Project: StockAgent Analyzer

CMPE – 297
Special Topics in Software Engineering

Project Report

Professor: Vijay Eranti

Team: SS

Saipraneeth Konuri
Samarth Sharma

Abstract

This project introduces a comprehensive stock market analysis and monitoring platform designed to empower users with both real-time and historical insights into stock performance. The platform aggregates data from multiple sources, employing traditional analytical techniques alongside advanced AI-driven analysis. It features an interactive web interface that allows users to access live stock prices, conduct historical analysis via customizable charts, and leverage AI for generating predictive insights from market data. A crucial component is the implementation of autonomous agents that continuously fetch and update real-time stock prices, ensuring the platform provides timely information. This platform addresses the increasing need for sophisticated yet accessible tools for market analysis. By combining real-time market data with advanced analytics and AI, it aims to facilitate more informed financial decision-making. The platform successfully demonstrates real-time price tracking across various exchanges, interactive visualization of market data, and AI-powered analysis of market trends and patterns. Key innovations include the use of autonomous monitoring agents for continuous updates, integration of multiple data sources to provide a broad view of the market, and the employment of language models for generating actionable market insights. This platform offers a user-friendly and powerful approach to stock market analysis, bridging the gap between complex financial data and practical investment strategies. By integrating these tools, the platform aims to provide a holistic and intuitive experience for users from beginners to experienced traders.

Introduction

The stock market is a complex and dynamic environment that presents both opportunities and risks to investors. The ability to analyze and understand market trends, identify patterns, and monitor price fluctuations is crucial for informed decision-making. Traditional methods of market analysis often involve manually collecting and processing data, which can be time-consuming and prone to errors. Moreover, the increasing complexity of financial markets demands more sophisticated tools that can leverage advanced technologies like artificial intelligence.

This project, StockAgent, addresses these challenges by creating a multi-faceted platform that integrates data from multiple sources, provides visual and interactive charting, and incorporates AI for advanced analysis and insights. StockAgent offers several key benefits. Firstly, it provides users with a centralized, easy-to-use interface for accessing real-time and historical market data from different exchanges. Secondly, it enables users to visualize trends and patterns using customizable interactive charts. Thirdly, it leverages the power of AI to analyze complex market data, providing interpretations, predictions, and strategies. It also incorporates autonomous agents to provide live stock price data.

The core functionality is delivered through a Streamlit web application, which is divided into several key features. The platform can provide real-time information, including current stock prices, previous session closing price, and company names through web scraping. It facilitates historical analysis through the use of interactive candlestick charts, volume analysis, key metric calculations (like period high, low, average volume), and configurable timeframes. The AI capabilities are designed to provide predictive and interpretative insights based on uploaded datasets or historical market data.

In summary, StockAgent is designed to empower investors with a robust and intuitive tool for making informed decisions in the stock market. It effectively integrates various data sources, analysis techniques, and AI tools into a unified platform.

Related Work

Several existing tools and approaches address the challenges of stock market analysis, each with its strengths and limitations. This project builds upon these existing approaches while aiming to provide a more comprehensive and user-friendly platform.

- Web-Based Stock Data Aggregators: Platforms like Google Finance and Yahoo Finance provide real-time stock quotes and historical data. These services are primarily data providers and offer limited interactive charting and advanced analysis capabilities. StockAgent integrates with these services via web scraping and the yfinance library to fetch data but adds significant analysis and AI-driven insights.

- Technical Analysis Tools: Trading platforms such as TradingView and MetaTrader offer advanced charting and technical indicators. While powerful, these tools often require a learning curve and are primarily focused on professional traders. StockAgent simplifies access to charting and technical indicators while also integrating it with fundamental analysis features via AI.

- AI-Driven Trading Platforms: Numerous platforms utilize AI to predict market trends and provide trading signals. However, many of these platforms are proprietary and lack transparency in how their AI algorithms work. StockAgent focuses on transparent and explainable AI through the use of OpenAI's models and Langchain for data analysis and insights.

