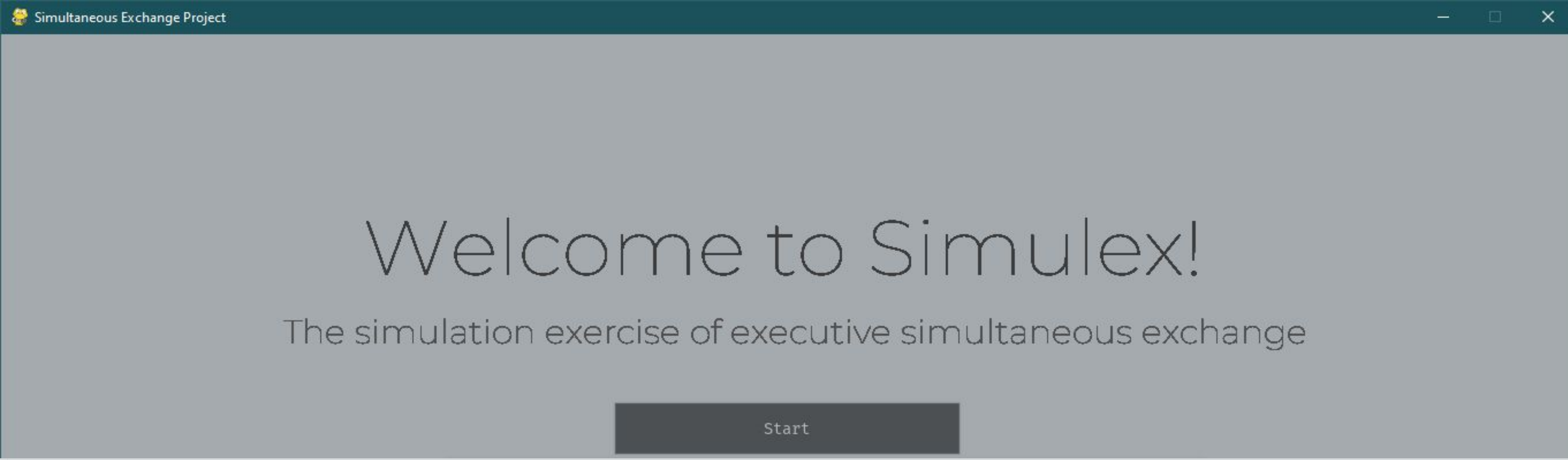


Welcome to Simulex!

The simulation exercise of executive simultaneous exchange

Simulex

A scenario-based financial literacy learning tool

Lachlan Pham, Peter Mahhov, Mathilde Simoni

Software Engineering

Spring 2022

Table of Contents

Background 3
Requirements 4
Solution 5
Stakeholders 6
Design
    Timelines 7
    Splitting time 8
    Stocks and loans 9
Implementation 10
Testing 11
Demo 12
Future Work 13

Background - Requirements Gathering

Problem - Literature Review

- Educate young people in financial literacy
    - Address the barriers in learning about financial markets and investing
    - Help individuals know the risks of investing in the stock market

Motivation - Interviews

- Most means of financial education are boring and are discouragingly complex
- Inspired by the low-risk environment of other trading simulators

Only 33% of adults worldwide are "financially literate"
(Klapper, Lusardi and van Oudheusden, 2015)

Requirements

- The user should be able to play through various financial simulation scenarios of increasing complexity
- The user should be able to experiment with various financial instruments (stocks, loans) with differing risk profiles (changing trends, volatility).
    - The user shall be able to buy and sell stocks
    - The user should be able to take out and pay off a loan
- The user should be able to access information about the historical behavior of the financial instruments they can invest in
- The user should be able to learn new mechanics from helpful information included in each scenario
- The user should enjoy using the system

Stakeholders

Developers

User 1: Beginner

Uses the system as a learning tool and a way to gain experience in investing in the stock market

