159.352 Assignment 1:

Weight: 30%

Deadline: April 24th 2022

In this assignment you will extend the minimalistic HTTP server (server3.py - we developed in the tutorials) to serve a password protected website which enables the user to manage their investment portfolio.

The conceptual framework of this web application is shown in Fig. 1. Your HTTP server will run in docker container hosted on Heroku (PaaS cloud). To service client requests, your server will use REST API to access data from the Investor Exchange (IEX) cloud. You will need to sign up for free developer accounts on IEX (https://iexcloud.io/cloud-login#/register/) and Heroku (https://www.heroku.com).

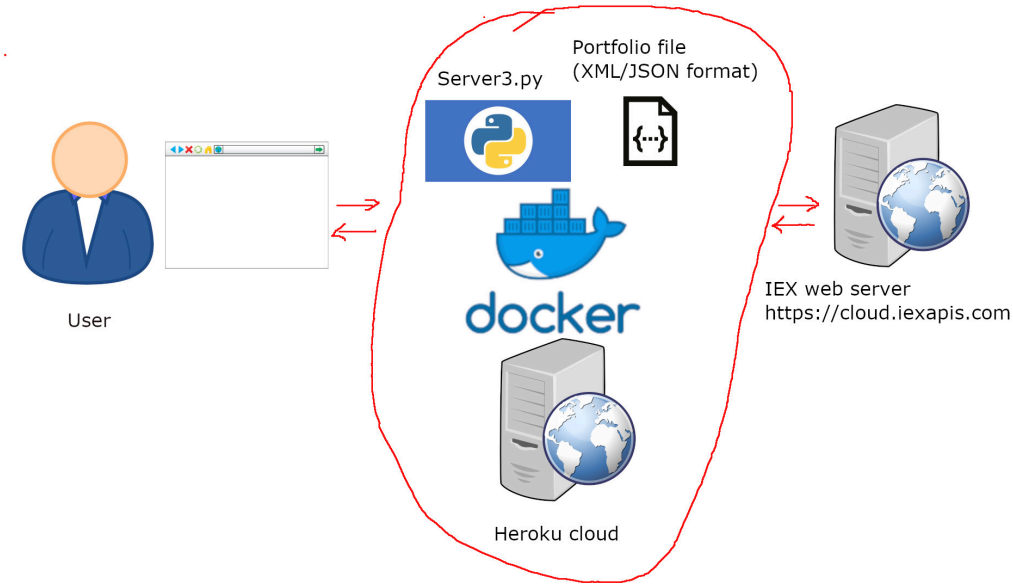


Fig. 1 Conceptual framework

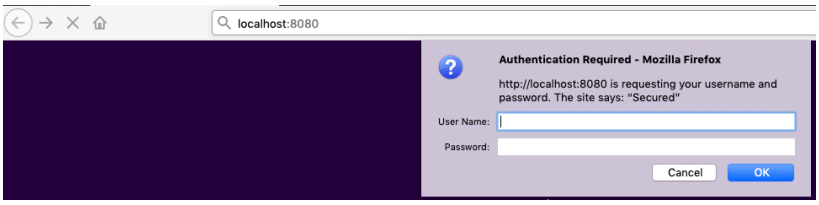
The objective of this assignment is to gain hands on experience with HTTP fundamentals, so strictly do not use any high-level frameworks (e.g. Servlets, flask, node.js, Django etc.,) since they abstract the low level HTTP functionality.

1) Authentication (6 marks)

Implement **basic access authentication** scheme in your python HTTP server such that without the correct login credentials none of the resources on the server should be accessible.

When the browser sends the initial GET request, the server should check for authentication credentials in the headers and respond with a suitable HTTP status code requesting for authentication credentials using the basic access authentication.

At this stage the browser would prompt the user to enter username and password



When the browser sends the authentication credentials, the server checks the headers and serves the requested content if the authentication credentials match. Set the authentication credentials for your website to be your student ID number:

username: <*your student ID number*>
password: <*your student ID number*>

Assume your website is dedicated for only one user, that is you.

[Hint: Revisit Part C of week 2 tutorial exercise to see how the authentication headers are exchanged between client and server]

2) Portfolio (10 marks)

When the user visits the url **/portfolio** your HTTP server should respond with a html page showing the user's investment portfolio, along with input boxes and command buttons to update the portfolio.

Stock	Quantity	Price	Gain/Loss
TSLA	100	$1000	-20%
AAPL	50	$150	12%

Stock symbol
Quantity
Price

reset update

The portfolio will be stored on your HTTP server as either XML or JSON file. For our assignment, the portfolio is made up of stocks only. For each stock the server needs to store the ticker symbol, the quantity of shares purchased (i.e. owned by the user), and the purchase price of a share. Work out a suitable data representation to store the portfolio information. You can choose either to work with XML or JSON. Name the file as portfolio.xml or portfolio.json.

