# Macedonian stock market analysis - web application

## Functional Requirements (FR)

FR1: Data Retrieval

Automatically retrieve a list of all **valid** issuers (companies) from the Macedonian Stock Exchange website, *excluding* bonds and issuers with codes containing numbers.

FR2: Check Existing Data

For each issuer, check the database (or structured file) to determine the last available date of stock data. If data already exists, fetch only the missing data; if no data exists, fetch data for the last 10 years.

FR3: Download Missing Data

Automatically download **missing stock data** for each issuer, starting from the last available date to the present. If no data exists, fetch data for the last 10 years.

FR4: Data Transformation and Formatting

Transform the raw stock data into a **consistent format** (correct date and price formatting) before storing it in the database or structured file.

FR5: Data Storage

Store the retrieved stock data in a **structured database**, ensuring that new data is added without duplication and that it is correctly merged with previously existing data.

FR6: Performance Measurement

Implement a **timer** to measure how long it takes to fully populate an empty database with stock data. Display the time taken for the data-fetching process.

**Non-Functional Requirements (NFR)**

NFR1: Performance

The system must process and retrieve data **efficiently**. The application should minimize the time required for data fetching, transformation, and storage to ensure a fast response, especially as the volume of stock data grows

.

NFR2: Scalability

The system should be **scalable to handle a growing amount of stock data over time**. As the number of issuers or the time window for historical data increases, the application should be able to scale without significantly impacting performance.

NFR3: Reliability

The system should be reliable and robust, meaning it **should handle errors gracefully** (e.g., missing data, connection issues) and continue processing other tasks without crashing or losing data.

NFR4: Data Integrity

The system must **ensure data integrity** throughout the entire process. Data must be correctly retrieved, transformed, and stored. There should be no inconsistencies such as incorrect dates, prices, or duplicates in the database.

NFR5: Maintainability

The system should be designed to **be maintainable and easy to update**. Code should be modular, with clear documentation, so future developers can easily modify, extend, or troubleshoot the application.

NFR6: Security

Although security is not a primary concern in this application, basic security practices should be followed to protect any sensitive information, such as API keys or user credentials, if applicable.