Has low initial level of financial education

User 2: Experienced

Uses the system to have fun and is looking for a new interesting experience

Has high initial level of financial literacy

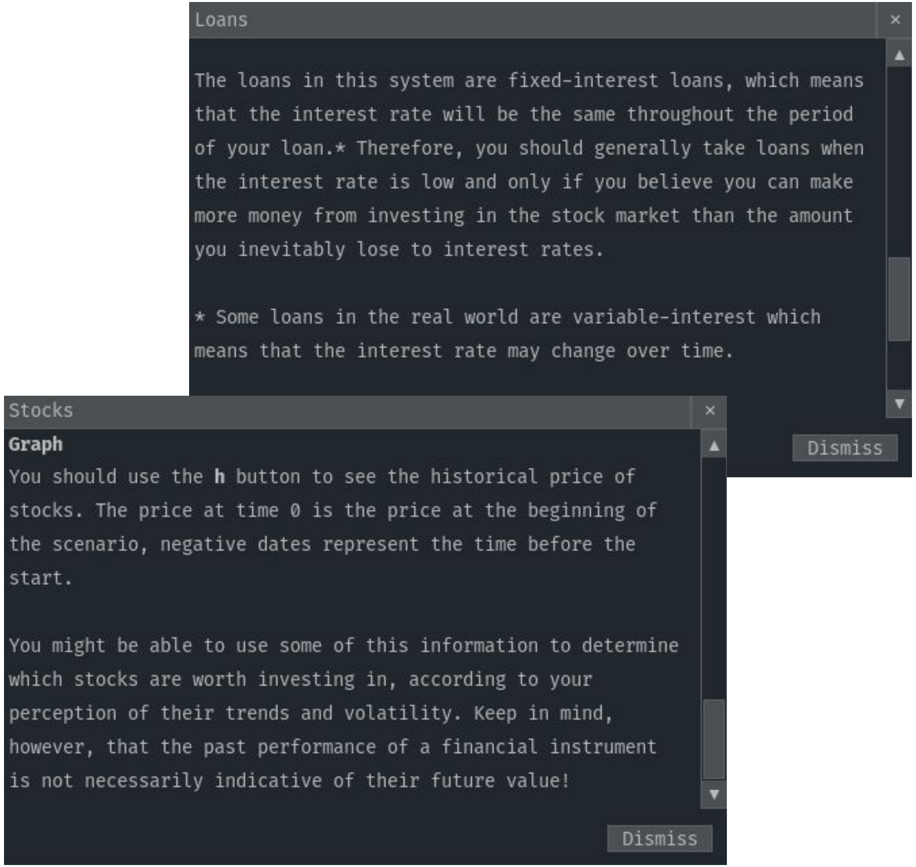
Solution

A captivating twist on stock exchange simulators

Featuring few, essential financial instruments

Crafted scenarios which gradually introduce strategic and educational complexity

A novel and unique time-splitting mechanic



The loans in this system are fixed-interest loans, which means that the interest rate will be the same throughout the period of your loan.* Therefore, you should generally take loans when the interest rate is low and only if you believe you can make more money from investing in the stock market than the amount you inevitably lose to interest rates.

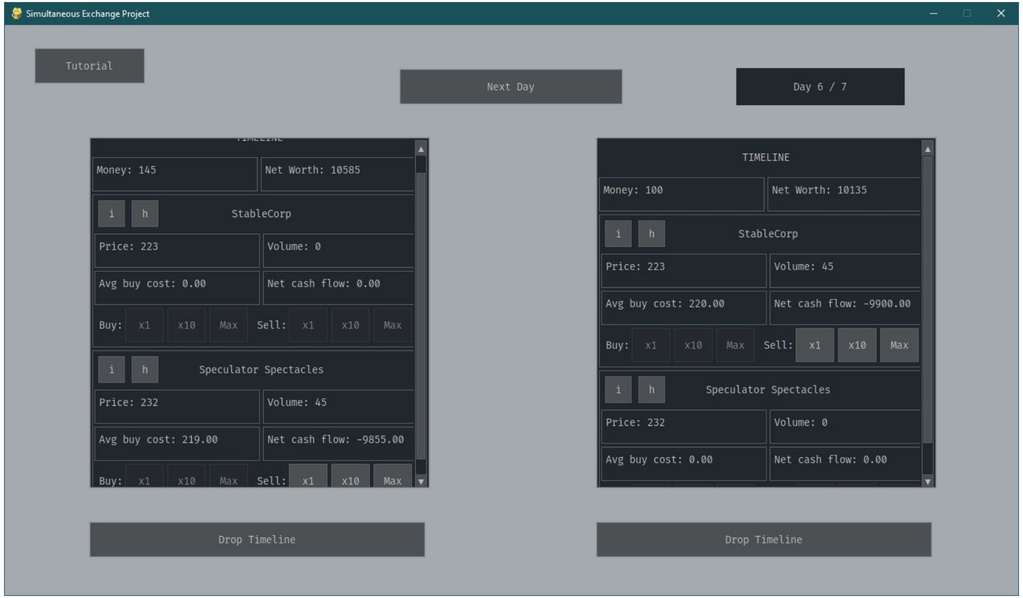
* Some loans in the real world are variable-interest which means that the interest rate may change over time.

