Functional Requirements:

1. User Registration and Profile Management:

1.1 Users must be able to register with a unique username, a valid email address, and a secure password.
1.2 Registered users must be able to securely log in using their username and password.
1.3 User profiles must store the following information:

1.3.1 Risk tolerance level (expressed as an integer value within a predefined range).

1.3.2 Investment goals, which may include specific financial targets.

1.3.3 Historical investment data, which will be stored in an array or a suitable data structure.

2. Optimized Portfolio Calculation:

2.1 Data Integration

2.1.1Gather and integrate financial news data from diverse sources, ensuring data relevance, and comprehensiveness.

2.1.2 Acquire historical stock price data for the selected stock universe.

2.2 Sentiment Analysis

2.2.1 Utilize machine learning models (e.g., Natural Language Processing techniques) to conduct sentiment analysis on news articles.

2.2.2 Assign sentiment scores (positive, negative, neutral) to each article.
Aggregate sentiment scores over a defined time period (e.g., daily, weekly).

2.3 Stock Selection and Ranking:

2.3.1 Implement a machine learning-based ranking mechanism that considers sentiment scores along with other relevant factors (e.g., historical performance, fundamentals).

2.3.2 Select a pool of potential stocks based on the ranking.

2.4 Efficient Frontier Optimization

2.4.1 Utilize machine learning to estimate expected returns and volatility (standard deviation) for each selected stock.

2.4.2 Generate a range of potential portfolios with varying asset allocations, including those that maximize returns for a given level of risk and those that minimize risk for a given level of return.

2.4.3 Apply machine learning optimization techniques, such as Mean-Variance Optimization, to identify the portfolio that best fits the user's risk-return preferences.

3. Data Integration

3.1 The system shall integrate with the Yahoo Finance API to retrieve real-time stock data.
3.2 Real-time stock data shall include current stock prices, financial ratios, and historical performance.
3.3 The system shall integrate relevant stock news feeds and make them available for user consumption.

4. Display Charts:

4.1 The system shall display interactive stock price charts using a suitable Python charting library.
4.2 Users shall be able to select different time frames for charts, such as 1 day, 1 week, 1 month, and 1 year.

5. Filter Stocks by Attributes:

5.1 Users shall have the ability to filter stocks based on attributes, such as the percentage change in the last 24 hours, through a user-friendly interface.

6. Transaction History:

6.1 The system shall maintain a transaction history for each user.
6.2 Transactions shall be logged in an array or database, including details such as stock symbols, transaction type (buy/sell), quantities, prices, and timestamps.
6.3 Automatic updates of the transaction history shall occur when users accept the recommended portfolio.
6.4 Manual logging of transactions shall be possible when users do not accept the recommended portfolio.

7. Notifications:

7.1 The system shall send notifications to users when there are significant changes in the stocks they are tracking.
7.2 Market updates shall be sent through notifications, including key financial events.

8. Performance Metrics:

8.1 The system shall calculate relevant financial metrics for the optimized portfolio, including:

- Expected return.
- Standard deviation (risk).
- Sharpe ratio (risk-adjusted return).

9. Backtesting and Machine Learning Model Validation:

9.1 Use historical data to backtest the performance of the optimized portfolio.
9.2 Validate the accuracy and robustness of the machine learning models used for sentiment analysis and portfolio optimization.

Non-Functional Requirements:

1. Performance:

1.1 The system must respond to user requests within a maximum of 2 seconds, especially when fetching real-time stock data and calculating portfolio optimizations.

2. Security:

2.1 Implement strong encryption mechanisms (e.g., SSL/TLS) for sensitive data transmission.
2.2 Ensure secure authentication and authorization mechanisms to protect user accounts.
2.3 Implement industry-standard security best practices to safeguard user data and financial transactions.

3. Scalability:

3.1 The system should be designed to handle a minimum of 10,000 concurrent users.
3.2 It should accommodate a dataset of up to 100,000 stocks without performance degradation.

4. Availability:

4.1 Aim for a minimum uptime of 99.5% during peak trading hours to minimize downtime.
4.2 Implement redundancy and failover mechanisms to ensure high availability.

5. Reliability:

5.1 The system must be capable of recovering gracefully from failures, with a maximum downtime of 30 minutes.
5.2 Implement automated backup and disaster recovery procedures.

6. User-Friendly Interface:

6.1 Design an intuitive and user-friendly web interface for both desktop and mobile users.
6.2 Ensure compatibility with major web browsers (Chrome, Firefox, Safari, Edge).

7. Data Accuracy:

7.1 Stock data must have a maximum deviation of 1% from the actual market values.
7.2 Implement data validation checks to maintain data accuracy.

8. Compliance:

8.1 Ensure compliance with relevant financial regulations, including data protection and user privacy.
8.2 Regularly audit the system to ensure it follows industry standards and best practices.

9. Maintenance and Updates:

9.1 Plan for regular system maintenance, including updates and patches, to ensure security and performance.
9.2 Notify users in advance of scheduled maintenance windows.

10. Documentation:

10.1 Provide comprehensive user documentation, including user guides and FAQs.
10.2 Offer administrator documentation for system configuration and maintenance.

11. Testing:

11.1 Conduct thorough testing, including unit testing, integration testing, and user acceptance testing, before each software release.
11.2 Perform load testing to ensure the system can handle peak user loads.

12. Data Privacy:

12.1 Implement data privacy measures to protect user information, including encryption of stored data.
12.2 Comply with data protection laws, such as GDPR or CCPA, regarding user data handling.

13. API Usage:

13.1 Implement robust error handling and retry mechanisms for API requests to Yahoo Finance.
13.2 Monitor API usage and set up alerts for rate limits or service disruptions to ensure uninterrupted service.