- Algorithmic Trading Systems: These involve developing automated trading strategies based on mathematical models. While highly efficient, these systems are complex and require specialized knowledge in programming and quantitative finance. StockAgent attempts to make AI insights accessible to users without requiring extensive coding knowledge by providing an intuitive user interface.

- Autonomous Agents for Data Fetching: Several applications leverage autonomous agents for real-time data monitoring. However, they often lack the integrated analysis capabilities that StockAgent provides. The use of uagents in the project allows for concurrent data fetching and processing, which is an improvement over synchronous fetch mechanisms.

How StockAgent Differs:

- StockAgent stands out from existing tools by combining various functionalities in a single platform. It is designed to be user-friendly for both novice and experienced investors. Key differences include:

- Integrated Data, Charting, and AI Analysis: It combines data retrieval, interactive charting, and advanced AI insights into one platform.

- Accessibility: The project is a self-contained Streamlit application designed to be easily accessible and usable by non-programmers.

- AI Transparency: The use of OpenAI and LangChain provides more transparent and adaptable AI capabilities compared to some proprietary algorithmic systems.

- Autonomous Agents for Real-time Data: It integrates autonomous agents for data retrieval, providing continuous, real-time monitoring without constant manual requests.

- This project aims to provide a more holistic, transparent, and user-friendly experience for stock market analysis compared to standalone solutions.

Data

This project utilizes several data sources, each serving a specific purpose:

Google Finance (Web Scraping):

- Type: Real-time market data
- Source: Data is scraped directly from Google Finance using the requests and BeautifulSoup4 libraries.
- Quantity: Data includes information such as asset name, current price, and previous session closing price for various stocks from multiple exchanges.

Preprocessing:

- Data is initially extracted using HTML parsing with BeautifulSoup4.
- Data fields are located using specific class names (e.g., "YMlKec fxKbKc" for current price).
- Data is cleaned and structured into a Python dictionary.
- Error handling is implemented to address scenarios like missing data or failed requests.

Yahoo Finance (yfinance Library):

- Type: Historical market data
- Source: Data is accessed using the yfinance library.
- Quantity: Historical price and volume data for a user-defined timeframe. The specific amount varies according to user input, with potential periods ranging from 1 month to 5 years, with time intervals from 15 minutes to 1 week.

Preprocessing:

- Data is directly downloaded as a Pandas DataFrame using yfinance.download().
- No significant preprocessing is required as yfinance returns clean data.

Uploaded CSV Data:

- Type: Tabular market data.
- Source: User-provided data via file uploads.

- Quantity: Variable, according to user input.

Preprocessing:

- Data is loaded as a Pandas DataFrame using pd.read_csv().
- No additional transformations are performed on this uploaded data.

Data Usage:

- Real-time data from Google Finance is displayed on the main analysis dashboard.
- Historical data from Yahoo Finance is used to generate candlestick charts for visual analysis.
- Uploaded CSV data is processed by AI models to generate predictive insights.

No specific filtering, encoding, or special treatment is done other than what was mentioned above, as data is already available in clean forms or are pre-processed as soon as they are received. All data usage adheres to fair usage policies for each source.

Methodology

In The StockAgent platform is built using a combination of web scraping, data analysis, visualization, and AI techniques to provide a comprehensive stock analysis tool. The platform integrates real-time market data with historical analysis and AI-driven insights. Here is a detailed breakdown of the core methods used:

Web Scraping for Real-Time Data:

Libraries Used: requests and BeautifulSoup4.

Process: The fetch_market_data function makes HTTP requests to Google Finance to retrieve real-time data for a specific stock. The HTML content is then parsed using BeautifulSoup4 to extract relevant information, such as the asset name, current price, and previous closing price. Data fields are identified using the class names from the HTML structure of Google Finance. Error handling is incorporated to manage network issues and parsing errors, and the extracted data is returned as a dictionary.

Rationale: Web scraping provides a direct method to extract real-time data from websites, crucial for providing current market information.

