Rise & Shine

project documentation

Prepared, designed and programmed
by Katarina Miksche

as the final project at
CFGdegree SW Development summer 2022

Table of content

Main objectives

Rise & Shine is a stock market application. It allows users to track and trade stocks, manage multiple portfolios and see the gains in a friendly graphic interface. Results are available online through API as well.

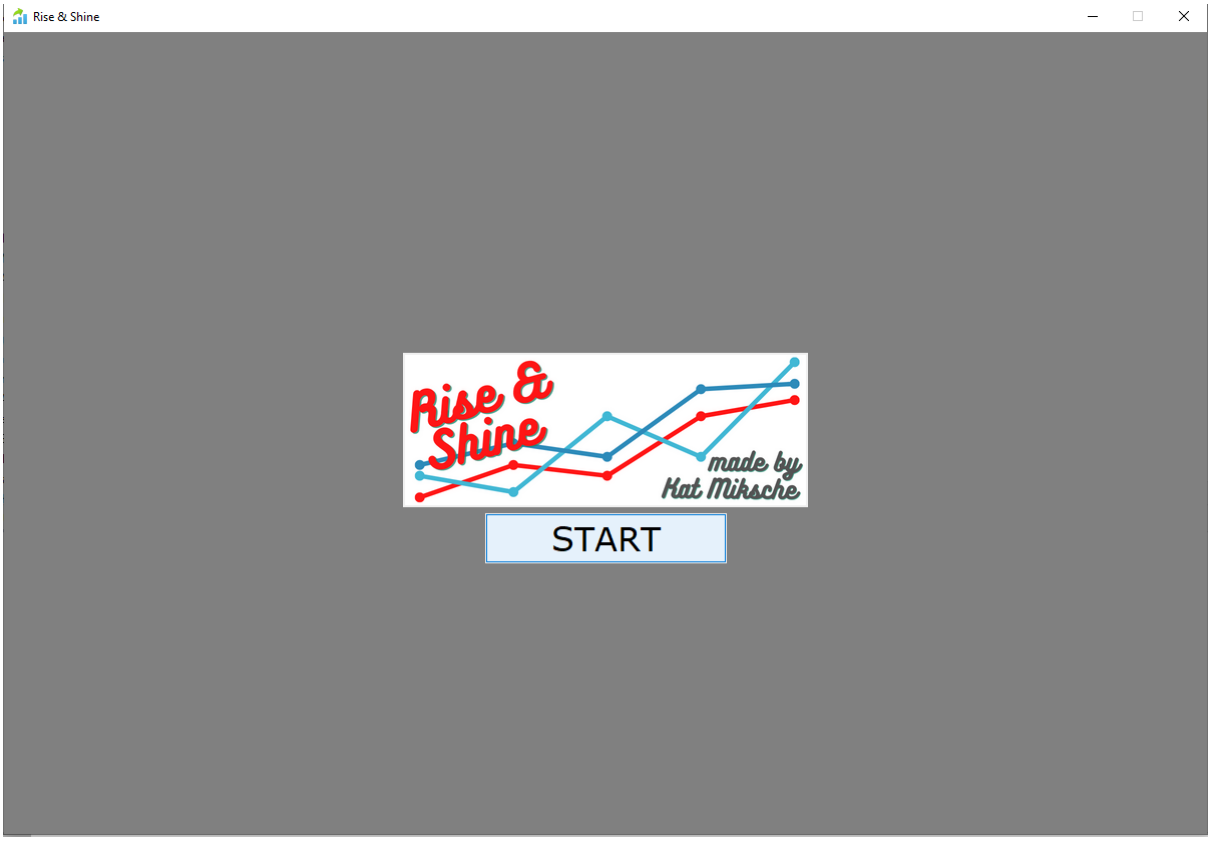
Full functionality available through the graphic interface includes an overview of portfolios including their current values, more detailed view of each portfolio and its historic performance including graph, search tool for tickers and stock performance view, buy/sell functionality, monetary operations in wallet, start of new portfolio and option to close it fast as well.

Limited functionality available online via own API includes an overview of portfolios including their current values, more detailed view of each portfolio and its historic performance, stock performance view and most recent stock price.

