Architecture design

1. Conceptual architecture

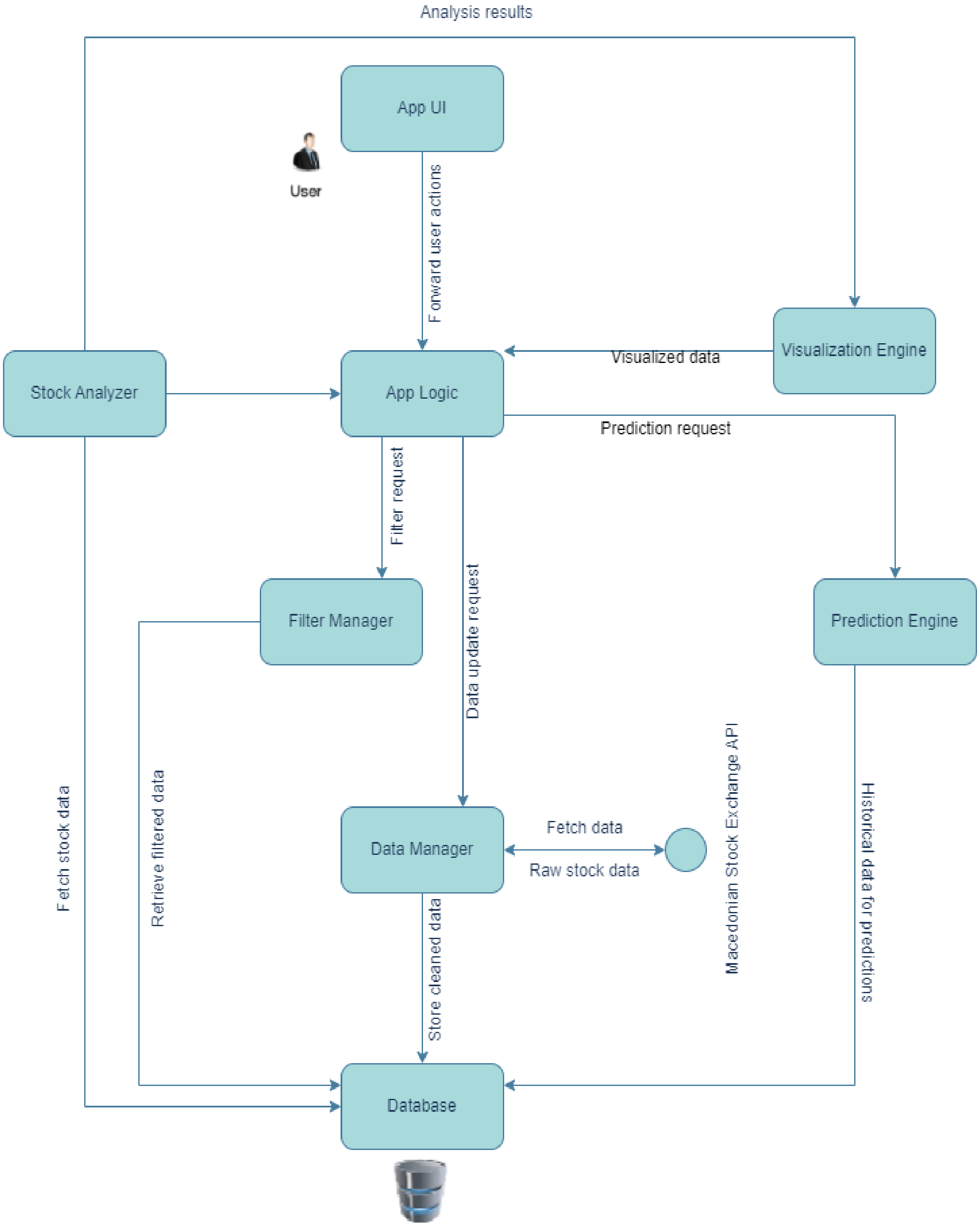
1.1. Categorisation of key concepts

1) The system must automatically collect and update missing records for historical stock data, ensuring comprehensive datasets for analysis.
2) It should provide the capability to calculate RSI, trend lines, and other technical indicators for technical analysis, aiding professional investors in decision-making.
3) The application must display key financial metrics (EPS, P/E, ROE) for fundamental financial analysis, helping investors and financial managers assess stock values.
4) Using historical data, the system should enable stock predictions through machine learning models, offering traders insights into future trends.
5) Users should be able to filter data by companies and time periods, allowing for personalized and focused analysis within the user interface.
6) The application must include interactive graphical visualizations for price history and trends, enhancing the experience for students and investors.
7) It should feature a table of the top 10 most active stocks, presenting real-time trading activity for users.

