Projektbericht Trading Simulations Spiel

Mustafa Drescher, Jonas Jäger, Muhammad Tariq und Noah Weisbrod

Projektbeschreibung

Das Projekt ist ein Multiplayer-Trading-Simulator, in dem mehrere Benutzer in einem rennen gegen die Zeit und dem Aktienmarkt gegeneinander antreten können. Jeder Spieler übernimmt die Rolle eines Traders und versucht, innerhalb einer bestimmten Zeitspanne durch den Kauf und Verkauf von Aktien den größten Gewinn/ kleinsten Verlust zu erzielen. Dazu kann der Spieler zwischen mehreren Kursen und Zeitintervallen entscheiden und verschiedene Arten von Käufen tätigen.

Umsetzung

Das Projekt wurde hauptsächlich in Python implementiert, wobei einige Zeilen von CSS das Design etwas aufbessern sollen. Die Architektur des Projekts lässt sich grob in zwei Hauptkomponenten unterteilen: Backend und Frontend.

1. Backend-Komponente

- **Server (server.py)**: Der Server bildet das Rückgrat des Spiels und wurde mit Python und dem socket-Modul realisiert. Der Server verwaltet die Kommunikation zwischen den Clients, kontrolliert den Spielfluss und verarbeitet die Profite der Spieler.
    - **Client-Management**: Der Server kann mehrere Clients akzeptieren und verwaltet diese in einer Liste. Jeder Client wird mit einem eigenen Thread behandelt, damit Spieleraktionen gleichzeitig und unabhängig verarbeitet werden können.
    - **Spielverwaltung**: Der Server verfolgt den Status(ready/ not ready) der Spieler in der Lobby und startet das Spiel, wenn alle Spieler bereit sind. Während des Spiels läuft ein Countdown-Timer, der regelmäßig an alle Clients gesendet wird. Der Timer dient als Taktgeber für das Spiel und endet, sobald die festgelegte Zeit abgelaufen ist.
    - **Profitberechnung**: Während des Spiels werden die Gewinne der Spieler in Echtzeit berechnet und gespeichert. Dies muss zwar nicht jede Sekunde passieren, hätte uns aber noch die Möglichkeit gegeben, das Spiel mit einer Art Leaderboard zu erweitern, welches die Gewinne aller Spieler in Echtzeit erweitert. Am Ende des Spiels werden die Gewinne an alle Clients gesendet, damit jeder Client eine Leaderboard laden kann. Danach wird das Spiel zurückgesetzt.
- **Backend-Logik (backend.py)**: Diese Datei erstellt das User Objekt, das alle Informationen über den Spieler, wie z.B. seine offenen Positionen, Cash usw. speichert. Das Position-Objekt enthält alle Infos zu der jeweiligen offenen Position, die der User besitzt. Ein User-Objekt kann mehrere Position-Objekte besitzen.

Außerdem ist das Backend für das periodische Laden der Dataframes zuständig, mithilfe dessen die Chart angezeigt und aktualisiert wird.

2. Frontend-Komponente

- **Benutzeroberfläche (tradingviewgui.py)**: Das Frontend bietet den Spielern eine GUI, mit der sie interagieren können, während sie durch Kerzen und (hoffentlich grünen) Zahlen ihre Investments verfolgen können. Hierbei werden verschiedene Python-Bibliotheken verwendet, darunter auch PyQt5. Obwohl dieses Modul relativ alt ist, ermöglicht es die Integration des lightweight_chart-Moduls mit Standard-GUI Elementen, wie z.B. Buttons und Textanzeigen Das lightweight_chart ist ein Modul, mit dessen Hilfe die Dataframe, die den Aktienkurs abbildet, als eine Candlestick-Chart veranschaulicht werden kann.
- **Marktdatenanzeige**: Die GUI zeigt dabei nicht nur die Marktbewegung als Chart an, sondern gibt auch Preis, wie viel Cash der Nutzer momentan zur Verfügung hat, wie viel sein Gesamtvermögen ist. Das „unrealized PNL“ (Profit and Loss) gibt an, wie viel der Spieler im Positiven/Negativen mit seinen gekauften Aktien liegt. Das heißt, wenn der Spieler alle seine Aktien verkauft, liegt das PNL bei 0, da er keine Aktien im Besitz hat, mit denen er im Positiven liegen könnte.
    - **Interaktive Handelsfunktionen**: Spieler können über die Benutzeroberfläche Aktien kaufen oder shorten. Außerdem können sie zwischen verschiedenen Zeitintervallen und Aktienkursen wechseln und ihre offenen Positionen mit dem Button oben rechts anzeigen lassen.

Einsatz von KI(ChatGPT)

In dem Projekt wurde ChatGPT unter anderem als Hilfsmittel benutzt, um komplexe Sachverhältnisse zu verstehen, wie z.B. die Berechnung von Inverse ETFs, die eine Möglichkeit bieten, Aktien zu shorten. Außerdem wurde die KI eingesetzt, um bestimmte Codebereiche zu optimieren, wie z.B. bei der Verarbeitung von DataFrames oder der Erstellung von PyQt-Objekten. Auch bei der Serverlogik wurde die KI zur Unterstützung bei Bugs und Problemen herangezogen.

Um das Spiel in Multiplayer zu starten, muss zuerst die server.py Datei ausgeführt werden und schließlich die tradingviewgui.py Datei für jeden Spieler in einem seperaten Terminal gestartet werden. server.py muss nicht nach jedem Spiel neu gestart werden, da die Variablen nach jedem Spiel automatisch zurückgesetzt werden.

Quellen und LLM/GenAI: https://lightweight-charts-python.readthedocs.io/en/latest/tutorials/getting_started.html, https://doc.qt.io/qtforpython-5/, https://chatgpt.com/, https://twelvedata.com/docs#getting-started