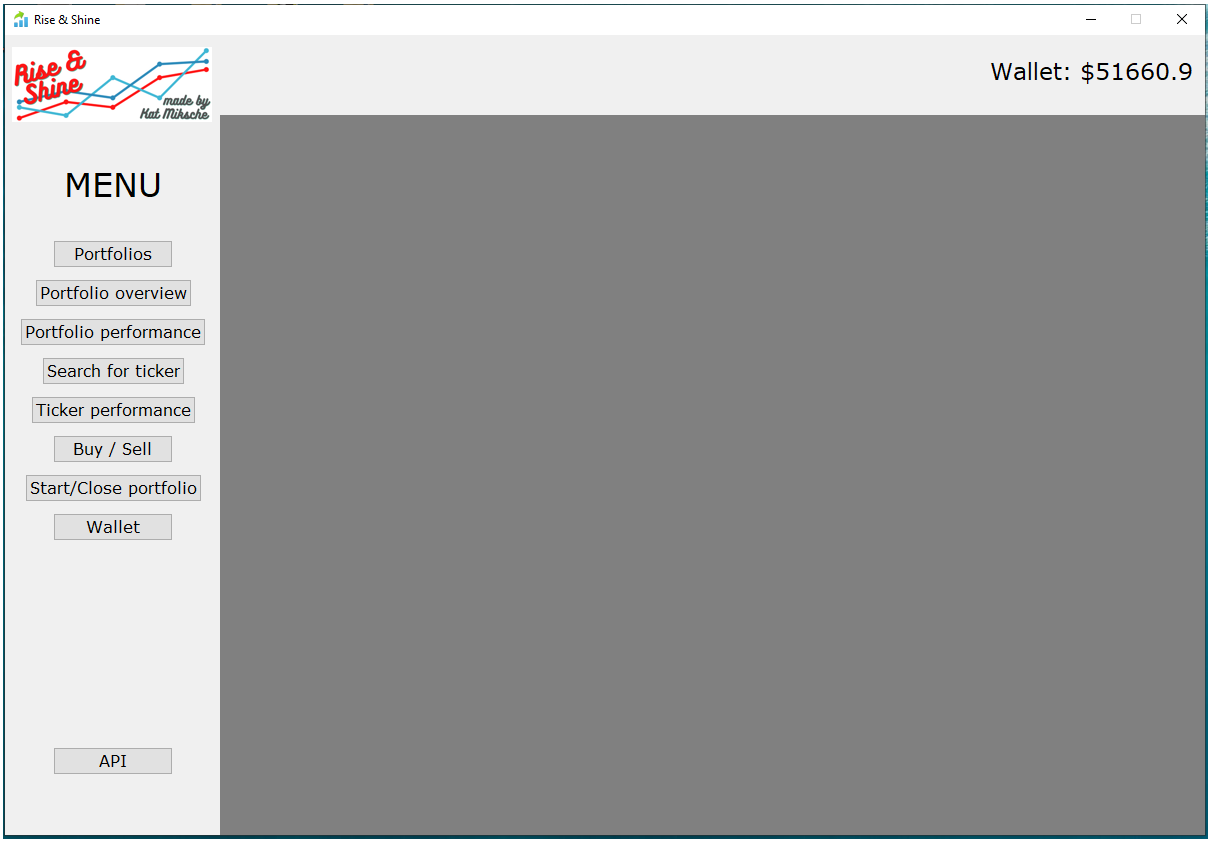
Functionality

GUI

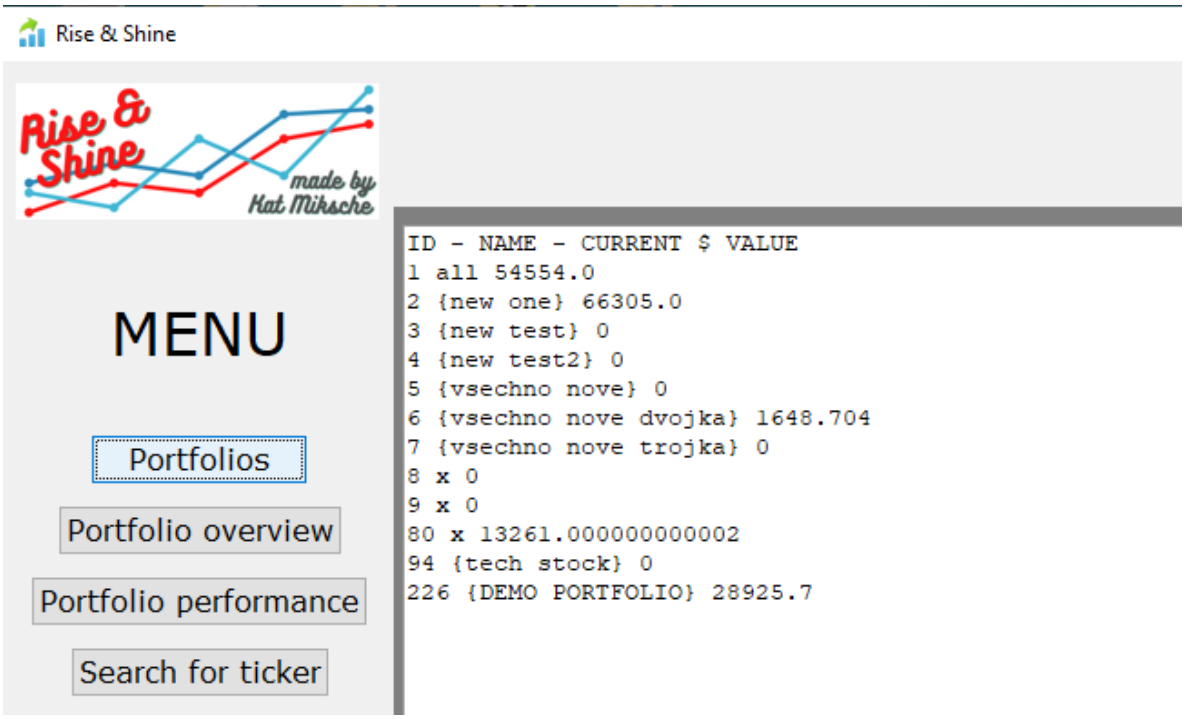
Start screen



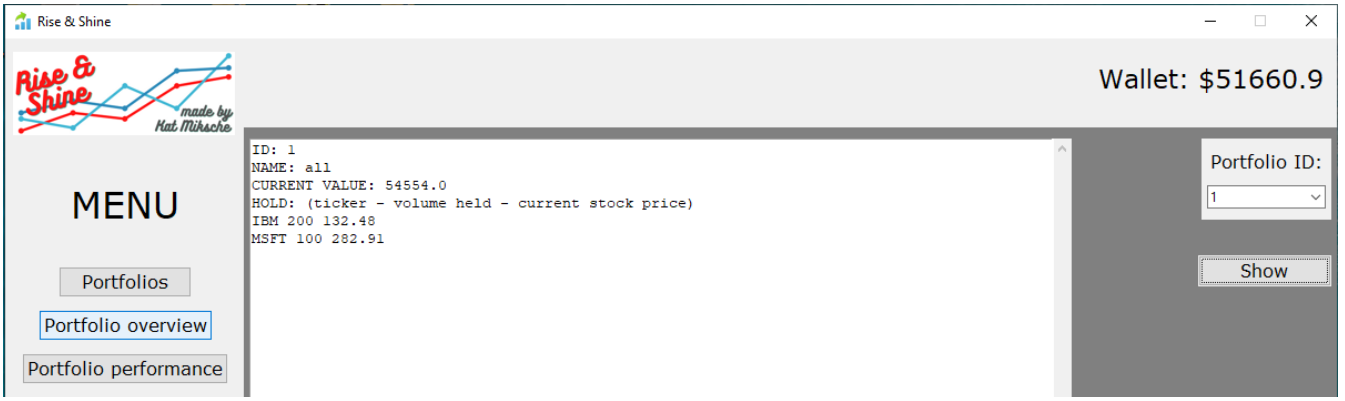
First screen after start and successful load of data



Portfolios - top view

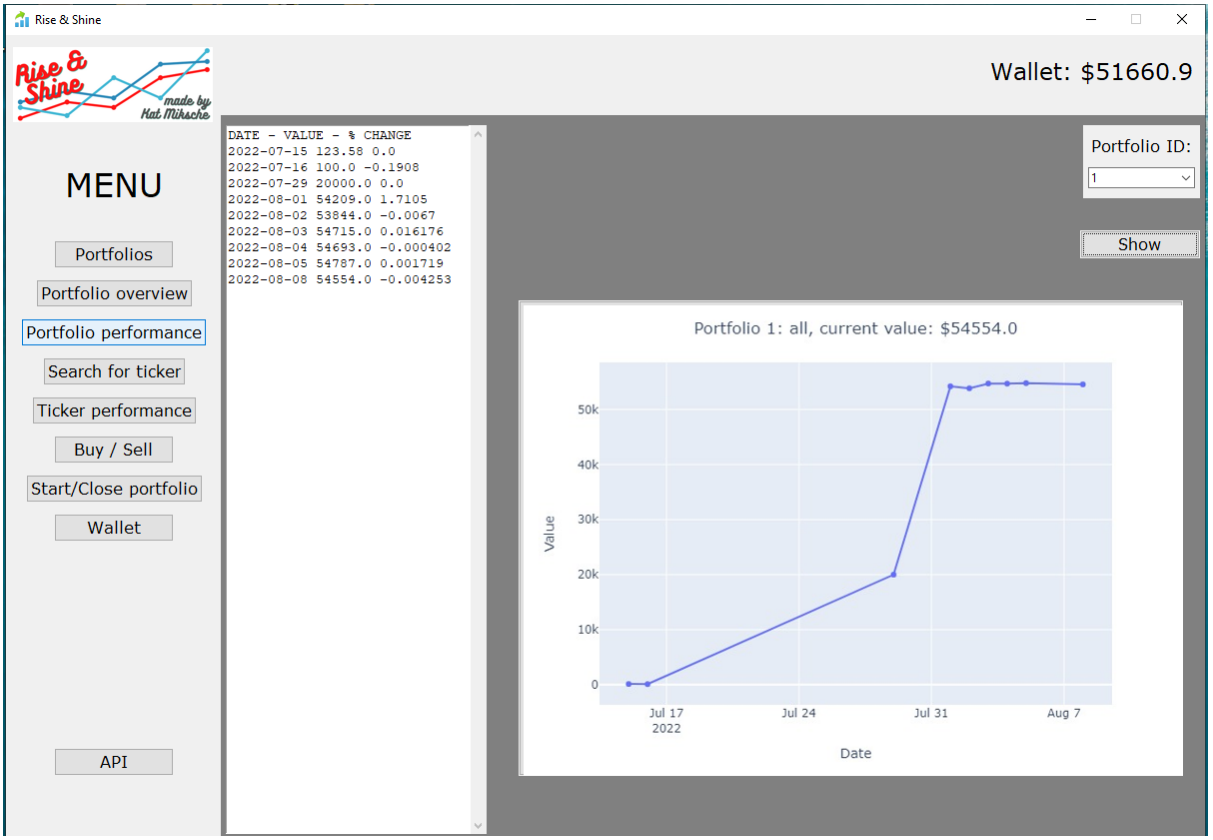


Portfolio overview

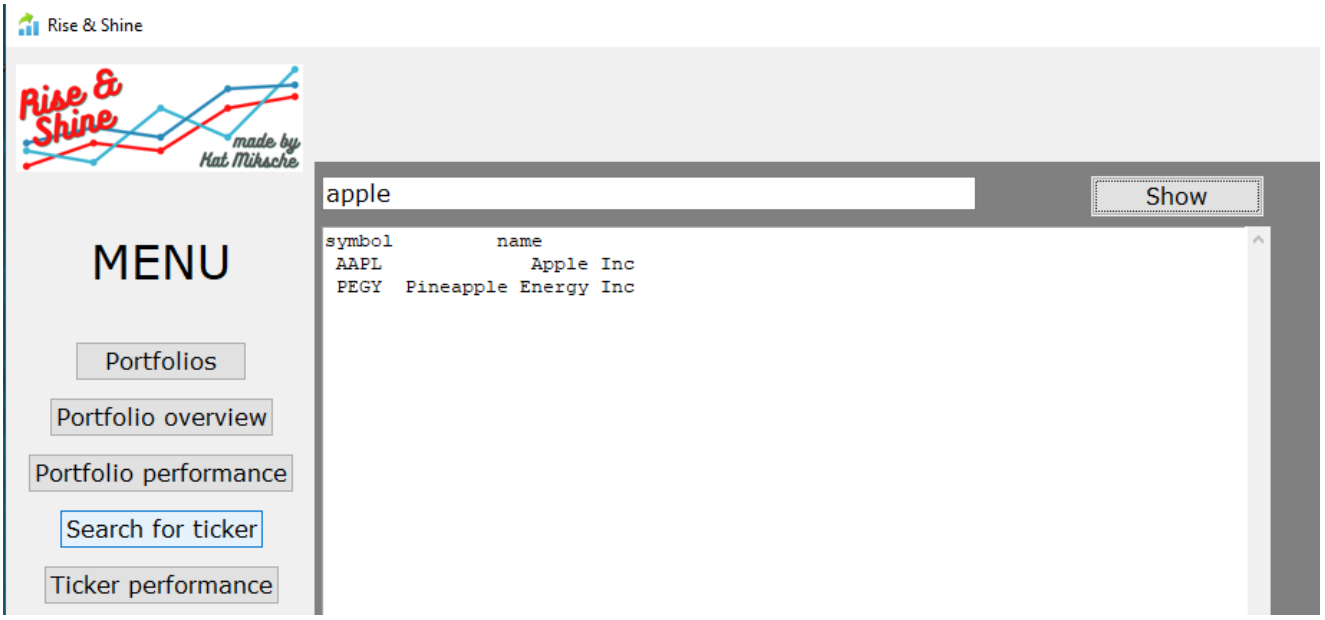


Portfolio performance with graph

(graph is clickable, opens as an active object in a browser)