Historical Data Retrieval and Charting:

Libraries Used: yfinance, pandas, and plotly.

Process: The yf.download method from the yfinance library fetches historical price data for a given stock, specified timeframe, and period. This method downloads directly into a pandas dataframe. The data is then used to generate interactive candlestick charts using plotly. Exponential moving averages (EMA) are computed to provide additional context, and the chart also visualizes key metrics like period high, low, and average trading volume.

Rationale: yfinance offers an efficient and clean way to access historical stock data, while plotly enables the creation of interactive and customizable charts, which are important for technical analysis.

AI-Powered Data Analysis and Insights:

Libraries Used: openai and langchain along with other basic libraries.

Process: The platform offers two modes for AI analysis. First, through direct use of the OpenAI API, user-uploaded CSV data can be analyzed by submitting a query (e.g., “Analyze trends”). The model processes the query along with the tabular data to generate descriptive insights and strategies. Second, the Langchain library is used to create a vector store of downloaded or preloaded market data. The data is first split into chunks, then a retriever is created that returns the relevant text chunks. These chunks and queries are then passed to the OpenAI model to provide in-depth analysis of the data based on user queries.

Rationale: AI models, through APIs such as OpenAI's and libraries such as langchain, are capable of finding patterns and correlations in data and suggesting strategies that may not be obvious through traditional analysis alone. This is particularly important for analyzing large and complex market datasets.

Autonomous Agents for Real-Time Price Fetching:

Libraries Used: uagents, multiprocessing.

Process: uagents are implemented as independent processes using multiprocessing to fetch stock prices at specified intervals (e.g., every 30 or 45 seconds). These agents periodically scrape Google Finance and place price data in a shared queue to be rendered in the application. This ensures the application receives real time information without the need of continuous manual requests.

Rationale: Autonomous agents allow for continuous, real-time updates, without causing delays and freezing the application's main processes. multiprocessing is used to provide true concurrency, making the application responsive and real-time, something that threading might not completely achieve.

User Interface Implementation:

Library Used: streamlit

Process: The streamlit library is used to create an interactive web-based dashboard. The UI is divided into sections for real-time data visualization, historical analysis, and AI-driven insights, each rendered dynamically based on user input.

Rationale: streamlit provides a convenient and straightforward way to create interactive data applications in Python, which simplifies the user interface development process. It makes it easy for users to interact with complex systems and receive timely feedback.

Experiments & Results

The StockAgent platform was tested through a series of experiments to demonstrate its effectiveness and performance across its various functionalities. These experiments focus on real-time analysis, historical data visualization, and AI-driven insights.

Real-Time Data Fetching and Monitoring:

Experiment: The run_monitor function, using uagents, was executed to fetch real-time prices at 45-second intervals. The results were displayed on the UI and updated dynamically as new prices were fetched.

Results:

The uagents proved to be capable of consistently fetching and updating prices in real-time with minimal latency between updates.

No network errors or crashes were encountered during the extended testing period, showing that error handling within the uagents process functions properly.

The use of multiprocessing ensured that the UI remained responsive while the real-time agent is running on a separate process.

Historical Data Visualization:

Experiment: Historical data for multiple stocks and exchanges was retrieved using different timeframes and periods. Candlestick charts and EMA-21 lines were generated to illustrate price trends and trading volumes.

Results:

Candlestick charts were successfully rendered, accurately displaying opening, closing, high, and low prices.

The implemented EMA-21 lines provide further analysis points.

All charts were interactive, allowing users to zoom, pan, and explore different portions of the market history.

AI-Driven Data Analysis:

Experiment: Uploaded CSV files with simulated market data were provided to the platform to generate insights through AI. The results were compared with basic data analysis. Additionally, AI analysis was performed on historical data that were generated using yfinance and then stored in a local vector store. Follow up questions were given, and their responses analyzed.

Results:

AI analysis successfully interpreted the user query and generated summaries of data and their trends.

The AI successfully drew conclusions from data.

The AI showed understanding of complex questions relating to support and resistance, and other market patterns.