Graph

You should use the **h** button to see the historical price of stocks. The price at time 0 is the price at the beginning of the scenario, negative dates represent the time before the start.

You might be able to use some of this information to determine which stocks are worth investing in, according to your perception of their trends and volatility. Keep in mind, however, that the past performance of a financial instrument is not necessarily indicative of their future value!

Design: Splitting time



Day 6 / 7

TIMELINE

Money: 145 | Net Worth: 10585

StableCorp
Price: 223 | Volume: 0
Avg buy cost: 0.00 | Net cash flow: 0.00

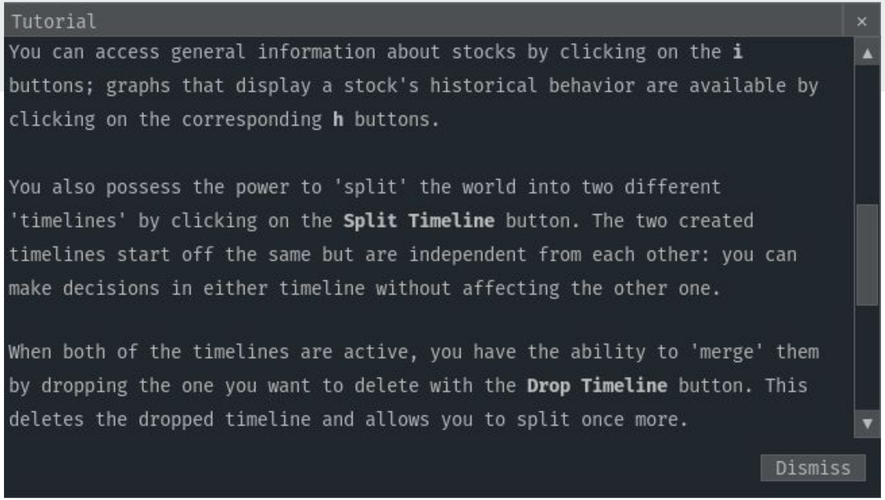
Speculator Spectacles
Price: 232 | Volume: 45
Avg buy cost: 219.00 | Net cash flow: -9855.00

TIMELINE

Money: 100 | Net Worth: 10135

StableCorp
Price: 223 | Volume: 45
Avg buy cost: 220.00 | Net cash flow: -9900.00

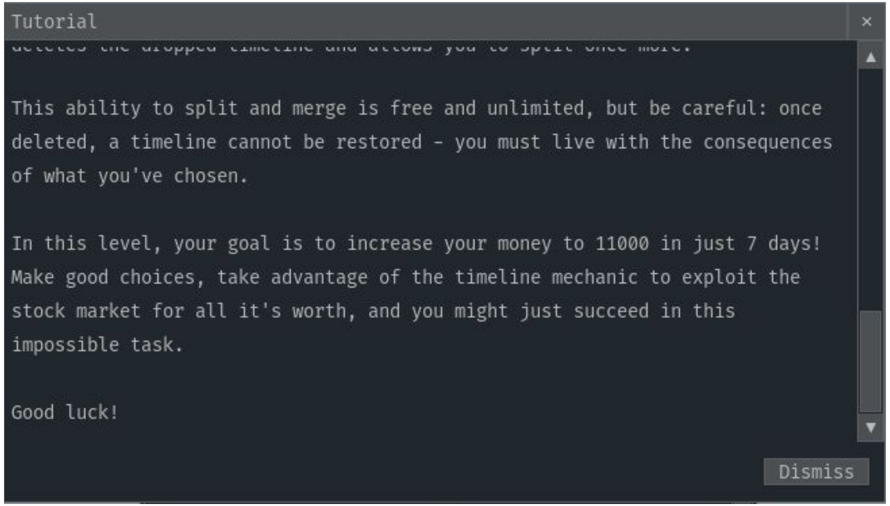
Speculator Spectacles
Price: 232 | Volume: 0
Avg buy cost: 0.00 | Net cash flow: 0.00



You can access general information about stocks by clicking on the **i** buttons; graphs that display a stock's historical behavior are available by clicking on the corresponding **h** buttons.