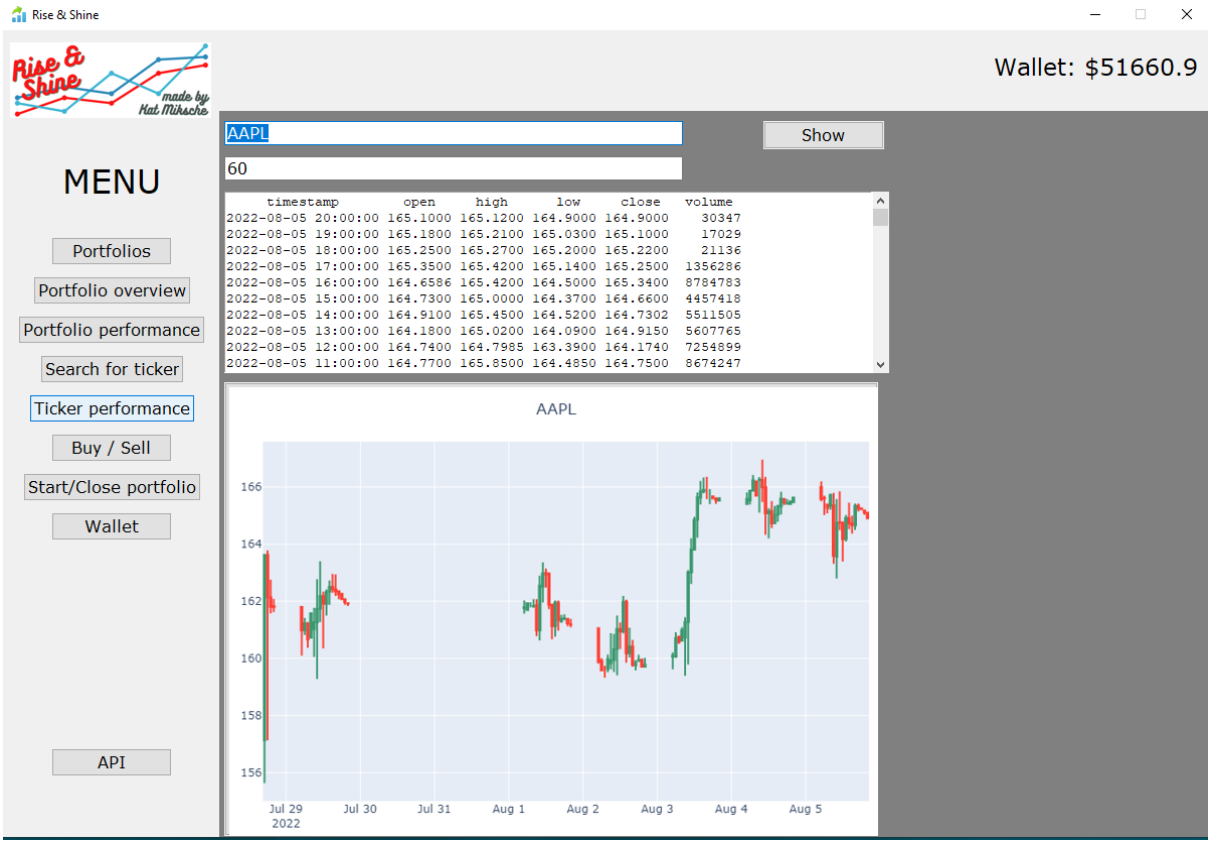


Search for ticker tool

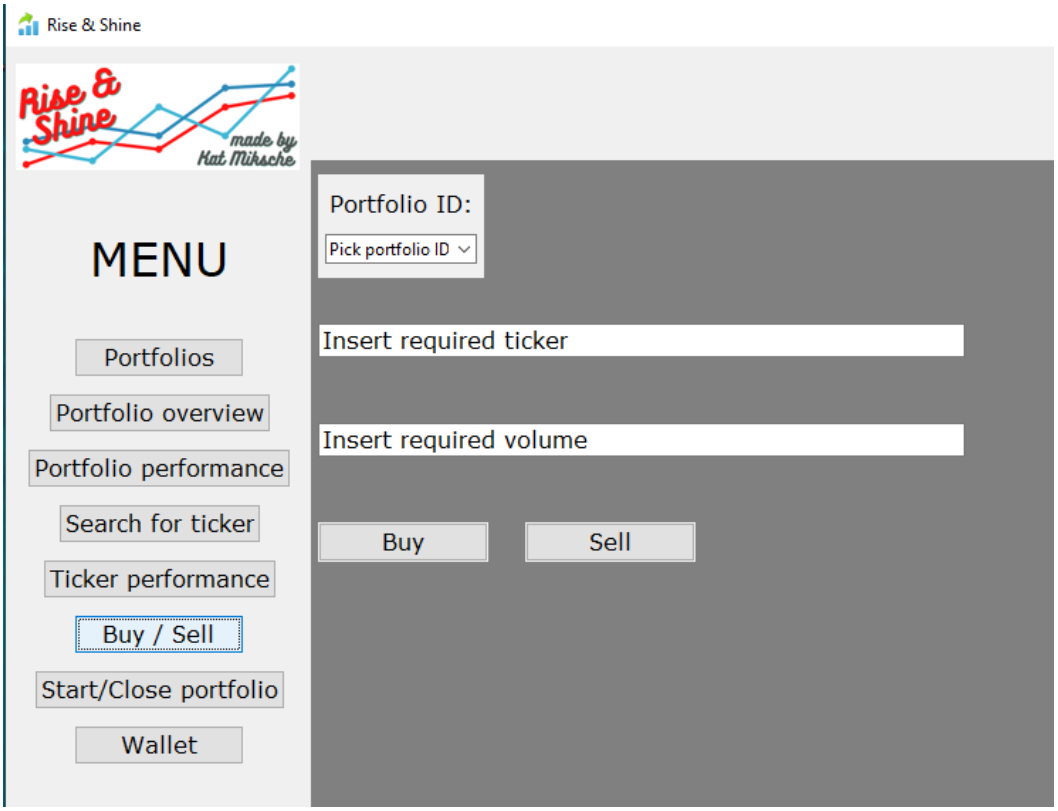


Ticker performance with graph

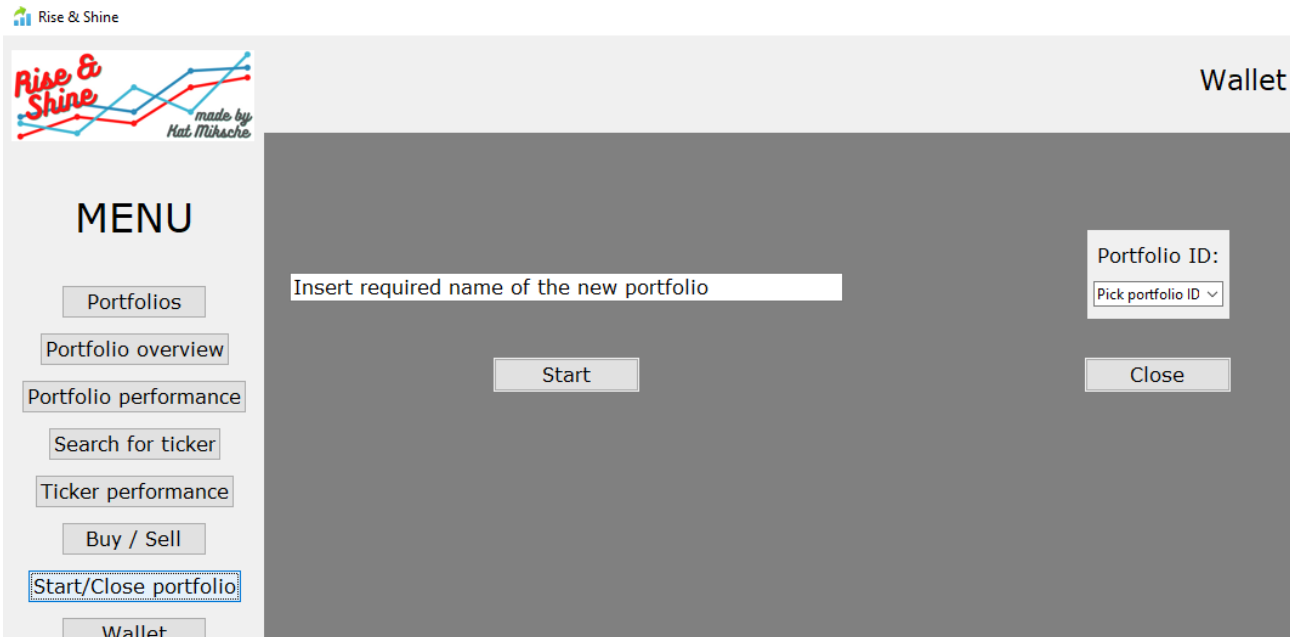
(graph is clickable, opens as an active object in a browser)