Data	Function	Stakeholder	System	Abstract Concept
Historical stock data	Automatically collect and update missing records	Economics Students	Macedonian Stock Exchange	Historical stock data
Technical indicators	Calculate RSI, trend lines, and other technical indicators	Professional Investor		Technical analysis for trading trends
Financial metrics	Display EPS, P/E, ROE for fundamental analysis	Financial Manager		Fundamental analysis of stock value
Stock predictions	Use historical data for future trend projections	Trader		Stock prediction
Filtered data	Filter by companies and time periods	User		Personalized and focused data analysis
Graphical visualizations	Display interactive charts for price history			Visual trends and insights
Most active stocks	Display a table of the top 10 most active stocks			Real-time trading activity

Table 1: Categorisation of key concepts

1.2. Conceptual architecture design



Picture 1. Conceptual architecture

1.3. Component responsabilities

App UI:

- Show the filtered stock information and analysis results.
- Provide interactive charts and graphs for users.
- Allow users to submit filters and analysis requests.

App Logic:

- Process and forward user requests to the appropriate components (e.g., Filter Manager, Stock Analyzer).
- Collect results from other components and prepare them for display in the App UI.

Filter Manager:

- Apply user-defined filters (e.g., time period, company name) to retrieve specific data subsets from the Database.

Data Manager:

- Fetch raw stock data from the Macedonian Stock Exchange API.
- Clean, format, and prepare the data for storage in the Database.
- Ensure data consistency and updates with the external API.

Stock Analyzer:

- Calculate RSI, trend lines, and other indicators for stock analysis.
- Analyze EPS, P/E, ROE, and other metrics.
- Determine the top 10 most active stocks based on trading data.

Prediction Engine:

- Use machine learning models to predict future stock trends based on historical data.

Visualization Engine:

- Generate visualizations for stock trends, analysis results, and predictions.
- Allow users to interact with data visualizations for better insights.

Database:

- Store all historical and processed stock data.

- Provide quick access to data subsets for filtering and analysis.

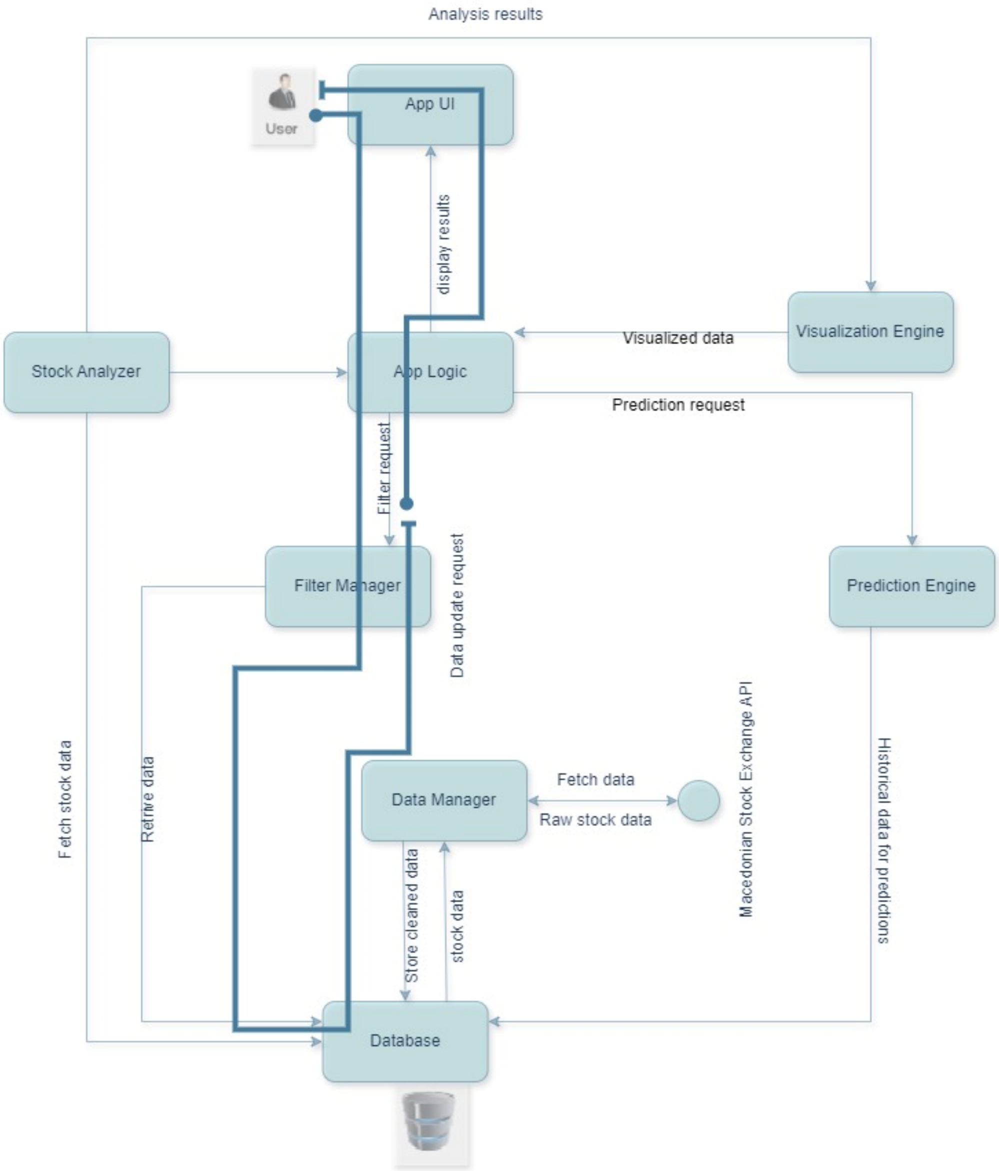
Macedonian Stock Exchange API (External System):

