Deree
School of Graduate and Professional Education

# Big Data Architectures
# Winter 2025

## Term Project

**Due: Friday April 4, 2025, 11:59.**

This project will setup a network of financial and investment services of stock exchanges and investors as well as applications that analyze those data. The components of the network are described below.

**Two Stock Exchange servers**. Each server trades 12 stocks and emits JSON objects to Kafka at the close of each day. Stock Exchanges trade every day except weekends and national holidays.

Consider the Stock Exchange server in file server.py. The server simulates the functions of a fictious Stock Exchange and emits JSON objects to port 9999 at the closing of each day. You need to modify the server to emit the JSON objects to a Kafka topic "StockExchange" for every day since 1-1-2000, except weekends and Greek national holidays. The duration of each day will be simulated with at least 2 seconds of sleep, i.e., the server will emit no earlier than every 2 seconds the closing stock prices for a date. One server should emit the first 12 stocks of those shown in server.py and the other will emit the rest. The two servers should be implemented in files se1_server.py and se2_server.py, respectively.

**Three Institutional Investors** Inv1, Inv2, and Inv3. Each investor manages two portfolios as follows: Inv1 manages P11 and P12, Inv2 manages P21 and P22, and Inv3 manages P31 and P32. The portfolios are structured as follows:

| Inv1 | | | Inv2 | | | Inv3 | | |
|---|---|---|---|---|---|---|---|---|
| Portfolio | Stock | Qty | Portfolio | Stock | Qty | Portfolio | Stock | Qty |
| P11 | IBM | 1300 | P21 | HPQ | 1600 | P31 | HPQ | 2200 |
| | AAPL | 2200 | | CSCO | 1700 | | ZM | 1800 |
| | FB | 1900 | | ZM | 1900 | | DELL | 2400 |
| | AMZN | 2500 | | QCOM | 2100 | | NVDA | 1200 |
| | GOOG | 1900 | | ADBE | 2800 | | IBM | 1900 |
| | AVGO | 2400 | | VZ | 1700 | | INTC | 1600 |
| P12 | VZ | 2900 | P22 | TXN | 1400 | P32 | VZ | 1800 |
| | INTC | 2600 | | CRM | 2600 | | AVGO | 2900 |
| | AMD | 2100 | | AVGO | 1700 | | NVDA | 1600 |
| | MSFT | 1200 | | NVDA | 1800 | | AAPL | 2200 |
| | DELL | 2700 | | MSTR | 2600 | | DELL | 2500 |
| | ORCL | 1200 | | MSI | 1800 | | ORCL | 2000 |

Each investor reads from the Kafka topic "StockExchange" and

1. Evaluates each of its portfolios for each day.
2. Writes to a Kafka topic 'portfolios' a JSON object that contains

a. the evaluation of the portfolio for each day,
b. difference from the previous day's evaluation,
c. percentage difference from previous day's evaluation,

as well as any other necessary information.

The investors should be implemented in files inv1.py, inv2.py, inv3.py, respectively.

**An application**, which creates a MySQL database 'InvestorsDB' with the following tables

1. 'Investors' with fields 'Id', 'Name', 'City'.
2. 'Portfolios' with fields 'Id, 'Name', 'Cumulative'.
3. 'Investors_Portfolios' with fields 'iid', 'pid' that connects investors and portfolios.
4. <investor>_<portfolio> per investor and portfolio, e.g., Inv1_P11, Inv12_P12, and so on. The content of each file should be similar to the following.

| As Of | NAV per Share | Daily NAV Change | Daily NAV Change % |
|---|---|---|---|
| 03-Mar-23 | 51.47 | 1.25 | 2.49 |
| 02-Mar-23 | 50.22 | -0.44 | -0.87 |
| 01-Mar-23 | 50.66 | 0.06 | 0.12 |
| 28-Feb-23 | 50.6 | -0.43 | -0.84 |
| 27-Feb-23 | 51.03 | 0.39 | 0.77 |

and initializes the first three tables with the data shown above. (Field 'Cumulative" is of type Boolean; mutual funds we model in this project are ither Cumulative or Distributing). The application should be implemented in file investorsDB.py.

**An application** that reads the Kafka topic 'portfolios' and adds data to the 'InvestorsDB' database, as new JSON objects are read from topic 'portfolios'. The application should be implemented in file app1.py.

**A Spark DF application** that queries the 'InvestorsDB' database and for a given investor produces a file for each of its portfolios with name <investor>_<portfolio>_stats that contains the following.

- The maximum and minimum daily change and percentage change for the full history of the portfolio.
- The maximum and minimum daily change and percentage change for the full history of the portfolio per year.
- The average evaluation and the standard deviation of evaluation for the full history of the portfolio.
- The average evaluation and standard deviation of the portfolio for a given period, e.g., 2020 to 2024.
- The average evaluation per month for the full history of the portfolio, with the most recent month listed first.

The application should be implemented in file app2.py.

# Big Data Architectures
# Winter 2025

**Submission**

Each team should submit

1. All Python code.
2. A Powerpoint presentation containing
    a. A short description of the problem.
    b. The architecture of the solution.
    c. The design decisions made.
    d. Screenshots of the running application.
    e. Lessons learnt.

    The presentation should not be more than 15 slides excluding the first one that should contains the name and logo of the team and the names of its members, and the last one that should thank and prompt for questions.
3. A file Team.txt containing a clear statement stating which parts of the application where implemented by which team member and a self-assessment of the percentage of contribution of each team member to the project (e.g., Efthimiou 45%, Papadopoulou 55%).
4. A README.txt file with instructions on how to run the network of applications.

To submit your work add

- all source code files,
- the Powerpoint presentation,
- the file Team.txt
- the file README.txt

into a rar or zip compressed file with name <Lastname>_<Lastname>.rar or .zip (e.g., Efthimiou_Papadopoulou.zip) and submit it to Blackboard. Names must appear in alphabetical order.

**Penalties**

Violation of any naming conventions will result in 30 points reduced from your grade.

Submission of unnecessary files will result in 20 points reduced from your grade.