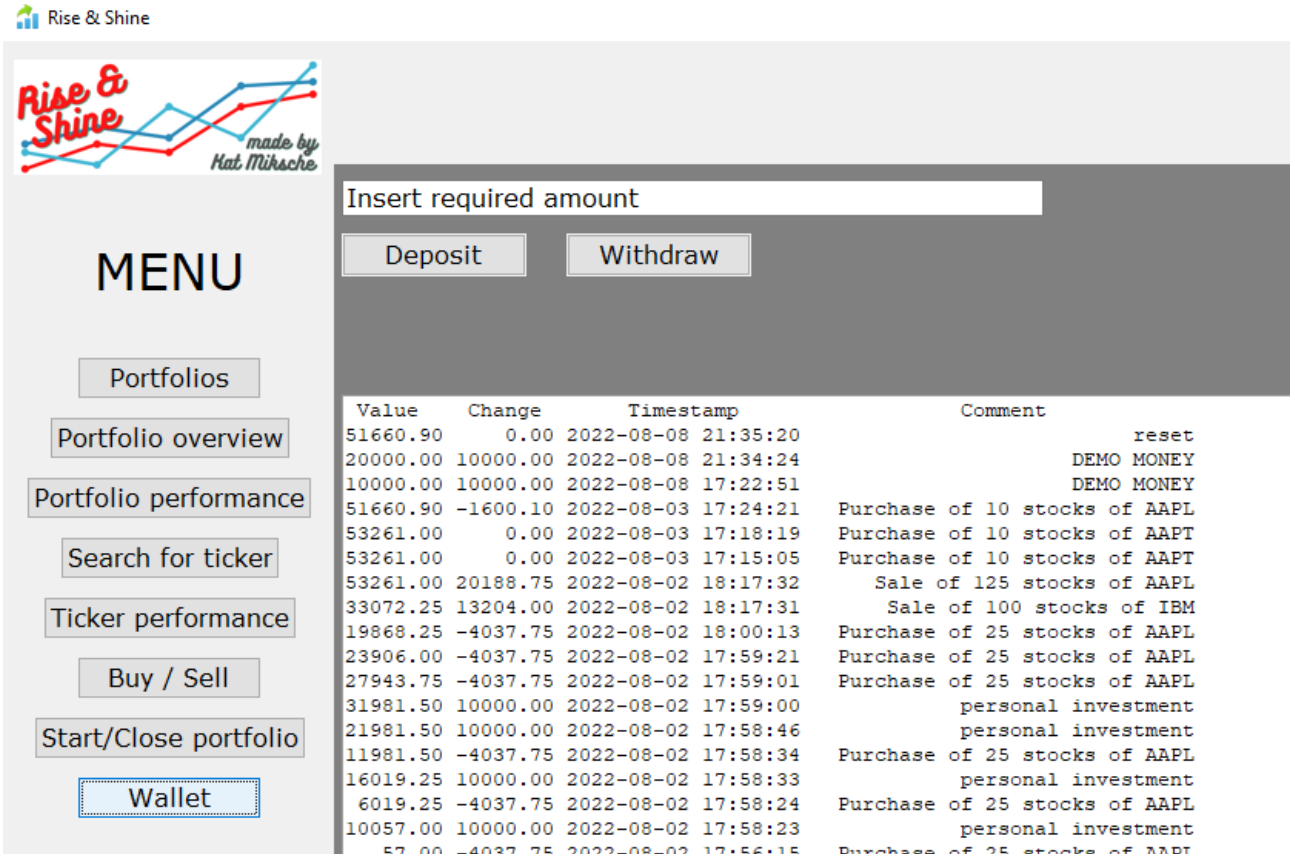
Buy / Sell tool



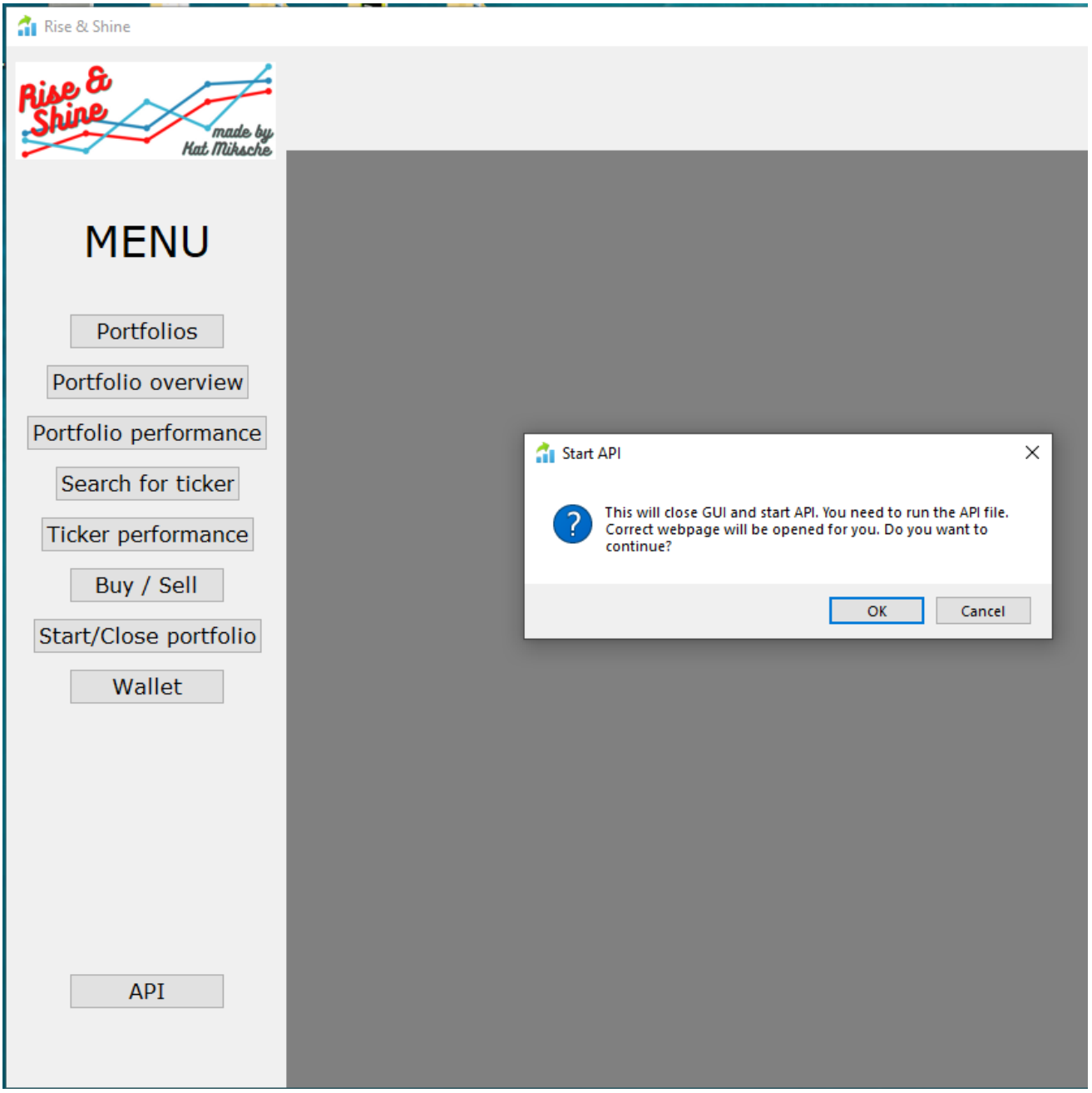
Start / Close portfolio



Wallet - Deposit / Withdrawal, see records

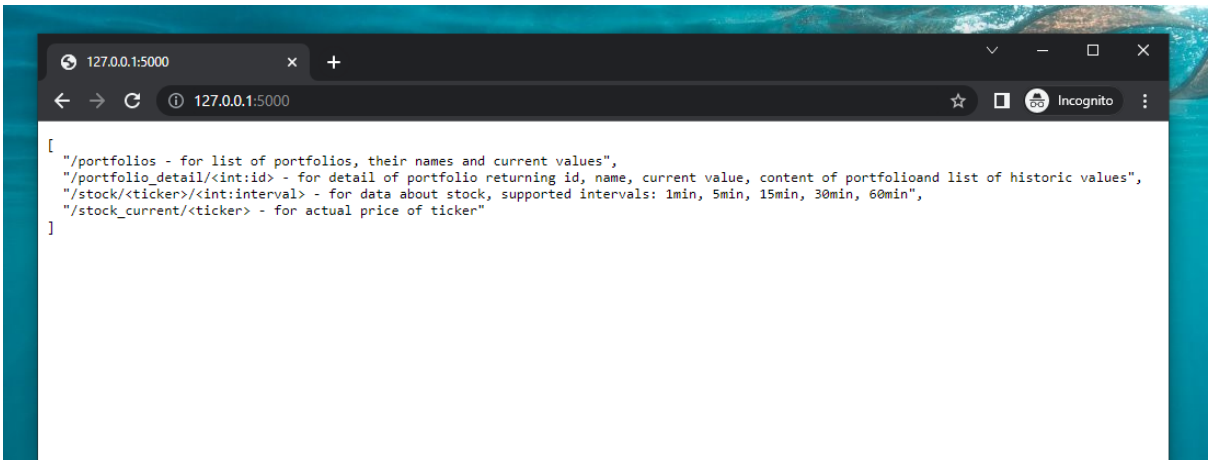


API - button will open API file and close GUI

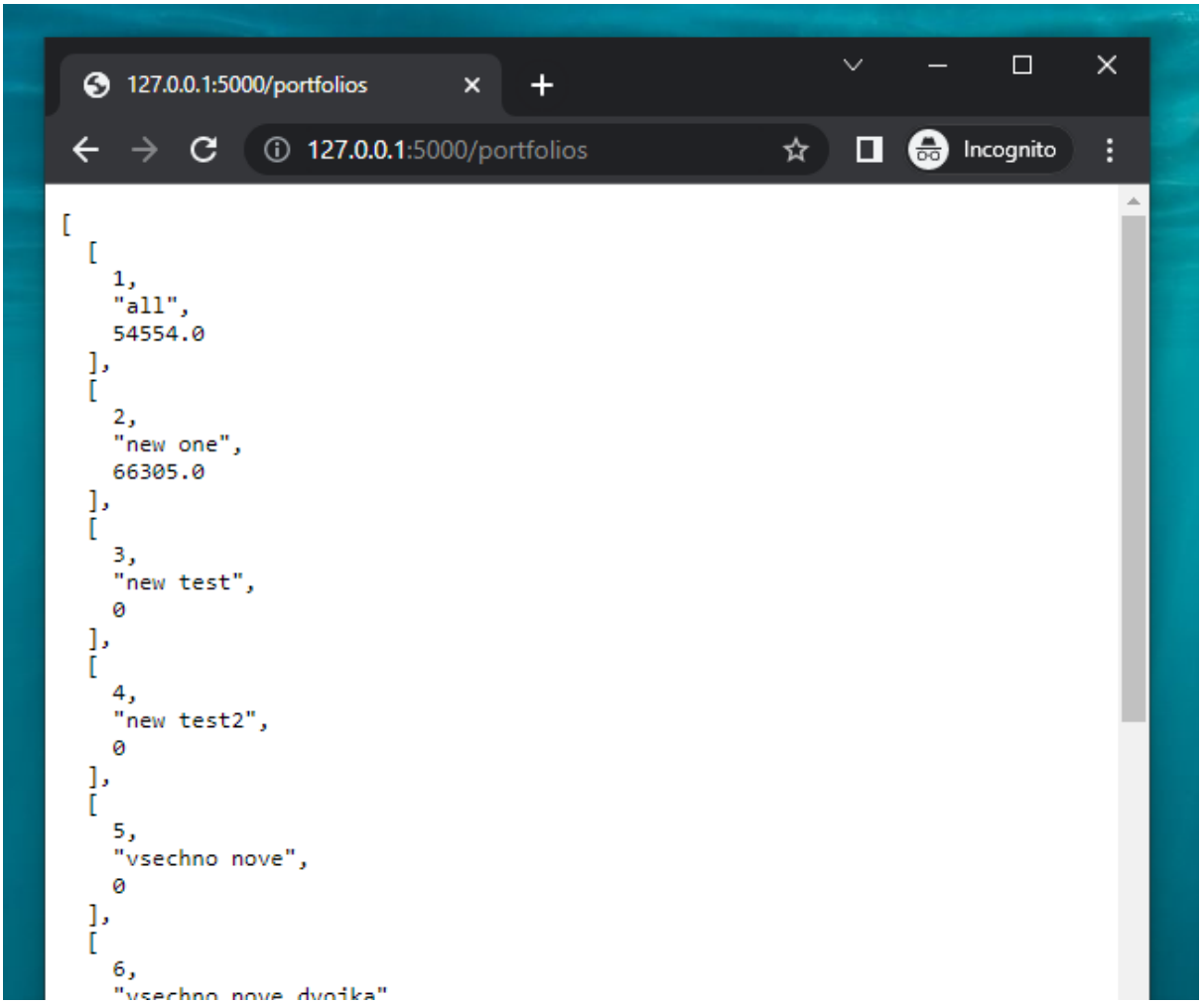


API