You also possess the power to 'split' the world into two different 'timelines' by clicking on the **Split Timeline** button. The two created timelines start off the same but are independent from each other: you can make decisions in either timeline without affecting the other one.

When both of the timelines are active, you have the ability to 'merge' them by dropping the one you want to delete with the **Drop Timeline** button. This deletes the dropped timeline and allows you to split once more.



deletes the dropped timeline and allows you to split once more.

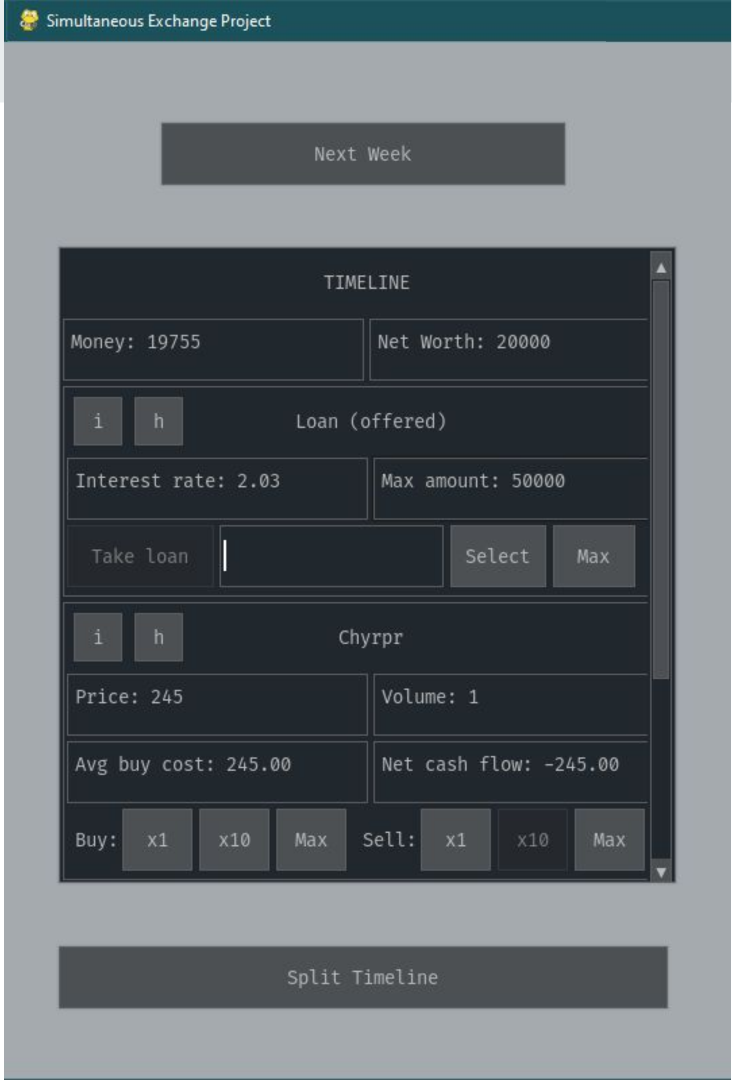
This ability to split and merge is free and unlimited, but be careful: once deleted, a timeline cannot be restored - you must live with the consequences of what you've chosen.

In this level, your goal is to increase your money to 11000 in just 7 days! Make good choices, take advantage of the timeline mechanic to exploit the stock market for all it's worth, and you might just succeed in this impossible task.

Good luck!

Design: Timeline

- Money
- Net worth
- Loan (offered)
    - (current) Interest rate
    - Max amount
- Loan (taken)
    - (fixed) Interest rate
    - Debt
- Stock
    - Price
    - Volume
    - Average buy cost
    - Net cash flow



Simultaneous Exchange Project

TIMELINE

Money: 19755

Net Worth: 20000

Loan (offered)

