Software Requirements Specification

Stock Market Analysis Application for Macedonian Stock Exchange (MSE)

Marko Minovski
Nikola Janev
Ermal Baki

December 4th, 2024
Revision 2

1. Introduction

1.1 Purpose

The purpose of this project is to develop a web application that automates data acquisition and transformation for stock market data from the Macedonian Stock Exchange (MSE). The data will span at least the last 10 years, updated daily for all available issuers. Using the Pipe and Filter architectural pattern, the application will retrieve, transform, and store stock data, preparing it for further analysis.

1.2 Scope

The application will:

- Collect daily stock data for issuers listed on the MSE, excluding bonds and certain numerical codes.
- Use a series of independent filters to transform and standardize the data for database storage.
- Perform automated updates to maintain up-to-date historical data, allowing for efficient future analysis.

1.3 Definitions, Acronyms, and Abbreviations

- **MSE**: Macedonian Stock Exchange.
- **Pipe and Filter**: An architectural pattern where data flows through a series of processing filters.
- **Issuer**: A company or financial institution whose stock is traded on the MSE.
- **Database**: A database is an organized system for storing, managing, and retrieving data efficiently.

2. Overall Description

This project focuses on developing a data processing and analysis application tailored to the Macedonian Stock Exchange (MSE). The application's primary goal is to provide users with reliable, up-to-date historical stock data on all available issuers (companies and financial institutions) listed on the MSE over the past decade. Built using the Pipe and Filter architectural style, the application automates the processes of data retrieval, transformation, and storage, allowing users to focus on analysis and decision-making rather than data management.

The application operates in multiple stages. First, it retrieves a comprehensive list of valid issuers from the MSE's historical data page, excluding irrelevant items like bonds or numeric codes. Next, it checks existing database records to determine the last date for which data is already available, fetching new data only as needed to maintain an unbroken timeline. Any missing or updated data is then seamlessly integrated, ensuring the database is both complete and current. During this process, a series of transformations are applied to standardize date formats and price representations, making the data ready for analytical tools and reducing inconsistencies.

Designed with efficiency in mind, the application also includes a timer that measures the time taken to populate an empty database with the entire dataset. This performance metric encourages optimization, ensuring the application runs quickly and can handle future data updates effectively. In addition, the application's modular design supports the use of multiple data sources, should they be required, with reusable filters enhancing consistency across datasets.

Overall, this project delivers an essential tool for financial analysts, researchers, and institutions interested in the Macedonian stock market, providing clean, structured data to facilitate informed financial analysis and forecasting.

3. Requirements

Priority level	Description
Priority 1	Necessary for system functionality
Priority 2	Beneficial functionality
Priority 3	Other features

3.1 Functional Requirements

- The system shall retrieve a list of issuers from the MSE historical data page. *Priority 1*
- The system shall exclude bonds or codes containing numbers. *Priority 1*
- The system will check if each issuer is present in the database. *Priority 1*
- The system will download any missing data for each issuer in the last 10 years. *Priority 1*
- The system will integrate any new data with the existing records for each issuer. *Priority 1*
- The system will use standard date formats to ensure consistency *Priority 1*
- The system will use proper price formatting with delimiters. *Priority 1*
- The system will be divided into three separate components – the frontend, backend and database client component. *Priority 1*
- The frontend component shall display the User Interface to the clients.
    - The frontend component shall generate a list of selectable tickers by fetching the "/all" endpoint.
    - The frontend shall generate a rudimentary table of information for a specific ticker by fetching the "/tickers/{ticker_code}" endpoint, where {ticker_code} represents the code of the ticker.
    - The frontend component shall display the results of Oscillators and Moving Averages technical analysis for a selected ticker.
    - The frontend component shall provide a range of information for which it requires technical analysis by sending a POST request with the parameters "tickerCode, fromDate and toDate" to the API endpoint "/tickers/analyze".
- The backend component shall include the web scraper and provide a callable API for the frontend component.
    - The backend component shall respond to "/all" GET requests with an array of json objects containing two fields, the code of the ticker and the latest date for which the database has information.