Home - http://127.0.0.1:5000/

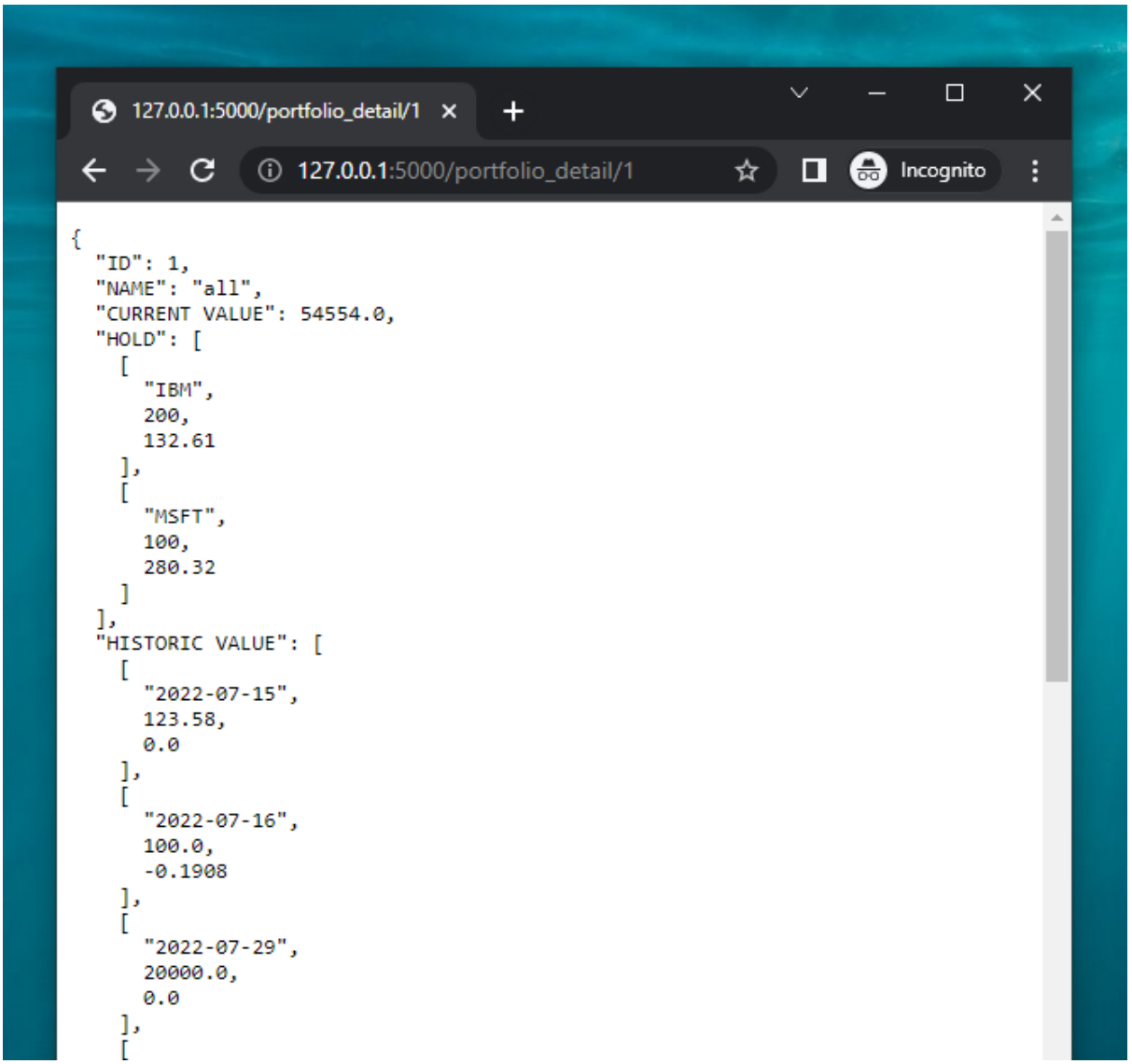


/portfolios - for list of portfolios, their names and current values



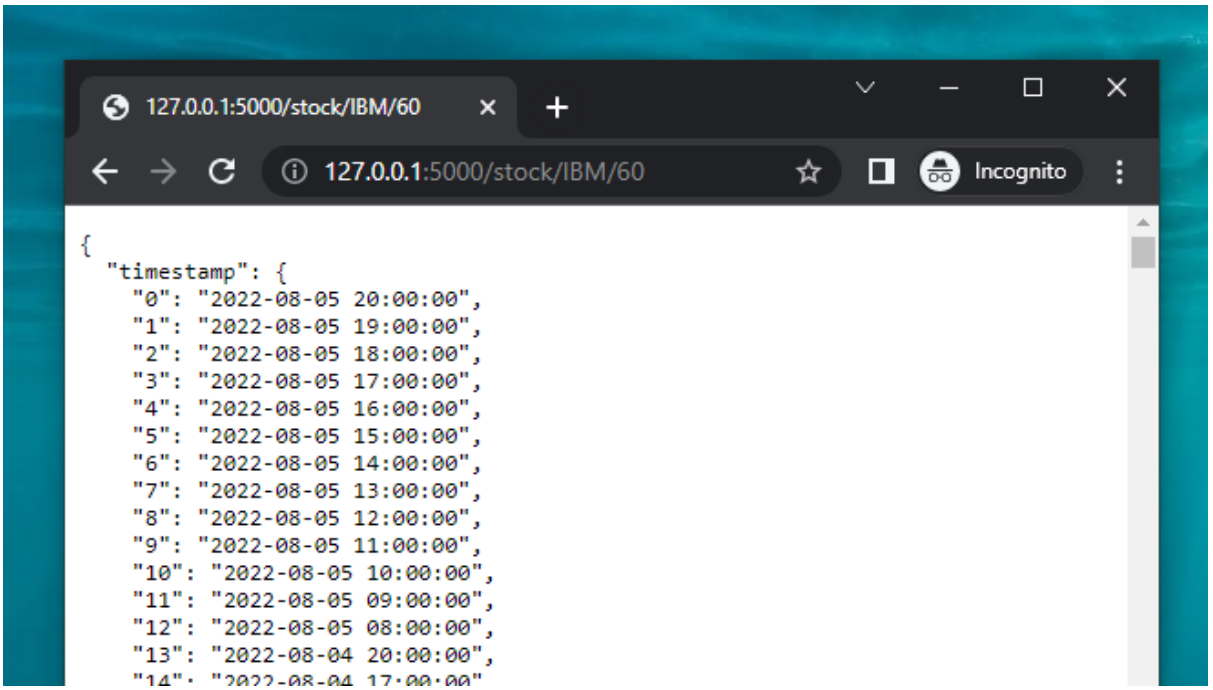
/portfolio_detail/<int:id> - for detail of portfolio

returning id, name, current value, content of portfolio and list of historic values