- Serve as the primary source of historical stock data for the application.

1.4. Behaviour model

Narative:

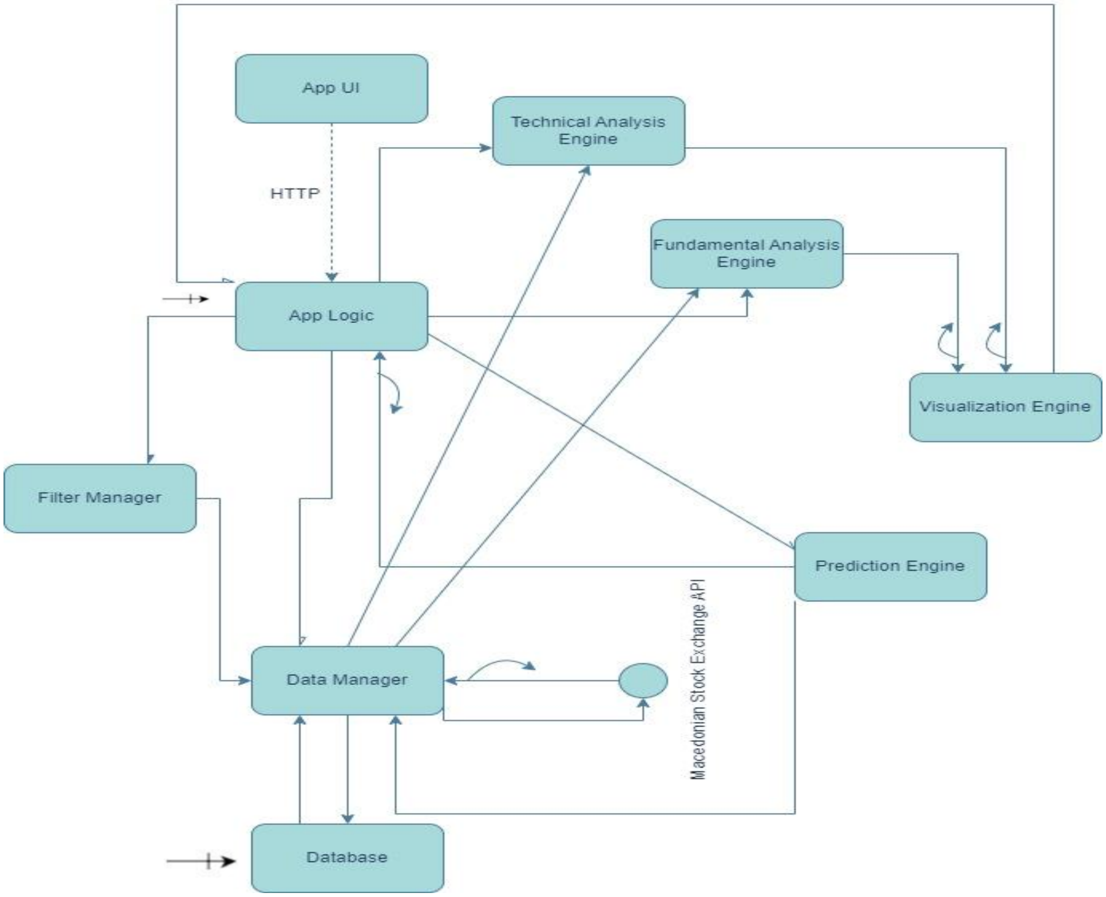
The user filters stock data based on specific criteria, such as company name or time period. After that filtered results are displayed.