Interest rate: 2.03

Max amount: 50000

Chyrpr

Price: 245

Volume: 1

Avg buy cost: 245.00

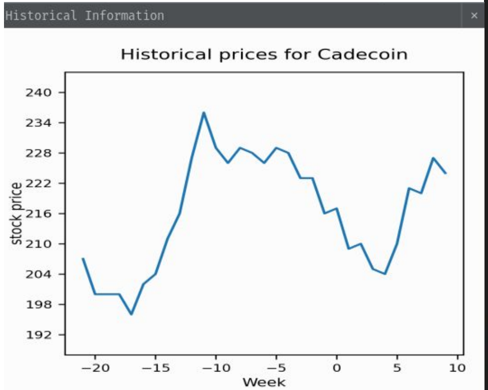
Net cash flow: -245.00

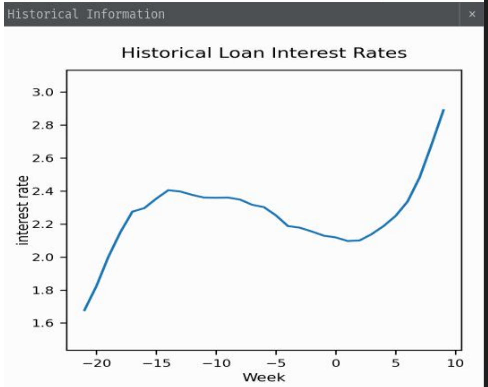
Design: Stocks and Loans

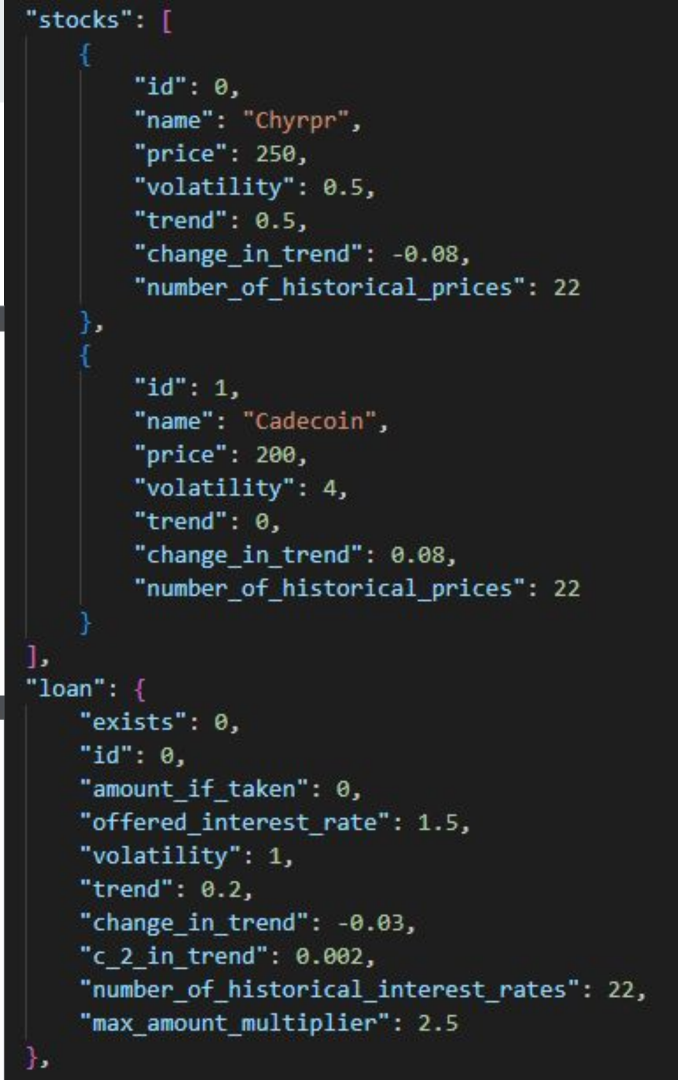
- Access to “Historical Information”
- Parameters:
  - Price/Interest rate
  - Volatility
  - Trend
  - Change in trend
  - Second derivative of trend (loans only)
- New price/interest rate from normal distribution after time skip:

$$X \sim N(\text{mean: price} + \text{trend}, \text{std: price} \times \text{volatility} \div 100)$$