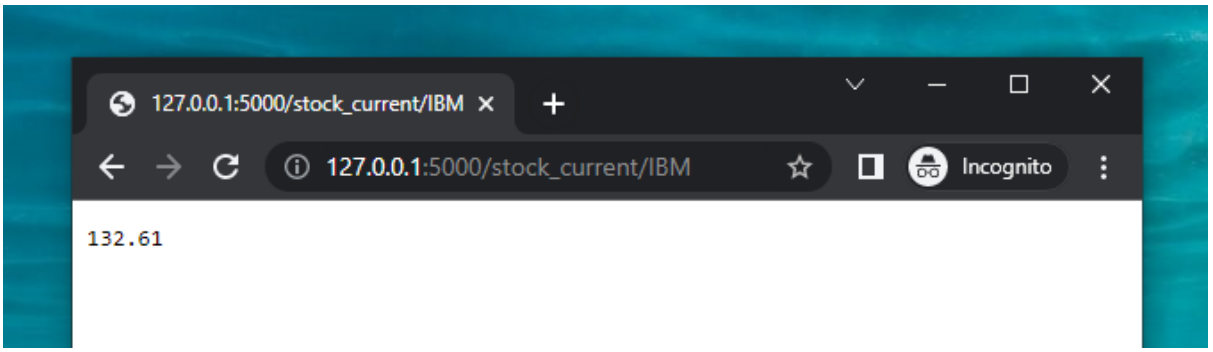


/stock/<ticker>/<int:interval> - for performance data about stock

supported intervals: 1min, 5min, 15min, 30min, 60min



/stock_current/<ticker> - for actual price of ticker



Technical specifications, used technologies & architecture

Application system requirements

- Browser (ideally Chrome)
- MySQL (any version)
- Python 3 including modules mysql, requests, io, os, csv, itertools, plotly, datetime, pandas, numpy, webbroser, PIL, tkinter and flask (for API - optional)

Installation

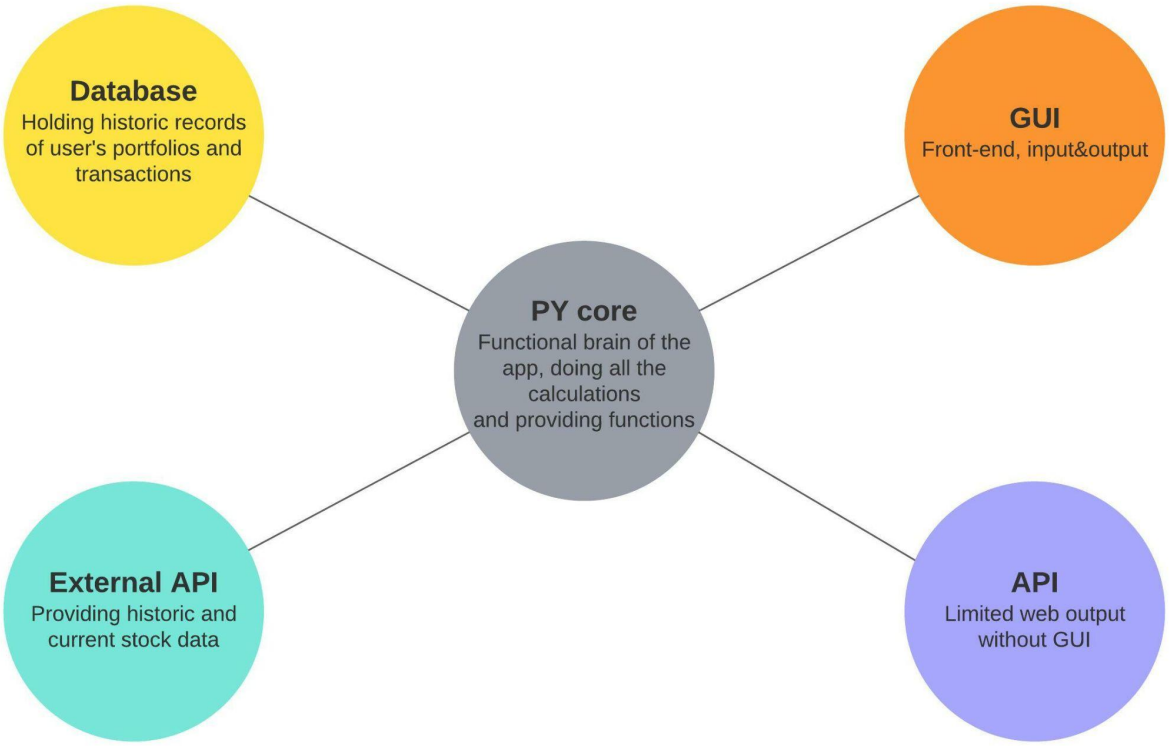
1. rise-shine-SQL-setup.sql - required
2. DEMO data.sql - optional
3. change sql connection config details in RSSQL.py
4. run tests in Test folder - optional
5. GUI.py - run the main file
6. API.py - optional for API use

Other used technologies during project

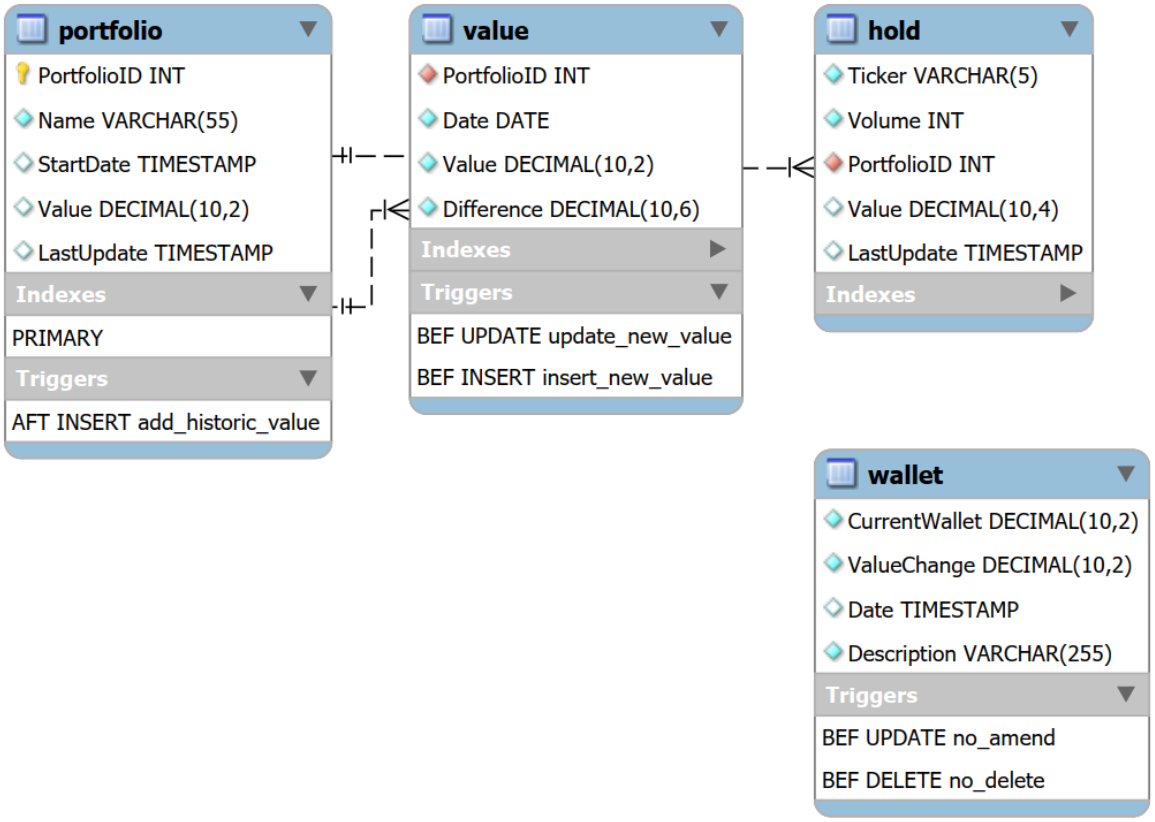
Application was developed in following environment:

- Asus G752VS, Intel i7--6820HK 2.7GHz, 32GB RAM
- Windows 10 Pro 64-bit
- MySQL Community Server - GPL 8.0.27 64-bit
- PyCharm 2022.1.1 (Community Edition) 64-bit
- Python 3.10.4 64-bit
- Google Chrome 104.0 64-bit
- ALPHA VANTAGE API - https://www.alphavantage.co