Picture 2. Behaviour of conceptual architecture

2. Execution architecture

2.1. Execution architecture design



Picture 3. Execution architecture

2.2. Components of the execution architecture

App UI:

- Handles user requests.
- Displays visualized results and graphs through an interactive interface.
- Provides an interface for entering criteria for analysis.

App Logic:

- Coordinates data from all modules (technical analysis, fundamental analysis, predictions).
- Sends requests to relevant components (Filter Manager, Data Manager, Prediction Engine).
- Processes requests and returns results for visualization.

Technical Analysis Engine:

- Calculates technical indicators such as RSI and trend lines.
- Sends technical results to the Visualization Engine.

Fundamental Analysis Engine:

- Computes financial metrics (EPS, P/E, ROE).
- Sends results to the Visualization Engine.

Visualization Engine:

- Generates interactive charts and visualizations based on received data.
- Sends visualized data to App Logic for display in the App UI.

Filter Manager:

- Applies filtering criteria to the data (e.g., company, time period).
- Sends filtered results to the Data Manager.

Data Manager:

- Manages data requests from the database.
- Fetches data from the Macedonian Stock Exchange API.
- Stores and updates data in the database.

Database:

- Stores historical and updated stock data.
  Provides data for technical analysis, fundamental analysis, and predictions.

Prediction Engine:

- Calculates predictions for future trends based on historical data.
- Returns results to App Logic for visualization.

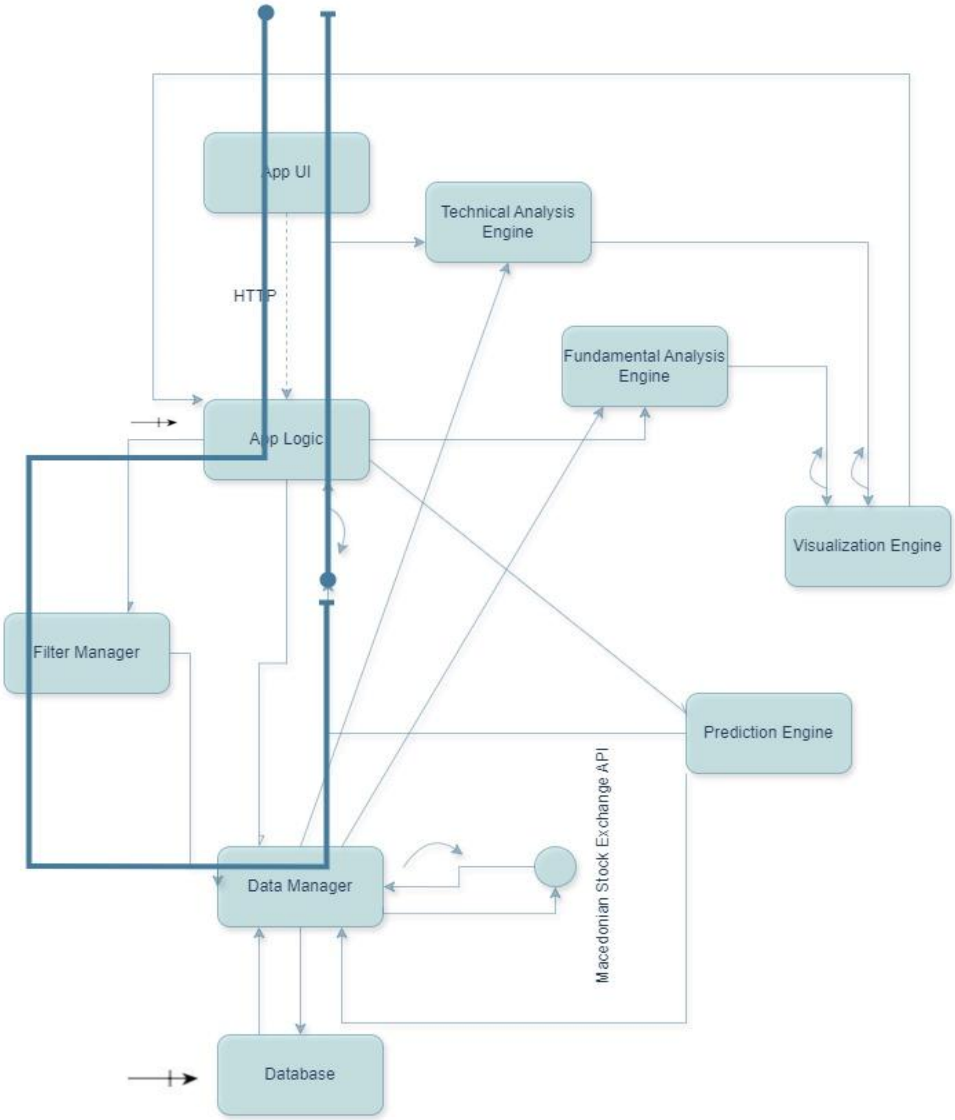
Macedonian Stock Exchange API:

- Provides real-time stock data.

- Sends raw data to the Data Manager.

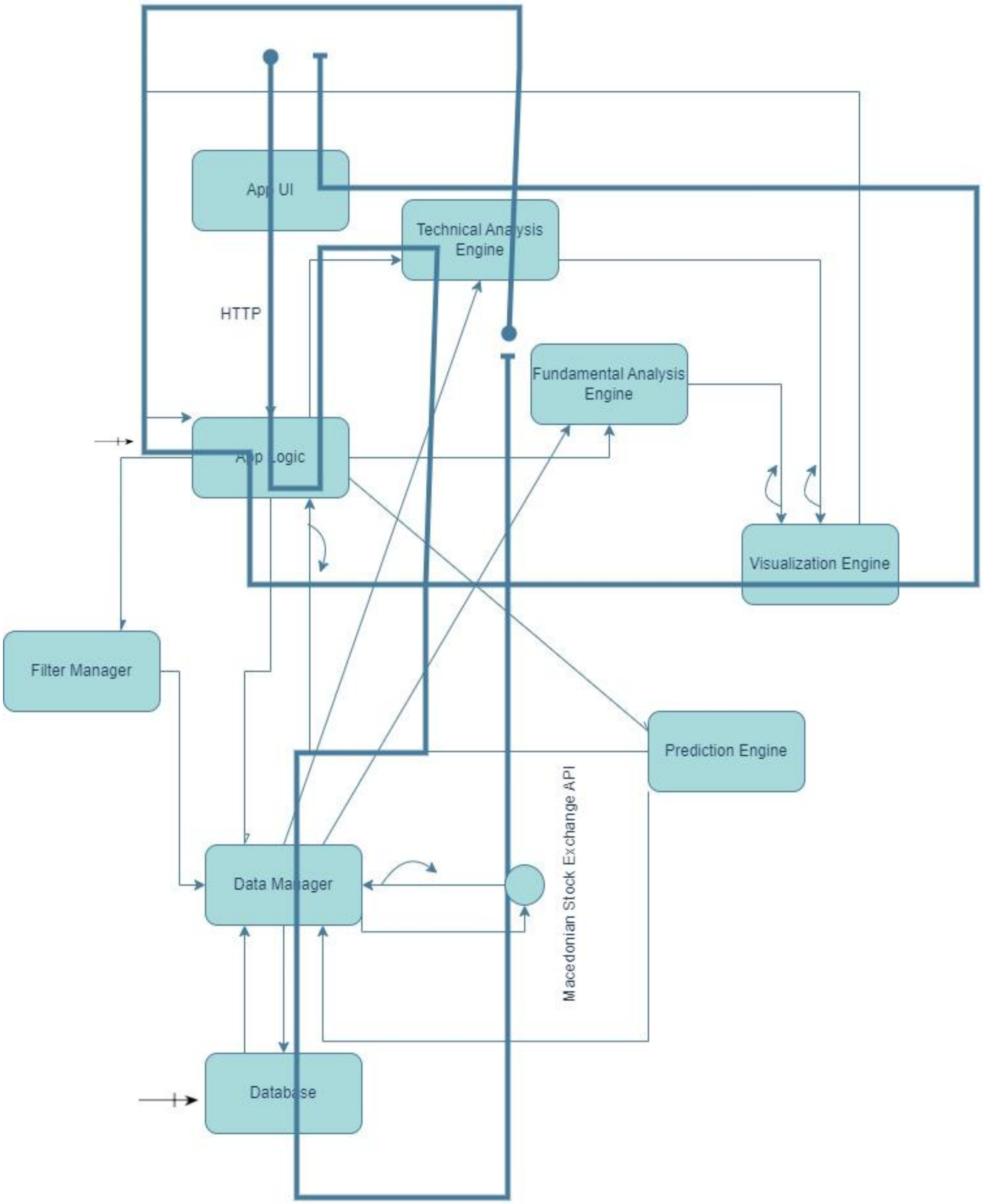
2.3. Execution behavior

1) Use case: The user selects specific filters (e.g., company name, time period) to view relevant stock data. The