**Software Requirements Specification**

**InsightPredict, A Smart Stock Recommendation System for Personalized Investment Guidance**

**By:**

**Bayan Yahia**

**Adeeb Ganadry**

**Introduction**

**Purpose:**

This document outlines the software requirements for the InsightPredict project. InsightPredict is a stock recommendation system designed to assist investors in optimizing their portfolios using advanced machine learning techniques and user-defined preferences. The system provides personalized recommendations based on financial metrics, historical data, and risk tolerance levels.

**Scope:**  
InsightPredict is designed to simplify and enhance the portfolio management process for investors by leveraging advanced machine learning models and financial optimization techniques. The system will enable users to:

* Predict stock returns across multiple investment horizons using the Temporal Fusion Transformer (TFT) model.
* Optimize portfolio allocations based on Markowitz's Mean-Variance Optimization model.
* Incorporate personalized preferences such as risk tolerance, investment horizon, and portfolio size.
* Access a user-friendly interface with interactive dashboards and visualizations.
* Stay up-to-date with quarterly updates of financial data from reliable APIs and sources.

**Intended Audience:**  
This document is intended for developers, testers, and stakeholders involved in the InsightPredict.

**System Overview**

InsightPredict comprises several key components:

* Data Processing Module: Aligns daily stock prices with quarterly reports, cleanses data, and prepares it for analysis.
* Forecasting Module: Utilizes the Temporal Fusion Transformer model to predict stock returns.
* Optimization Module: Implements Markowitz’s Mean-Variance Optimization to generate efficient portfolios.
* User Preferences Module: Tailors recommendations based on portfolio size, investment horizon, and risk tolerance.
* Visualization Module: Provides interactive dashboards displaying portfolio performance metrics.

**Features:**

* Multi-horizon forecasting using the Temporal Fusion Transformer (TFT) model.
* Identification of key input features that impact stock performance.
* Implementation of Markowitz’s Mean-Variance Optimization model.
* Support for user-defined constraints and investment strategies.
* Customization of investment recommendations based on user preferences, including risk tolerance and portfolio size.
* Adaptive interface with interactive charts and dashboards.
* Automatic alignment of stock price data with quarterly financial reports.
* Noise reduction and missing data handling for improved accuracy.
* Regular updates with the latest financial data.
* Fine-tuning of prediction models with new data for enhanced accuracy.
* Detailed performance reports including metrics such as returns, risk, and Sharpe ratios.
* Comparison of portfolio performance against S&P500 benchmarks.

**Requirements in the form I am......... interested.........to:**

* I am an investor looking to leverage AI-driven forecasting models to enhance my portfolio performance.
* I am an investor interested in receiving tailored recommendations based on my risk tolerance and investment horizon.
* I am a data analyst aiming to ensure that financial data is properly aligned and processed for accurate investment decisions.
* I am a market researcher who needs regular updates on stock performance to refine investment strategies.
* I am a portfolio manager who wants to compare my investment performance against key financial benchmarks.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Requirement Statement | Requirement Type | Explanation |
| 1 | Align stock price data with quarterly financial reports. | Functional | Ensures financial data consistency for accurate analysis. |
| 2 | Reduce noise in stock price fluctuations to highlight medium-term trends. | Functional | Improves forecasting accuracy by removing short-term volatility. |
| 3 | Handle missing data by applying forward fill techniques. | Functional | Prevents gaps in data that may affect machine learning model performance. |
| 4 | Utilize the Temporal Fusion Transformer model for multi-horizon forecasting. | Functional | Provides AI-driven stock return predictions over different timeframes. |
| 5 | Provide interpretable outputs indicating the impact of different input features. | Functional | Enhances transparency of AI model recommendations. |
| 6 | Use Markowitz’s Mean-Variance Optimization model to recommend asset allocations. | Functional | Optimizes portfolio diversification and risk management. |
| 7 | Support user-defined constraints such as maximum portfolio size and investment horizon. | Functional | Allows customization to align with user-specific investment goals. |
| 8 | Allow users to specify risk tolerance using a scoring formula. | Functional | Adapts recommendations based on individual risk appetite. |
| 9 | Provide flexibility in selecting portfolio sizes and investment horizons. | Functional | Gives users control over their investment strategy. |
| 10 | Display portfolio metrics such as returns, risk, and Sharpe ratio. | Functional | Provides key financial indicators for informed decision-making. |
| 11 | Compare portfolio performance against S&P500 benchmarks. | Functional | Allows users to measure performance against a standard index. |
| 12 | The system should process and analyze data for 500 stocks within 1 hour. | Non-functional | Ensures performance efficiency for large-scale analysis. |
| 13 | Forecasting models must generate predictions with 95% accuracy on test data. | Non-functional | Guarantees reliable AI predictions for investment decisions. |
| 14 | Provide an intuitive and responsive user interface. | Non-functional | Ensures ease of use for investors and analysts. |
| 15 | Ensure accessibility for non-technical users through comprehensive tooltips and guides. | Non-functional | Makes the system user-friendly for all experience levels. |
| 16 | The system must have 99.9% uptime. | Non-functional | Ensures high availability for users relying on real-time insights. |
| 17 | Data integrity checks should prevent incorrect or incomplete data processing. | Non-functional | Maintains accuracy and reliability of financial data. |
| 18 | Support simultaneous analysis of multiple user-defined portfolios. | Non-functional | Allows parallel processing of different investment scenarios. |
| 19 | Accommodate additional data sources and APIs as required. | Non-functional | Ensures scalability and adaptability to future enhancements. |

**Functional and Non-functional Requirements**

**User Interfaces:**

**Stock Ranking List**  
Displays stocks ranked from best to worst based on user preferences. The ranking considers factors such as profit percentage, risk level, and market trends.

**Stock Information Table**  
Shows essential details for each stock, including:

* Stock Name & Symbol
* Weight in Portfolio
* Current Price
* Purchase Price
* Price Change (%)
* Profit/Loss Indicator

**Profit/Loss Graph**  
A dynamic chart that visualizes the user's profit over time based on the stock's purchase price and market fluctuations. Users can adjust the time range (daily, weekly, monthly, yearly).

**Stock Details & Quick Actions**  
Clicking on a stock provides more in-depth information, including performance trends and historical data.

**System Interfaces:**

**Financial Data API Integration**

* Uses **SEC EDGAR API** and **YFinance** to collect real-time and historical stock market data.
* Utilizes **Twelve Data API** and **Finnhub API** to gather additional stock details and financial insights.
* Leverages **NewsAPI** to track the latest news related to stocks and market trends.
* Ensures data is stored in the database for accurate analysis and recommendations.

**System Requirements:**

**Hardware Requirements**

* A computer with at least 8GB RAM and a multi-core processor for local development.
* Cloud computing resources (e.g., Google Colab, AWS, or Azure) for machine learning model training and data processing.
* Storage capacity to handle large datasets for stock history and financial metrics.

**Software Requirements**

* **Development Environment:**
  + Visual Studio Code (VS Code) for writing and debugging code.
  + Google Colab for training and testing machine learning models.
* **Programming Languages & Libraries:**
  + Python with libraries: Pandas, NumPy, TensorFlow/PyTorch, Scikit-learn for AI and data processing.

**Environment Requirements**

* Internet access for API calls and real-time data retrieval.