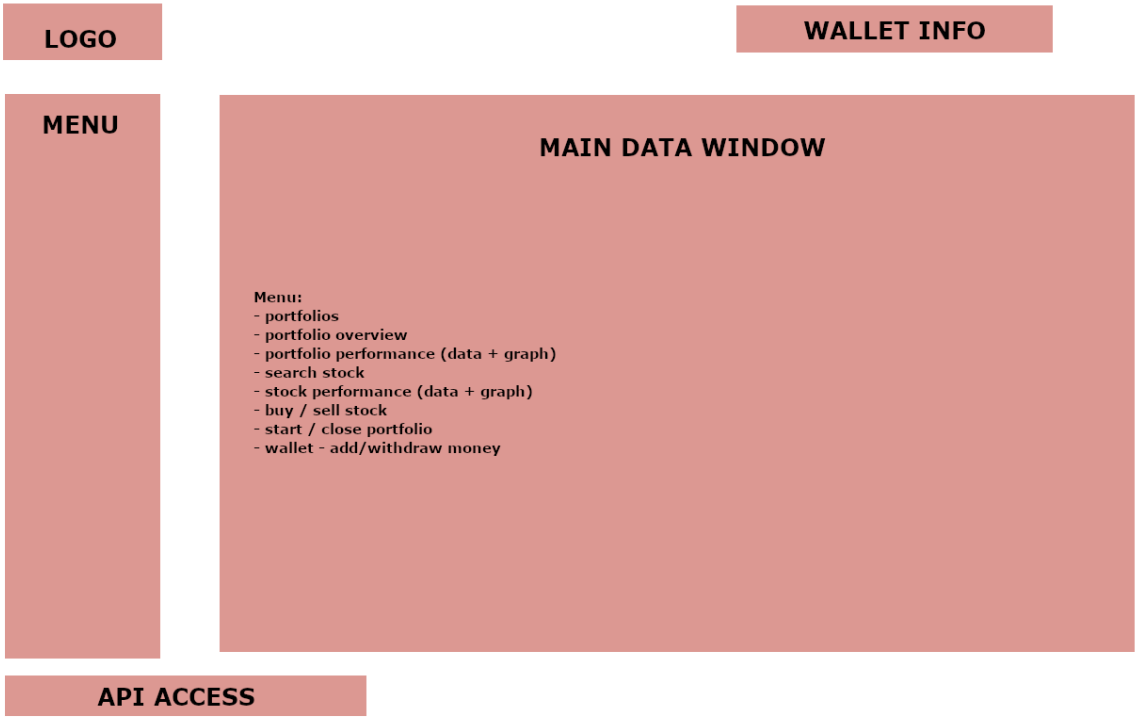
Application architecture



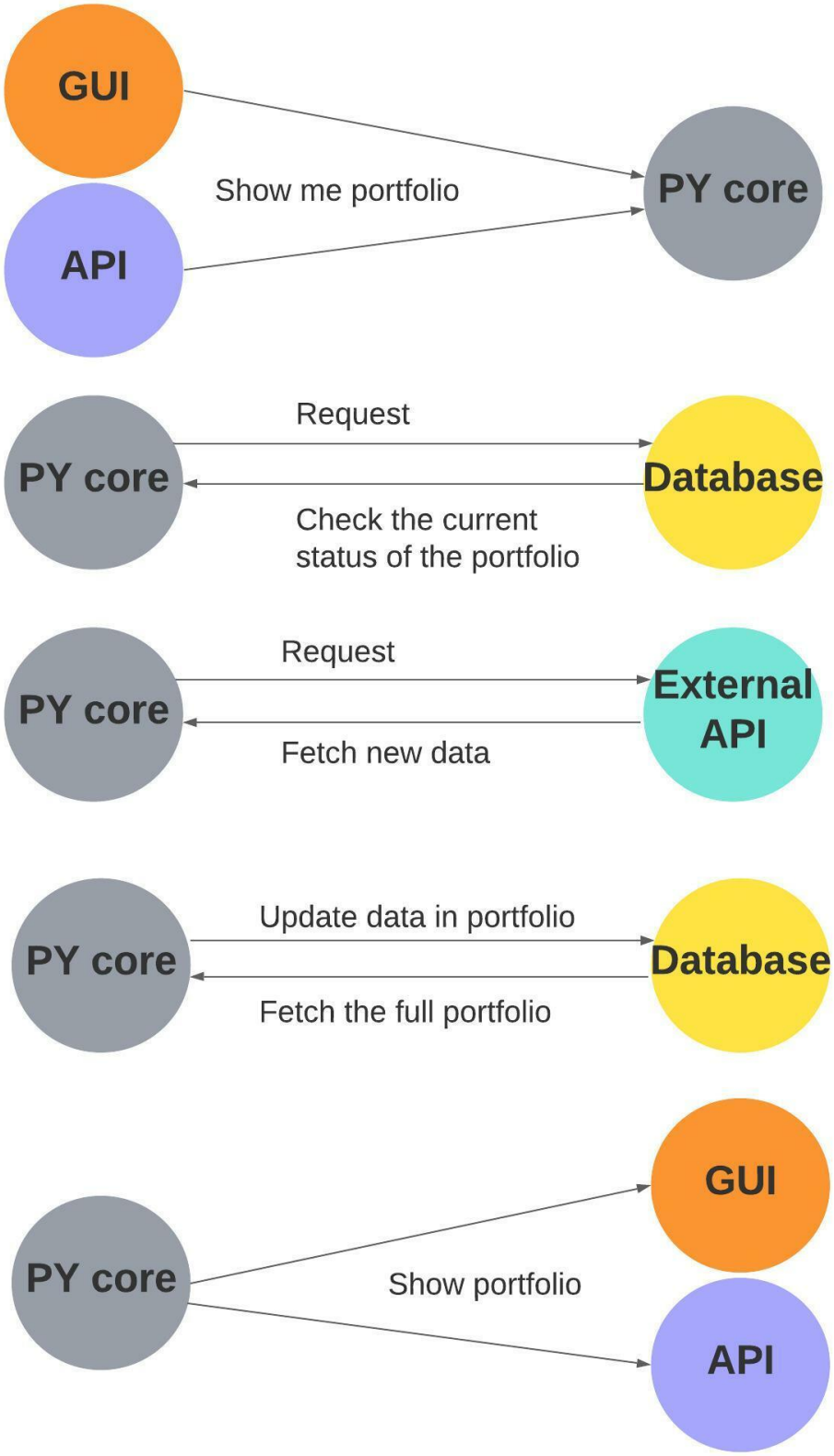
SQL architecture



GUI design



Example of function Show portfolio:



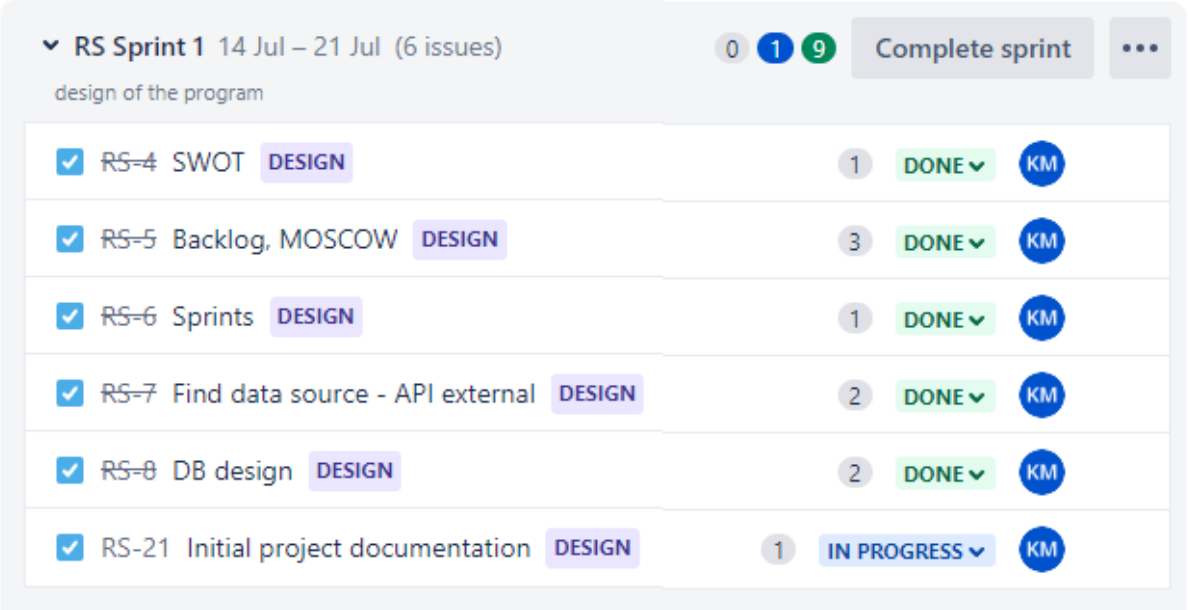
Development process

During project many other applications were used:
GitHub, Jira, Confluence, Gimp, Canva, Lucidspark, Google Docs, Slack, Zoom

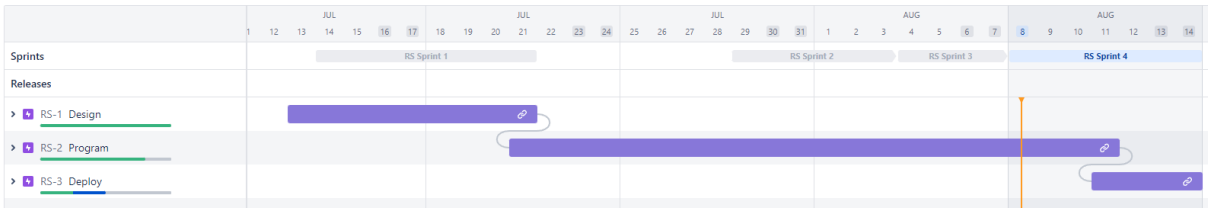
Planning of development was made in Jira.

