Software Design and Architecture
Homework 1

The goal of the homework assignments is to implement a database-based web application using different software architecture styles and applying the concepts covered in the lectures.

The application should focus on stock market analysis related to the **Macedonian Stock Exchange** and will involve **historical daily stock market data** for all available issuers (companies or financial institutions). The data processed should cover at least the **last 10 years on a daily basis**.

In this first homework, you will focus on data processing using the **Pipe and Filter architecture**. Specifically, you will need to **automate the process** of downloading and transforming stock market data for each issuer, ensuring that only the necessary information is retained and the data is correctly formatted for further analysis. The following tasks will be required:

1. Form your team (**max 3 members**).
2. Write a **short project description** (about half a page long).
3. Provide a **specification of the functional and non-functional requirements of the application**. Denote which requirements are functional and which are non-functional. Include **user scenarios**, **personas** and a **descriptive narrative** to illustrate these requirements.
4. Download the data you'll use and populate your database[1] with it. To do this, you should use the pipe-and-filter dataflow architecture. You'll need to define multiple simple, independent functions (filters) to transform the data and combine them in such a way that the output of each filter can be used as the input of another (the filters should be connected in a pipe). The goal of these filters and the entire pipe should be to transform the data from the raw format you downloaded them in to a format appropriate for input into a database.

   The necessary transformations should include the following steps:

   (a) **Filter 1: Automatically retrieve all issuers listed on the Macedonian Stock Exchange website**
   - Open the Macedonian Stock Exchange page for historical data and **automatically extract the list of all issuers** from the dropdown (excluding bonds or any codes that contain numbers).
   - This filter will programmatically gather the codes for each issuer listed, **without manual intervention**, ensuring that all valid issuers are captured.

   (b) **Filter 2: Check the last date of available data**
   - For each issuer code retrieved in the first filter, check your database (or structured file) to see **up to which date data has already been fetched for that issuer**.
   - If there is no existing data, fetch data for at least the last 10 years.
   - If data exists, identify the last recorded date and pass this information to the next filter.

   (c) **Filter 3: Fill in missing data**
   - For each issuer, use **the code and the last available date (from the second filter)** to retrieve any missing data up to the present date.

[1] If you aren't confident in your database skills, you can also use a structured file (CSV, Excel, JSON) or another non-relational approach to store the data.

- Ensure all new data is properly added or combined with existing data to the database (or structured file).
- Ensure all dates are correctly formatted in a consistent format (Tip: use the English version of the data to avoid inconsistencies in date formatting).
- Prices should be formatted with proper delimiters (comma for thousands and period for decimals, such as 21,600.00).

At the end of this process, the application will store the complete and up-to-date stock data in the database, merging it with previously recorded data and ensuring all formatting is accurate.

You can create multiple pipes if you're using more than one data source. If you're using multiple pipes, you should use the same filters for the same transformation in both pipes instead of creating new ones.

5. **Maximum Points Challenge**

   - Implement a timer to measure how long it takes for your application to fully populate an empty database with the stock data. Optimize this process for maximum speed and efficiency.
   - Your team's score will be scaled relative to the fastest time achieved. The best-performing solution will set the benchmark, with other scores normalized against it. Faster and more efficient solutions will receive the highest scores.

You can freely choose the programming language, database and operating system. Python is recommended for the next homeworks due to the availability of libraries suited for data transformation and financial calculations. **The project has to be turned in through GitHub (public repository). Project submissions sent by e-mail or uploaded to GitHub as a .zip file won't be accepted.**

Your **public GitHub repository** should contain the documents with the project description and requirements specification, the code used to transform and input the data in the database, and the raw (unprocessed) data used in the project. If you choose to use a structured file for storage, be sure to upload the processed data as well. All files should be in a folder called **Homework 1**.

The deadline for submitting the first homework is **November 10, 2024**. Please note that this deadline is **final and will not be extended under any circumstances**.