Project Requirements

1 INTRODUCTION

WealthFlow is a web application that provides users with a unified dashboard for tracking stocks and cryptocurrency assets. The product aggregates real-time and historical data from public financial APIs, allowing users to customize a list of assets they are interested in, view price trends on charts, and receive notifications when price thresholds are reached.

Project Scope: the product serves as a tracking and analysis platform but does not provide functionality for buying or selling assets. It focuses on data visualization rather than price prediction using machine learning algorithms.

2 USER REQUIREMENTS

2.1 Software Interfaces

The application will interact with the following external services:
1 Alpha Vantage API – source of stock and index quotes.
2 CoinGecko API – source of cryptocurrency quotes.
3 SendGrid – for implementing the notification system.
Primary development stack:
1 Client-side: TypeScript, React.
2 Server-side: Java, Spring Boot, PostgreSQL.

2.2 User Interface

The interface will be implemented as a single-page application (SPA) using React. It will include the following main screens:

1 Main Dashboard: a list of the user’s favorite assets with current prices, 24-hour changes, and short-term charts.

2 Search Page: a search bar and display of search results for stocks and cryptocurrencies.

3 Asset Details Page: detailed information about an asset, including an extended chart for various time periods and key metrics.

4 User Profile: personal user data and account settings.

2.3 User Characteristics

2.3.1 Finance Students. This group consists of students studying finance who will use the application for educational purposes. Their characteristics include:
1 Education Level: enrolled in higher education.
2 Experience: theoretical knowledge in finance.
3 Technical Literacy: high, with quick adaptation to new applications.

2.3.2 Amateur Investors. Эта группа представляет инвесторов-любителей, которые будут использовать This group represents amateur investors who will use the application for making small trades, monitoring, and market analysis. Their characteristics include:
1 Education Level: higher education in finance.
2 Experience: limited hands-on investment experience.
3 Technical Literacy: high, with the ability to analyze their virtual portfolio.

2.4 Assumptions and Dependencies

The following assumptions and dependencies apply to this product:

1 Access to the selected financial APIs (Alpha Vantage, CoinGecko) is stable and provides data in the specified format.

2 The notification functionality depends on the availability and capabilities of external services (SendGrid, Email).

3 Users have modern browsers with JavaScript enabled.

4 The project depends on the technology stack: Java 17+, Spring Boot, React, PostgreSQL.

3 SYSTEM REQUIREMENTS

3.1 Functional Requirements

The following functional requirements are defined for this product:

1 Asset Search and Viewing

- 1.1 The system must provide the ability to search for stocks and cryptocurrencies by name.
- 1.2 The system must display search results along with basic data.

2 Watchlist Management

- 2.1 Authorized users must be able to add assets to their watchlist and remove them from it.

3 Data and Chart Display

- 3.1 The system must display information about the selected asset.
- 3.2 The system must display a price chart for the selected asset over various time periods (day, week, month, year).

4 Notification Management

- 4.1 Authorized users must be able to set a threshold price value for an asset in their watchlist.
- 4.2 The system must track price changes and send a notification to the user if the price reaches the set threshold.

5 Authorization and Authentication

- 5.1 The system must provide the ability to register a new user (login, password).
- 5.2 The system must provide the ability to authenticate a registered user.

3.2 Non-Functional Requirements

3.2.1 Reliability. Justification: The application must operate stably and display up-to-date financial data, as users base personal financial decisions on this information. Metric: Uptime $\geq$ 99.5%. Errors due to external API unavailability must be handled with appropriate user notifications.

3.2.2 Performance. Justification: The user interface must be responsive, and data must load quickly to avoid hindering the analysis process. Metric: The main dashboard loading time (for an authorized user) must not exceed 2 seconds. Chart data should load within $\leq$ 1 second.

3.2.3 Security. Justification: Personal user data and settings must be protected. Metric: Passwords must be stored in the database in hashed form. All transmitted data must be protected using HTTPS. API access must be controlled using JWT tokens.

3.2.4 Scalability. Justification: The product architecture must allow for the addition of new data sources and features in the future. Metric: Clear separation between backend (Spring Boot) and frontend (React). Use of jOOQ for type-safe SQL queries to PostgreSQL, simplifying maintenance and database schema changes.