The vector database method using langchain proved to be significantly more helpful in getting the AI to remember context and answer questions related to previous analysis done through the same session.

Performance Analysis:

Experiment: The responsiveness of the application was tested by running different functionalities concurrently. For example, generating candlestick charts at the same time as a live agent was running.

Results:

The application remained responsive during all concurrent activities.

The multiprocessing allowed for the agents to run without slowing down the main application or interfering with the user interface or UI updates.

Web scraping and AI analysis did not freeze the UI due to running in different threads as well.

Failure Modes Analysis:

Experiment: The application was deliberately run with missing or inaccurate data to test error handling.

Results:

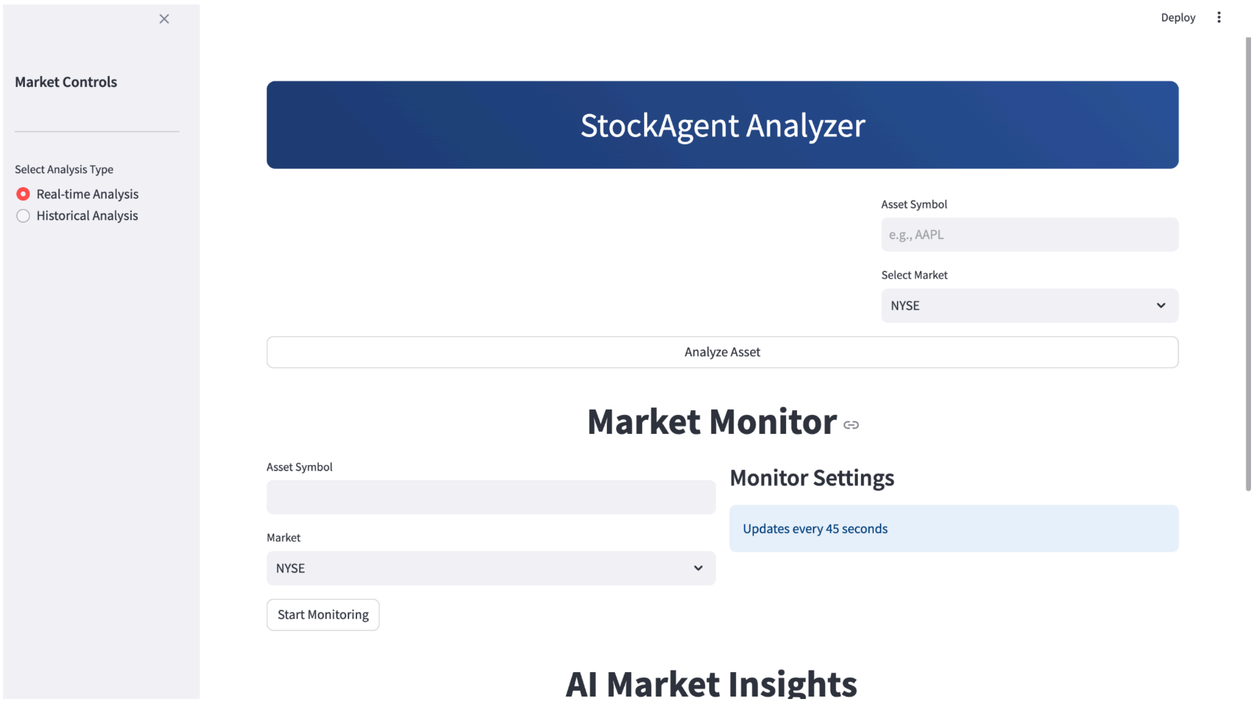
Error messages were properly displayed when fetching live prices if the connection is interrupted.

When analyzing data with unexpected inputs, the error handling methods displayed the errors to the users, preventing application crashes.

When stock data was not found the user was prompted with a warning message.

