Software Design and Architecture
Homework assignment 3

For the **third homework assignment**, you will need to **implement all functional and non-functional requirements of your application**. The code doesn't need to have high quality and it doesn't need to be refactored. At the end you should have a **web application that fills all requirements defined in the SRS document** you turned in for the first homework assignment.

In this homework assignment, you will need to implement **three different analyses** in your application. Each type of analysis has a distinct focus and methodology for data processing, and their successful implementation directly impacts the maximum grade you can achieve:

1. **Technical Analysis (mandatory)** - this analysis is fundamental to the assignment and is **mandatory for all students**. Implementing the technical analysis will earn you grades **from 6 to 7**.

   The technical analysis focuses on studying historical stock price and trading volume data to identify trends and patterns, which can help predict future price movements. For this analysis, you need to implement and analyze various technical indicators:

   - **Oscillators and Moving Averages (MA)**: Oscillators and Moving Averages are technical indicators used to analyze historical price and trading volume data to predict future market movements and make buy or sell decisions.
   - **Select the top 10 technical indicators (5 from the oscillators and 5 from the Moving Averages)**. For each of these indicators, calculate their values based on historical stock price data.
   - Using these indicators, you must **generate signals for the best times to buy or sell stocks**, i.e., when stock prices reach critical decision points (buy, sell, hold).
   - For each of the selected 10 technical indicators, calculate their values on **three different time periods: 1 day, 1 week and 1 month**. This will give you insights into short-term, medium-term, and long-term market trends.

   If you are using Python as your programming language, the `pandas` library offers convenient functions for calculating the selected indicators. For example, to calculate Moving Averages (SMA and EMA), you can use the methods `rolling().mean()` for SMA and `ewm().mean()` for EMA. For oscillators such as RSI, there are libraries like `ta` (Technical Analysis Library) that can help you implement these indicators. These tools will enable fast processing and analysis of data.

   For an example of technical analysis, use the stock chart for a given issuer, available at the following link: `https://www.tradingview.com/chart/?symbol=NASDAQ%3ATSLA` (in this case Tesla). In the chart's side panel, under the Technicals section, click on More technicals to open the technical indicators.

2. **Fundamental Analysis** - adding and implementing this analysis, alongside the technical analysis, will earn you grades ranging **from 7 to 8**.

   The fundamental analysis focuses on evaluating companies by reviewing their financial data, as well as information derived from public news and reports. In this analysis, you will need to evaluate the sentiment of news related to company stocks:

   - Use **Natural Language Processing (NLP)** techniques to determine **whether the news about companies is positive or negative**.

- Perform an analysis of the news published by companies, calculate the **positive or negative sentiment factors**, and use them to decide **whether the stock prices of a given company will rise or fall**.
- If the news is positive, the recommendation will be to buy stocks, whereas if it is negative, the recommendation will be to sell stocks.

For this analysis, you need to retrieve news related to each issuer from the Macedonian Stock Exchange website that are relevant for monitoring their financial performance and stock trends.

3. **LSTM Stock Price Prediction** - if you implement all three analyses (technical, fundamental, and LSTM prediction), you can earn grades ranging **from 9 to 10**.

   In this analysis, you will focus on using **machine learning to predict future stock prices** through **LSTM (Long Short-Term Memory) networks**. This method is specifically designed for time series analysis, making it ideal for forecasting stock price trends:

   - **Training and validation of the model**: You will need to prepare and train the LSTM network for price prediction using the historical stock price data. Split the data into two parts: **70% for training the model and 30% for validation**.
   - **Price prediction**: Using the prepared LSTM network, generate **predictions for future stock prices based on historical data**.
   - **Model testing and evaluation**: Test the model with different parameters and analyze how accurately it predicts price movements (using metrics like Mean Squared Error or Root Mean Squared Error).

The application must be tested and **function without errors**. If you wish, you can start applying software patterns, but keep in mind that this may complicate the work for the fourth homework assignment. At this stage, the quality of the code will not be considered during evaluation.

Upload the code for this task to GitHub in a folder named Homework 3. It is mandatory to update the code regularly (push) throughout the development process, rather than uploading all the code at once at the end. Note that the contribution of each team member will also be evaluated based on who wrote each part of the code individually.

You are also required to create a **screen recording demonstrating the functionality of the application**, with a maximum duration of **5 minutes**. The recording should showcase all functional and non-functional requirements listed in the SRS document from the first homework assignment. **Any requirements not shown in the video will be considered as not implemented.** The recording should be uploaded to GitHub along with the code.

During development, it's recommended that you create a new branch for every feature (or group of related features). You workflow could look something like this:

1. update your local repository with `git pull origin master`
2. create the branch you'll be working on `git checkout -b branchName`. branchName should clearly say what's being implemented in this branch.
3. make changes to the code, add and save them often with `git add` and `git commit`. Run git commit locally every time you finish your work for the day, and it's recommended you do it even more often. The commit messages should clearly describe what you've changed in that commit.
4. publish your changes with `git push origin branchName` when you're done implementing the feature you're woirking on (you can do this more often as well).
5. when you finish working on that branch, you can merge it with the master branch, but it would be better to open a pull request so the other team memmbers can look over your changes and discuess them before you merge. You can open a new pull request on GitHub by clicking on Pull requests, and then New pull request.

6. to get someone else's remote branch locally and make changes to it, run `git fetch` followed by `git checkout branchName`
7. return to the master branch using `git checkout master`.
8. `git pull` to get the newest version of the master branch.
9. `git pull origin branchName` (if you're merging a branch you haven't worked on/don't have locally) or, if you have the newest version of the branch locally, you can also run `git merge branchName` to merge the branchName and master branches.
10. `git push` to publish your changes to GitHub

You can read more about this git workflow at `https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow`. Another workflow that can be especially useful for keeping separate development and production code can be found at `https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow`.

The deadline for submitting the third homework is **December 29, 2024**. Please note that this deadline is **final and will not be extended under any circumstances**.