system retrieves and displays the filtered data.



Picture 4. Behaviour of the execution architecture

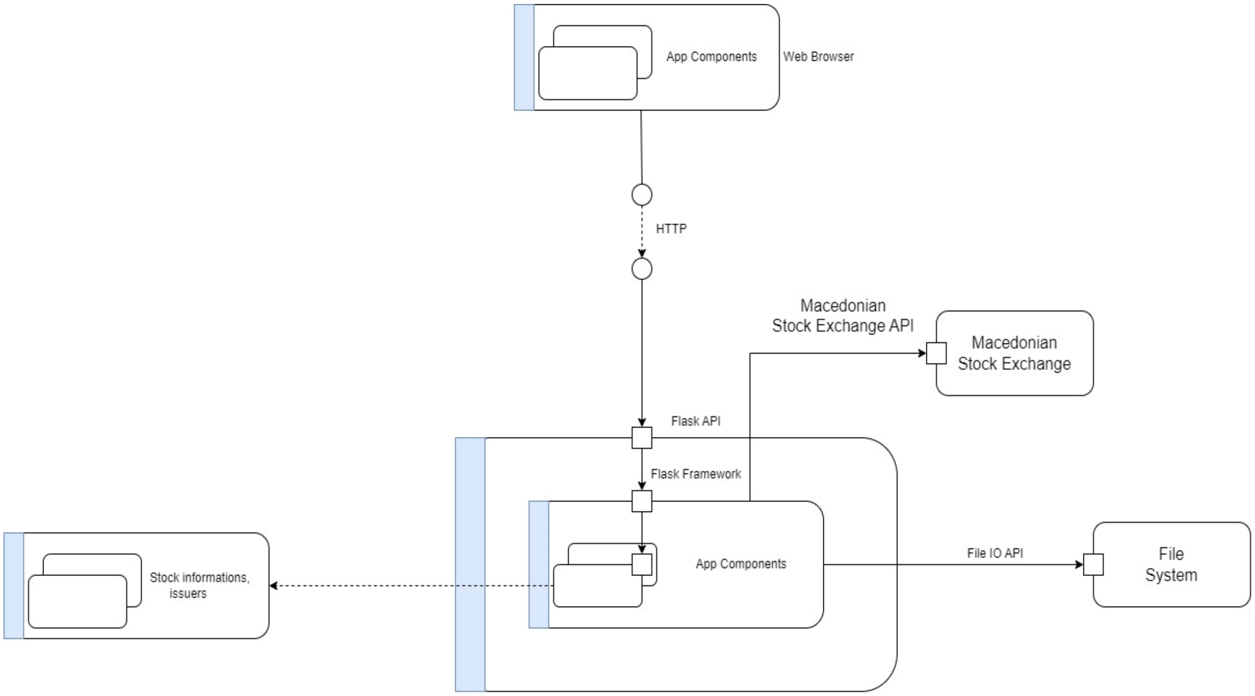
2) Use case: The user requests technical analysis (e.g., RSI or trend lines) for a selected stock.