In updating the portfolio file, the user can add a new stock, or reduce/remove an existing stock using negative quantity. You must validate the user inputs on the server side when updating portfolio file. Use HTTP POST method for updating the portfolio file.

To help the user input the correct symbol, use the following API endpoint to get the list of symbols. Note the JSON response from this API call contains all types of securities, so you will need to filter symbols for common stock (cs) type.

https://cloud.iexapis.com/stable/ref-data/symbols?token=***YourAPIToken***

make sure to substitute your API token in the above URL

Refer to API documentation for details; https://iexcloud.io/docs/api/#symbols

Compute the gain/loss percentage for each stock in the portfolio on the server side using the following API endpoint which fetches the latest quote for the given stock symbol.

Gain or loss = (latest quote – price) / price * 100

https://cloud.iexapis.com/stable/stock/***symbol***/quote?token=***YourAPIToken***

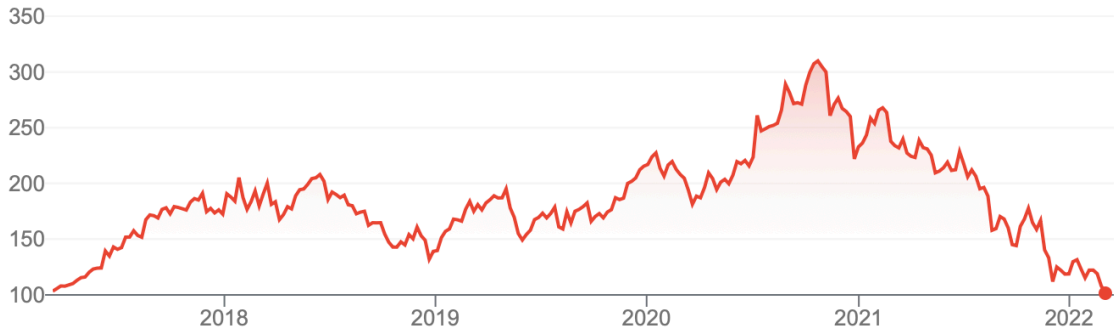
make sure to put the actual symbol (e.g TSLA) in the above URL

3) Stock research (7 marks)

When the user visits the url **/research** your HTTP server should respond with a html page containing an input boxes to enter the stock symbol. After entering the stock symbol, the page should show the following data about the stock, and display a graph of the daily closing price for five year period (5y)

Stock symbol

```
Symbol: BABA
Company Name: Alibaba Group Holding Ltd
PE ratio: 26.84
Market Capitalization: 290311742779
52 week high: 245.69
52 week low: 100.02
```



Use the same API endpoint discussed in the portfolio section to help the user enter the correct stock symbol.

Use the following Stats API endpoint to get the data about the stock.
https://cloud.iexapis.com/stable/stock/***symbol***/stats?token=**yourAPIToken**

Use the following API endpoint to extract the chart data for the given stock symbol.
https://cloud.iexapis.com/stable/stock/***symbol***/chart/5y?chartCloseOnly=true&token=***yourAPIToken***

You must plot the data on the client side (using any javascript library of your choice e.g. canvasJS).

4) Deployment (7 marks)

After you have finished developing and testing, deploy your web application in a docker container and host it on Heroku cloud.

Useful documentation

https://iexcloud.io/docs/api/#api-reference

https://docs.docker.com

https://devcenter.heroku.com

Submission

Complete and well-documented source files necessary to run your web application in docker. A README file containing instructions needed to run your web application. In your README file make sure to put down the URL of your web application hosted on Heroku. Submit it a in zipped file on Stream.

Here is a worked example to show how updates to the portfolio is recorded.
Suppose you start off with an empty portfolio stored on the server side.

Stock	Quantity	Price

When the user inputs

Stock symbol: AAPL

Quantity: 100

Price: 120

The portfolio is updated as follows:

Stock	Quantity	Price
AAPL	100	120

Now if the user buys 50 more shares of AAPL at $140 per share, she will enter the following input in the browser

Stock symbol: AAPL

Quantity: 50

Price: 140

The purchase price of each share held by the user is calculated as follows: (100*120 + 50*140)/150 = 126.67
The portfolio on the server will be updated as follows.

Stock	Quantity	Price
AAPL	150	126.67

Suppose now the users reduces their position in AAPL by 75 shares:

Stock symbol: AAPL

Quantity: -75

Note when the quantity is negative, the purchase price is not applicable because she is selling the shares.
The portfolio is updated as follows:

Stock	Quantity	Price
AAPL	75	126.67

The user should not be allowed to sell more shares than they own (i.e. short selling). If the user attempts it they should be given an error message, and the portfolio should not get updated.

Stock symbol: AAPL

Quantity: -100

<error short selling not allowed>

Stock	Quantity	Price
AAPL	75	126.67

If the user sells all the shares they hold, then the corresponding position should be removed.

Stock symbol: AAPL

Quantity: -75

Stock	Quantity	Price

Note: the gain/loss column is computed on the fly based on the current quote. There is no need to store gain/loss field on the server.