- Modules need careful balancing







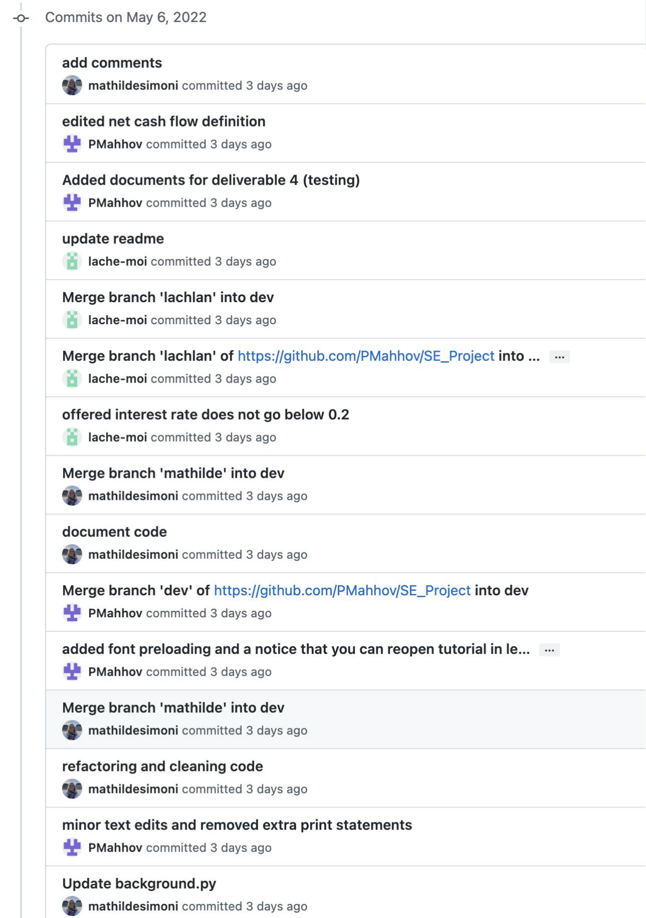
```
"stocks": [
    {
        "id": 0,
        "name": "Chyrpr",
        "price": 250,
        "volatility": 0.5,
        "trend": 0.5,
        "change_in_trend": -0.08,
        "number_of_historical_prices": 22
    },
    {
        "id": 1,
        "name": "Cadecoin",
        "price": 200,
        "volatility": 4,
        "trend": 0,
        "change_in_trend": 0.08,
        "number_of_historical_prices": 22
    }
],
"loan": {
    "exists": 0,
    "id": 0,
    "amount_if_taken": 0,
    "offered_interest_rate": 1.5,
    "volatility": 1,
    "trend": 0.2,
    "change_in_trend": -0.03,
    "c_2_in_trend": 0.002,
    "number_of_historical_interest_rates": 22,
    "max_amount_multiplier": 2.5
},
```

Implementation

Programming language: Python (3.9)

Libraries:

- pygame (2.1.2)
- pygame_gui (0.6.4)
- numpy (1.21.5)
- matplotlib (3.5.1)



Commits on May 6, 2022

add comments
mathildesimoni committed 3 days ago

edited net cash flow definition
PMahhov committed 3 days ago

Added documents for deliverable 4 (testing)
PMahhov committed 3 days ago

update readme
lache-moi committed 3 days ago

Merge branch 'lachlan' into dev
lache-moi committed 3 days ago

Merge branch 'lachlan' of https://github.com/PMahhov/SE_Project into ... ···
lache-moi committed 3 days ago

offered interest rate does not go below 0.2
lache-moi committed 3 days ago

Merge branch 'mathilde' into dev
mathildesimoni committed 3 days ago

document code
mathildesimoni committed 3 days ago

Merge branch 'dev' of https://github.com/PMahhov/SE_Project into dev
PMahhov committed 3 days ago

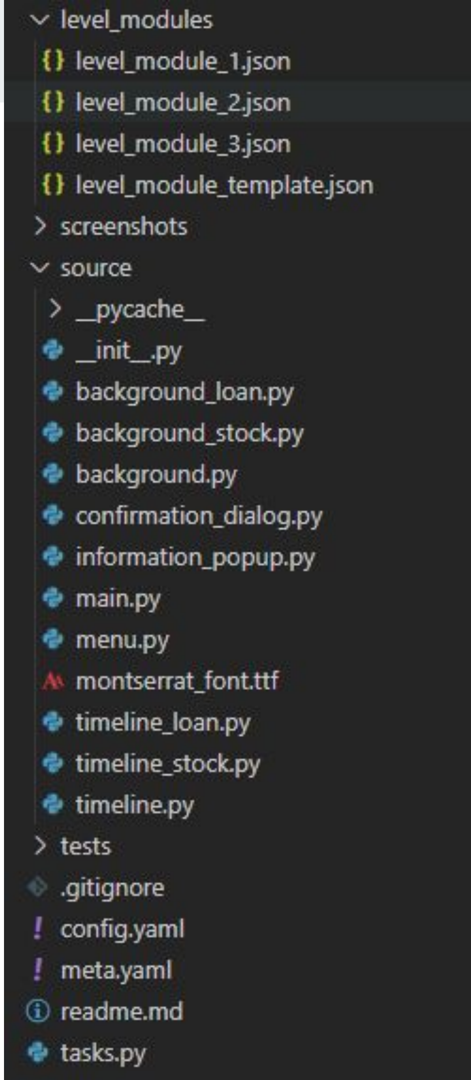
added font preloading and a notice that you can reopen tutorial in le... ···
PMahhov committed 3 days ago