Picture 5. Behaviour of the execution architecture

3. Implementation architecture

3.1. Implementation architecture design



Picture 6. Implementation architecture

3.2. Components in implementation architecture

Web Browser

- Provides a user interface (view) for interacting with the application.
- Utilizes the HTTP protocol for communication with the backend.
- Renders the application's user interface elements.
- Sends user requests to the server and receives responses.

Database

- Stores the necessary data to ensure the functionalities of the application.
- Holds stock information and analysis data.
- Facilitates efficient data retrieval and updates through the Data Manager.

Flask Framework

- Acts as the main backend framework for handling application logic and routing.
- Processes requests from the Web Browser.

- Connects with the Database for data manipulation.
- Manages API calls to external systems like the Macedonian Stock Exchange.

Flask API

- Handles HTTP requests for retrieving stock data.
- Integrates external APIs for data acquisition.
- Facilitates communication between the application components and external systems.

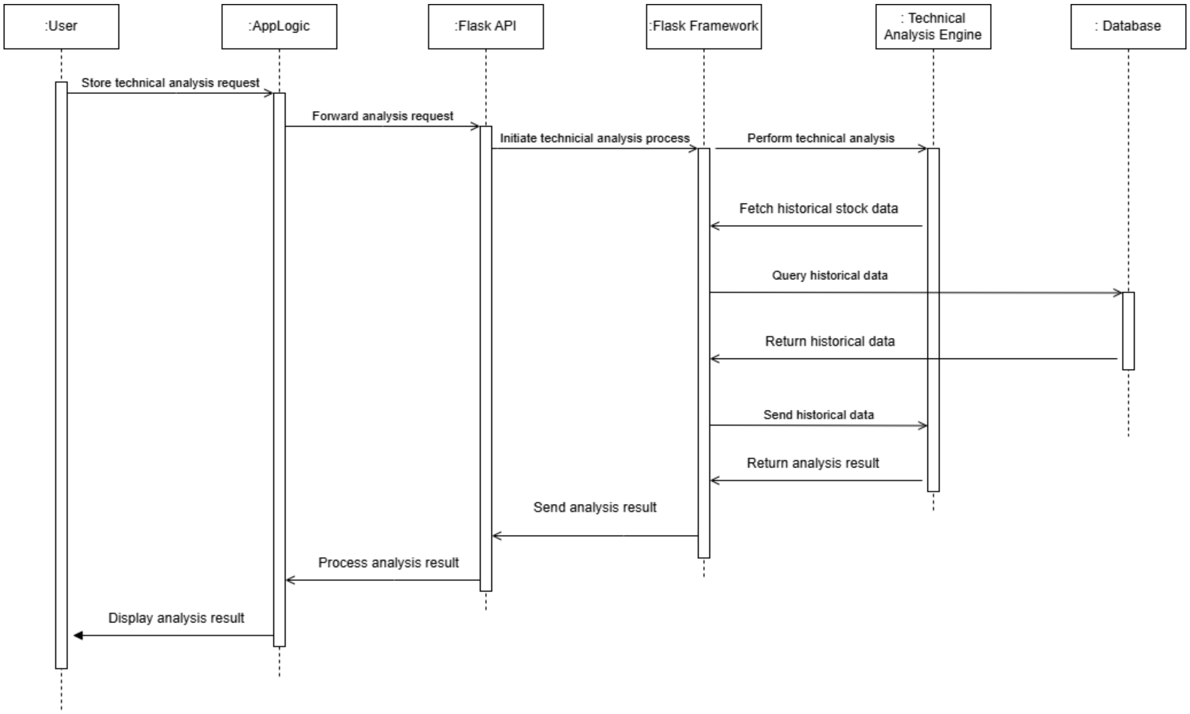
File System

- Manages file-based data storage for logging or exporting analysis data.
- Reads and writes stock-related files.
- Ensures backup or export functionality for critical information.

Macedonian Stock Exchange API

- Provides raw stock data from the Macedonian Stock Exchange.
- Enables real-time stock data retrieval.
- Supplies data for historical analysis and prediction modules.

3.3. Behaviour Design



Picture 7. Sequence diagram