# Projektbericht Trading Simulations Spiel

## Mustafa Drescher, Jonas Jäger, Muhammad Tariq und Noah Weisbrod

### Projektbeschreibung

Das Projekt ist ein Multiplayer-Trading-Simulator, der es mehreren Benutzern ermöglicht, in einer simulierten Handelsumgebung gegeneinander anzutreten. Jeder Spieler übernimmt die Rolle eines Traders und versucht, innerhalb einer bestimmten Zeitspanne durch den Kauf und Verkauf von Finanzinstrumenten den größtmöglichen Gewinn zu erzielen. Der Simulator stellt eine Echtzeit-Handelsplattform bereit, auf der die Spieler ihre Strategien anwenden können, um unter Druck zu handeln und Entscheidungen in einer dynamischen Marktumgebung zu treffen.

### Umsetzung

Die Implementierung des Projekts erfolgte hauptsächlich in Python, unterteilt in mehrere Module, die jeweils spezifische Funktionen innerhalb des Systems erfüllen. 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, was eine gleichzeitige und unabhängige Verarbeitung der Aktionen ermöglicht.
  + **Spielverwaltung**: Der Server verfolgt den Status der Spieler 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. Am Ende des Spiels werden diese Ergebnisse an alle Clients gesendet, und das Spiel wird zurückgesetzt.
* **Backend-Logik (backend.py)**: Diese Datei könnte für die Hauptlogik verantwortlich sein, die die Marktdaten verarbeitet, Handelsaktionen simuliert und die Spielregeln durchsetzt. Der genaue Inhalt dieser Datei wurde jedoch nicht direkt untersucht.

#### 2. **Frontend-Komponente**

* **Benutzeroberfläche (verschiedene Python-Dateien wie frontend\_tradingview2.py)**: Das Frontend bietet den Spielern eine grafische Benutzeroberfläche (GUI), mit der sie interagieren können. Hierbei kommen verschiedene Python-Bibliotheken zum Einsatz, wahrscheinlich PyQt oder eine ähnliche GUI-Bibliothek, um die Handelsansicht zu realisieren.
  + **Marktdatenanzeige**: Die GUI zeigt die Marktbewegungen in Echtzeit an, sodass die Spieler die Preisentwicklung verfolgen und fundierte Entscheidungen treffen können.
  + **Interaktive Handelsfunktionen**: Spieler können über die Benutzeroberfläche Kauf- und Verkaufsaufträge erteilen, die direkt an den Server gesendet und dort verarbeitet werden.
* **Datenmanagement (ohlcv.csv)**: Diese Datei enthält möglicherweise historische Marktdaten, die für die Simulation verwendet werden. Diese Daten könnten von der Backend-Komponente geladen und zur Generierung von Marktszenarien genutzt werden, mit denen die Spieler interagieren.

### Technologien bei der Umsetzung

1. **Backend in Python**: Der Server wurde in Python mit dem socket-Modul implementiert, um die Kommunikation zwischen Spielern zu ermöglichen. Python wurde aufgrund seiner schnellen Entwicklungszyklen und leichten Netzwerkprogrammierung gewählt.
2. **Multithreading**: Jeder Spieler wird in einem eigenen Thread verarbeitet, um eine gleichzeitige und reibungslose Spielerfahrung zu gewährleisten.
3. **Datenübertragung mit JSON**: JSON wurde für den Datenaustausch zwischen Server und Clients genutzt, da es leichtgewichtig, lesbar und plattformunabhängig ist.
4. **Frontend mit PyQt**: PyQt wurde verwendet, um eine interaktive und anpassbare Benutzeroberfläche zu schaffen. Es bietet die nötige Flexibilität für die Darstellung von Marktdaten und Handelsoptionen.
5. **Datenverwaltung mit CSV**: Marktdaten werden in CSV-Dateien gespeichert, da diese einfach zu handhaben, effizient und kompatibel mit verschiedenen Softwarelösungen sind.

Die gewählten Technologien und Methoden bieten eine effiziente, flexible und leicht erweiterbare Lösung. Python und seine Bibliotheken ermöglichen schnelle Entwicklung und zuverlässige Leistung, während JSON und CSV eine einfache und effiziente Datenverwaltung und -übertragung gewährleisten. PyQt bietet eine benutzerfreundliche Oberfläche, die das Handelserlebnis verbessert.

### Zusammenfassung

Dieser Trading-Simulator stellt eine komplexe, dennoch benutzerfreundliche Plattform dar, die sowohl Bildungszwecken als auch der Entwicklung und Optimierung von Handelsstrategien dient. Durch die Kombination von Echtzeit-Datenverarbeitung, einer interaktiven Benutzeroberfläche und einem Multiplayer-Ansatz wird eine immersive und herausfordernde Umgebung geschaffen, die den Benutzern ein umfassendes Handels-Erlebnis bietet.

## 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>