Real time finance game

Description

For my personal project, I decided to expand the peer assignment by adding trading features like currency exchange with real-time market values to the banking application. This enhancement leverages F#'s capabilities and differentiates from the original peer project by ensuring that no new money can be created. The application will support multiple accounts, such as "cash" and "bank." Withdrawals move funds into the cash account, and deposits do the reverse. Insufficient cash prevents deposits into the bank. Trading is restricted to the bank account. For example, with 100 CHF in your bank account, you can exchange it for USD at the current market rate. If you relaunch the application hours later, you can exchange the USD back to CHF at the updated rate.

The name of the project now is a short description of the application I make. It is on purpose just “finance game” and not “trading / exchange game” since I may expand the application with different possibilities to earn money.

Motivation

I think this project is well-suited for F# due to is strong functional programming features and type safety. Furthermore, I think this is an interesting project since you can extend it further anytime. You can create «mocked» functionalities like automatic cash outs or so on.

Key Language Features

- Pure Functions
- Immutable Data
- Pipelines
- Type Safety

Basic System Architecture

User Interface: Simple but user-friendly console user interface to interact with specified commands

Core Logic:

- Banking account where you can withdraw money (not necessary for project, but is part of the peer assignment and I would therefore like to implement it as well)
- Exchange Currency by market values
- Save current values to file on exiting program correctly

- Load current values from file to retrieve last status

Scope

Must-haves:

- Basic System Architecture
- Basic error handling for not enough money / wrong inputs

Nice-to-haves:

- Buy out method which looks on startup if it was reached since you bought a currency, and if it did you would have exchanged at this point. (not needed to run in background)
- Statistics about loss/wins due to trades
- A type of “gambling” as a minigame/casino were you can earn money “quickly”

Out-of-scope:

- Special user interface (only console interface, since I want to concentrate on F# and not waste time on creating a react app or something and having the F# as a backend only.

Potential Challenges

Since I do not know how hard this task will be, I already decided on a extendable project. As I see, the basic system architecture should be possible to do. As soon as the project is implemented, I first will try to improve my code further. Having better code leads to better understanding. This is also a reason why I decided on a not to hard project to implement but still have different aspects in it.

Challenges:

- Get Stuck with API calls for receiving current exchange course -> could be mocked with a formula and create a “fake” exchange course
- Real Time Market can change rapidly -> implement a mechanism that locks in the exchange course and do not update too quickly
- Too much complexity -> If I would need to cut any of the main features, I would contact Carol Alexandru and have a look at it
- Not enough complexity -> In case I am too quickly with the main aspects I need a good documentation for Carol to know what I added to my project.