- The backend component shall respond to “/tickers/{ticker_code}” GET requests with an array of json objects all corresponding to a row scraped from the https://www.mse.mk website.
- The backend component shall, in the event of outdated information, call the scraper module to fill in the remaining missing data (Up to today).
- The backend component shall respond to “/tickers/analyze” POST requests by analyzing the necessary interval of information using the Pandas library.

- The database client shall maintain a connection to the MongoDB Atlas database.

3.2 Non-Functional Requirements

The system shall be optimized to retrieve, transform and store data at minimum in 20 seconds. *Priority 2*

The system will follow standard practices to insure future modifications. *Priority 2*

The system should be able to handle data expansion for additional issuers and dates. *Priority 1*

The system shall maintain a consistent connection to the database. *Priority 1*

The system shall sanitize all API endpoints to prevent abuse. *Priority 1*

4. User Scenarios

1. **Persona: Financial Analyst**
   - **Scenario**: The analyst wants to view historical data for all issuers to conduct a comparative stock analysis.
   - **Actions**: Logs in to the application, selects data for each issuer, and exports data for further analysis.
2. **Persona: Individual Investor**
   - **Scenario**: An individual investor wants to perform a simple analysis on stock performance before deciding on investments.
   - **Actions**: Logs in, selects a few issuers of interest, and views or downloads recent data to analyze stock volatility and trends in a spreadsheet.
3. **Persona: Market Researcher**
   - **Scenario**: The researcher is gathering long-term stock performance data for a report on industry growth trends in Macedonia.
   - **Actions**: Opens the application, retrieves and selects the last ten years of data for all issuers, and exports the data in CSV format for use in statistical analysis software.
4. **Persona: Economist**
   - **Scenario**: The economist wants to evaluate the economic impact of stock performance across various sectors over the past decade.

- **Actions**: Uses the application to filter issuers by sector, downloads historical data for all relevant issuers, and creates aggregated reports on sector trends.

5. Descriptive Narrative

The application allows financial analysts to efficiently retrieve, process, and analyze historical stock market data from the Macedonian Stock Exchange. The system automatically retrieves a list of issuers from the exchange website, filtering out irrelevant data such as bonds or numeric codes. Users can then choose specific issuers to focus on.

Once an issuer is selected, the system checks the database to see if historical data is already available. If not, it fetches data for at least the past 10 years. For issuers with existing data, the system determines the last available date and only fetches the most recent missing data, ensuring that the analyst always has up-to-date information without redundancy.

The system processes this raw data, formatting it according to consistent standards (dates and prices) before storing it in the database. The filters automatically handle the transformation of data, so the analyst doesn't need to manually clean or reformat it. Once the data is processed, the user can export it for deeper analysis or reporting.

Additionally, a timer tracks the application's performance, ensuring that the process of populating the database is as fast and efficient as possible, which helps analysts save valuable time. Throughout the process, the system ensures that data integrity is maintained, allowing analysts to focus on actionable insights rather than data management.

6. System Architecture Design

The application will be designed using the Pipe and Filter architecture, enabling a modular approach to data transformation.

6.1 Data Pipeline

The data pipeline will include the following filters:

1. **Filter 1: Issuer Extraction**:
    - Retrieves issuer codes from MSE, filters out bonds or codes with numbers.
2. **Filter 2: Last Date Check**:
    - For each issuer, checks the database for the last available date.
3. **Filter 3: Missing Data Retrieval**:
    - Downloads missing data up to the present date and formats it.
4. **Filter 4: Data Storage**:
    - Stores or appends data in the database or structured file in the correct format.

7. Data Description and Input/Output Requirements

- **Input**: Raw daily stock data for all issuers from MSE.
- **Output**: Standardized data stored in a structured format within a database.

8. Acceptance Criteria

- The application retrieves and processes data for all issuers on the MSE.
- The database is correctly populated with 10 years of daily data for each issuer.
- The data pipeline is efficient, and all required transformations are applied.
- The application meets the specified performance benchmarks.

9. Appendix

A. Resources

- MSE Historical Data: [URL to MSE historical data page]
- Python Libraries: BeautifulSoup, Pandas (optional for data handling), SQLAlchemy for database integration.

B. Glossary

- **Database**: An organized collection of data stored and accessed electronically.
- **Pipe and Filter**: A pattern for chaining operations to process data incrementally