Merge branch 'mathilde' into dev
mathildesimoni committed 3 days ago

refactoring and cleaning code
mathildesimoni committed 3 days ago

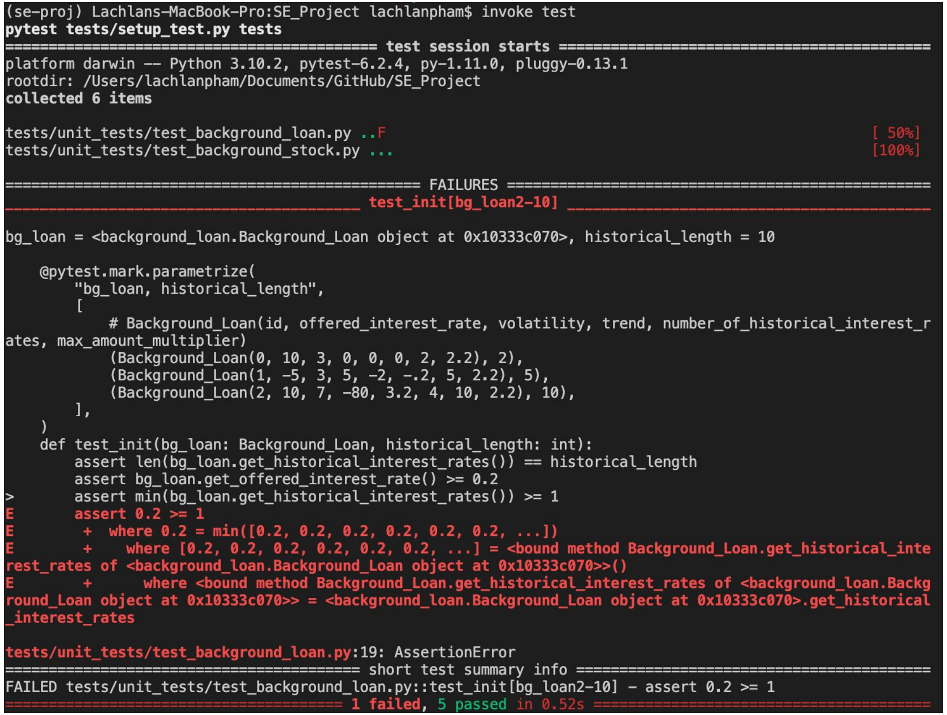
minor text edits and removed extra print statements
PMahhov committed 3 days ago

Update background.py
mathildesimoni committed 3 days ago



```
level_modules
  level_module_1.json
  level_module_2.json
  level_module_3.json
  level_module_template.json
screenshots
source
  __pycache__
  __init__.py
  background_loan.py
  background_stock.py
  background.py
  confirmation_dialog.py
  information_popup.py
  main.py
  menu.py
  montserrat_font.ttf
  timeline_loan.py
  timeline_stock.py
  timeline.py
tests
.gitignore
config.yaml
meta.yaml
readme.md
tasks.py
```

