	Parameter sind einstellbar	System gestartet, Parameter änderbar	Parameter ändern, speichern, System prüfen	Parameter werden übernommen und wirken korrekt

Unit-Tests:
Test 1, Test 2 und Test 3 sollen mit Unit-Tests überprüft werden.

11. Git-Repository Link

https://github.com/M0K097/MARKET_SIMULATOR