Sprint 1

Planning and designing the application without any programming using agile methodology. SWOT analysis completed as first, then stories were added to backlog, evaluated by MOSCOW and assigned story points estimate. Further sprints were created, database architecture designed (not programmed) and initial documentation created.



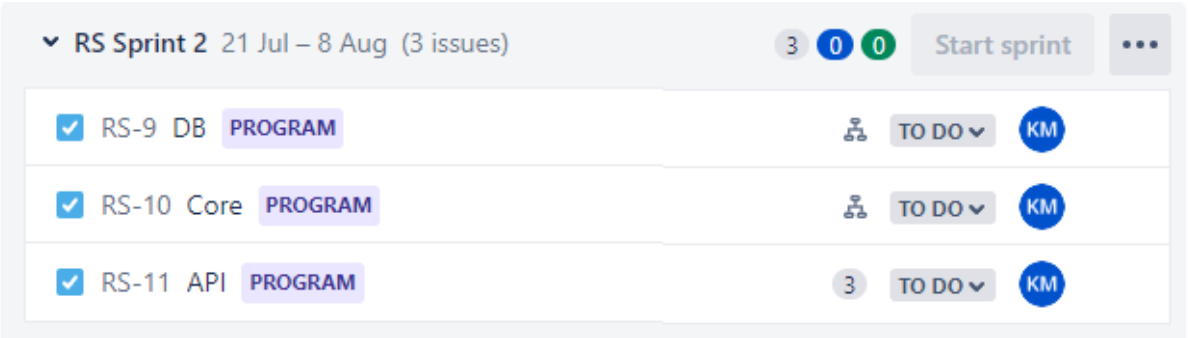
Roadmap created:



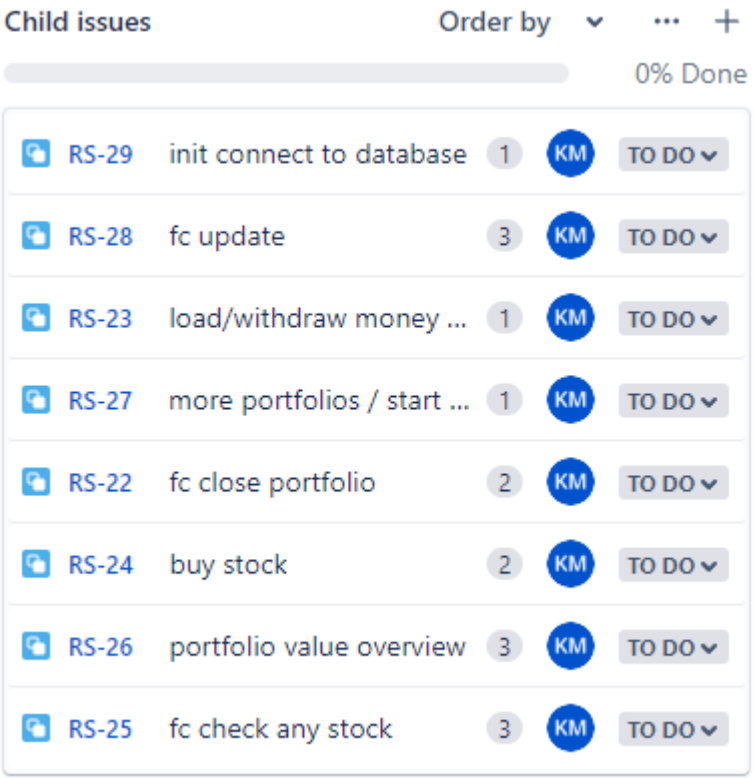
Sprint 2

Main programming part without GUI. First creating a database including triggers and functions. Then move to Python - creating functions for external API (exAPI.py) and classes including methods for wallet (wallet.py) and portfolio (portfolio.py). File RSSQL.py holds connection functions to database and configuration details. File Main.py administers a dictionary of portfolios and starts of new portfolios.

The most complicated function was the hidden one - update. It is updating historic values of the portfolio based on data for each held ticker from external API, so merging SQL and external API data through calculations.



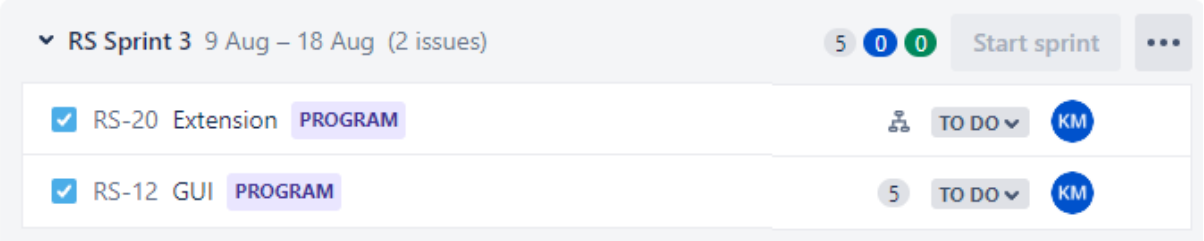
Children issues of Core:



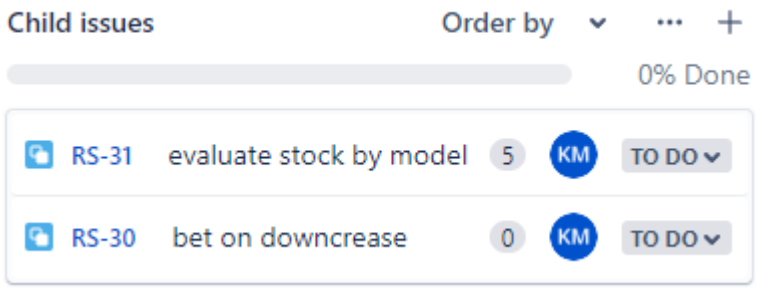
Sprint 3

Graphic user interface is made using tkinter. With almost 400 lines of code it proved to be a much bigger piece of work than originally planned. Apart from one option in the menu all other options require interaction with the user and therefore following functions called by buttons. Great focus was given to safeguard the program against any user error.

The delay led to reevaluation of sprint and moving extensions, which had MOSCOW value Could and Won't back to backlog. Hopefully in future new version of Rise & Shine will appear, where the functionality of stock being assessed by machine learning model will be available.



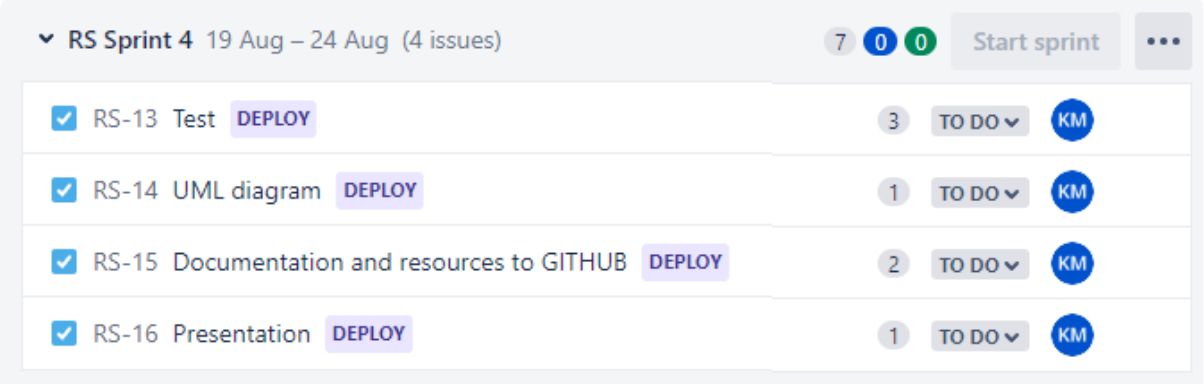
Children issues of Extension:



Sprint 4

Testing was done alongside the development, but after all parts finished tests were run again. Some issues occurred as small changes were made throughout the older code in terms of shaping it to the needs of GUI.
New SQL demo data was created as well.

SQL diagram was created and two UML diagrams from initial documentation were used in this document. As well, this document will be used at the final presentation of the project, together with the live presentation of the application.



Final words

This project took around 100 hours of screen time, where at least half of the time was used for learning and researching the solution. Test and fail was the most used method, there was no time for posting questions to online forums and waiting for any answer.

Where I started with basic knowledge of Python, minimum knowledge of optional modules and zero knowledge of tkinter, I am very proud of the result. This project took me completely from procedural programming (I was very comfortable with :-) to object oriented programming. I thank Code First Girls for the challenge, it was a tough one, therefore great!

Plan for the future is to come with a new release which will include data science tools for evaluating stocks. Then it can have that nice dodgy label “We don’t hold any responsibility upon results you receive” :-)