Testing



```
(se-proj) Lachlans-MacBook-Pro:SE_Project lachlanpham$ invoke test
pytest tests/setup_test.py tests
=========================================== test session starts ============================================
platform darwin -- Python 3.10.2, pytest-6.2.4, py-1.11.0, pluggy-0.13.1
rootdir: /Users/lachlanpham/Documents/GitHub/SE_Project
collected 6 items

tests/unit_tests/test_background_loan.py ..F                                                         [ 50%]
tests/unit_tests/test_background_stock.py ...                                                        [100%]

================================================= FAILURES =================================================
__________________________________________ test_init[bg_loan2-10] __________________________________________

bg_loan = <background_loan.Background_Loan object at 0x10333c070>, historical_length = 10

    @pytest.mark.parametrize(
        "bg_loan, historical_length",
        [
            # Background_Loan(id, offered_interest_rate, volatility, trend, number_of_historical_interest_r
ates, max_amount_multiplier)
            (Background_Loan(0, 10, 3, 0, 0, 0, 2, 2.2), 2),
            (Background_Loan(1, -5, 3, 5, -2, -.2, 5, 2.2), 5),
            (Background_Loan(2, 10, 7, -80, 3.2, 4, 10, 2.2), 10),
        ],
    )
    def test_init(bg_loan: Background_Loan, historical_length: int):
        assert len(bg_loan.get_historical_interest_rates()) == historical_length
        assert bg_loan.get_offered_interest_rate() >= 0.2
>       assert min(bg_loan.get_historical_interest_rates()) >= 1
E       assert 0.2 >= 1
E        +  where 0.2 = min([0.2, 0.2, 0.2, 0.2, 0.2, 0.2, ...])
E        +    where [0.2, 0.2, 0.2, 0.2, 0.2, 0.2, ...] = <bound method Background_Loan.get_historical_inte
rest_rates of <background_loan.Background_Loan object at 0x10333c070>>()
E        +      where <bound method Background_Loan.get_historical_interest_rates of <background_loan.Backg
round_Loan object at 0x10333c070>> = <background_loan.Background_Loan object at 0x10333c070>.get_historical
_interest_rates

tests/unit_tests/test_background_loan.py:19: AssertionError
========================================== short test summary info =========================================
FAILED tests/unit_tests/test_background_loan.py::test_init[bg_loan2-10] - assert 0.2 >= 1
======================================== 1 failed, 5 passed in 0.52s =======================================
```

We covered:

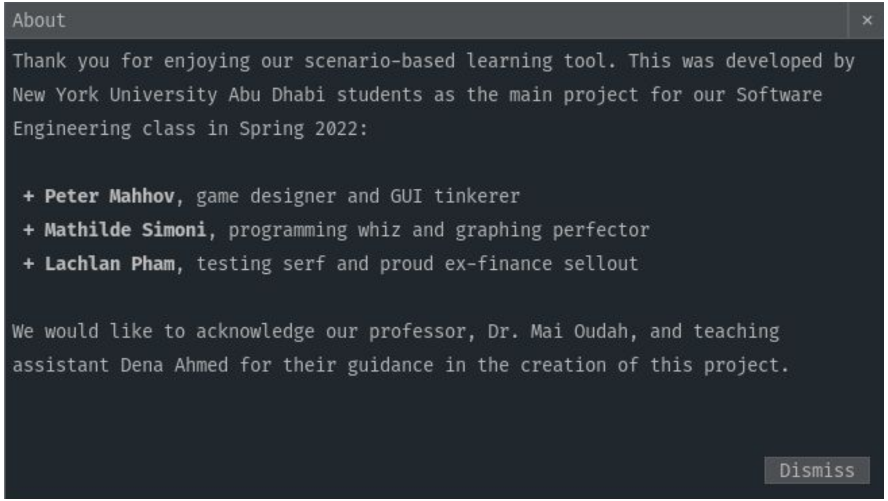
- Unit testing
  - **Pytest:** python testing tool to automate functional testing
  - **Invoke**: used to automatically run shell commands for testing
- Integration testing
- Validation testing: for each use case
- Regression testing

Live Demonstration

Future work

- More scenarios and win-conditions
- An overarching theme, cohesive plot
- Overhauling visuals
- Expanding number of tradable financial instruments
- Procedural generation

Questions?



Thank you for enjoying our scenario-based learning tool. This was developed by New York University Abu Dhabi students as the main project for our Software Engineering class in Spring 2022:

+ **Peter Mahhov**, game designer and GUI tinkerer
+ **Mathilde Simoni**, programming whiz and graphing perfector
+ **Lachlan Pham**, testing serf and proud ex-finance sellout

We would like to acknowledge our professor, Dr. Mai Oudah, and teaching assistant Dena Ahmed for their guidance in the creation of this project.

References:

Klapper, L., Lusardi, A. and van Oudheusden, P., 2015. *FINANCIAL LITERACY AROUND THE WORLD: INSIGHTS FROM THE STANDARD & POOR’S RATINGS SERVICES GLOBAL FINANCIAL LITERACY SURVEY*. [online] Standard & Poor's Ratings Services, p.7. Available at: <https://gflec.org/wp-content/uploads/2015/11/3313-Finlit_Report_FINAL-5.11.16.pdf> [Accessed 9 May 2022].