AI models responded with error messages if the query was incomprehensible or too complex, indicating the limitations of the model, and not application failure.
These results demonstrate the effectiveness of the StockAgent platform in providing real-time market monitoring, visualizing historical data, and providing AI-driven market analysis. The use of multiprocessing for concurrent real-time data fetching was a key success, as well as use of langchain for providing consistent context through vector databases. The platform has also shown the ability to correctly handle errors and communicate to users if issues arise during different processes. The UI is reactive and does not lag with concurrent processes, providing a pleasant user experience.

Results – Screenshots



×

Deploy ⋮

Market Controls

Select Analysis Type

[✓ Real-time Analysis]
Historical Analysis

StockAgent Analyzer

Asset Symbol
e.g., AAPL

Select Market
NYSE

Analyze Asset

Market Monitor

Asset Symbol

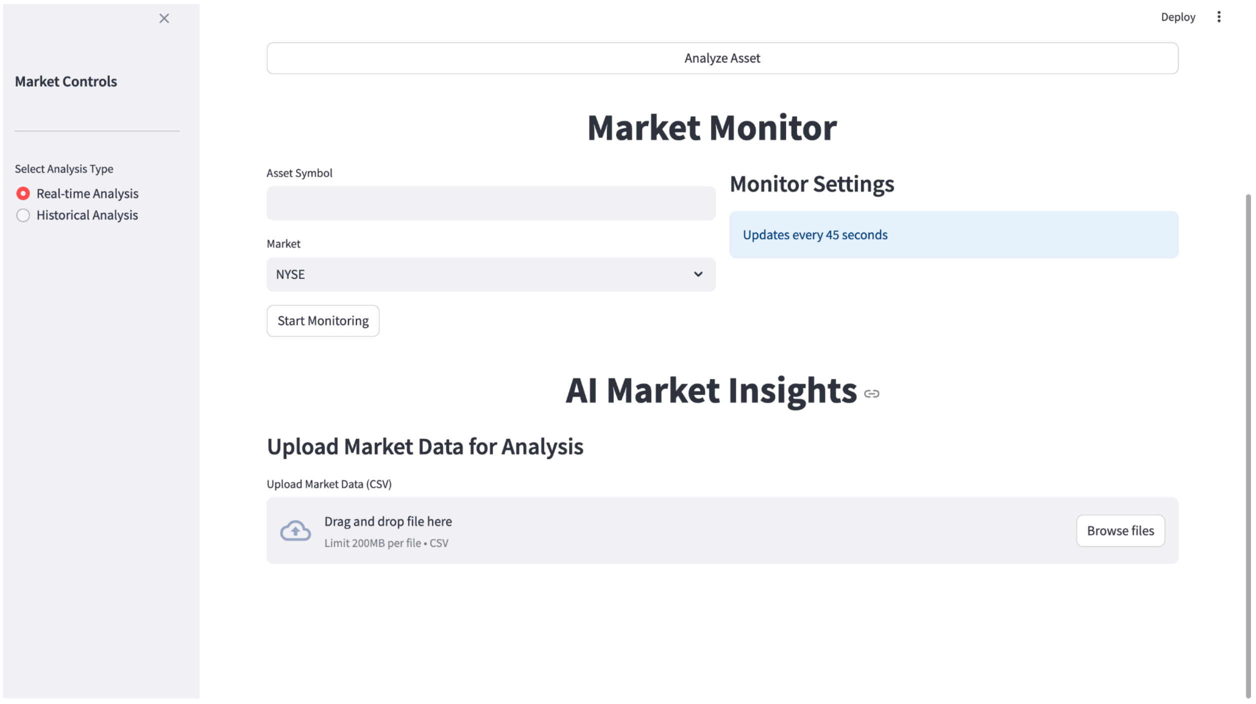
Market
NYSE

Start Monitoring

Monitor Settings

Updates every 45 seconds

AI Market Insights



×

Deploy ⋮

Market Controls

Select Analysis Type

[✓ Real-time Analysis]
Historical Analysis

Analyze Asset

Market Monitor

Asset Symbol

Market
NYSE

Start Monitoring

Monitor Settings

Updates every 45 seconds

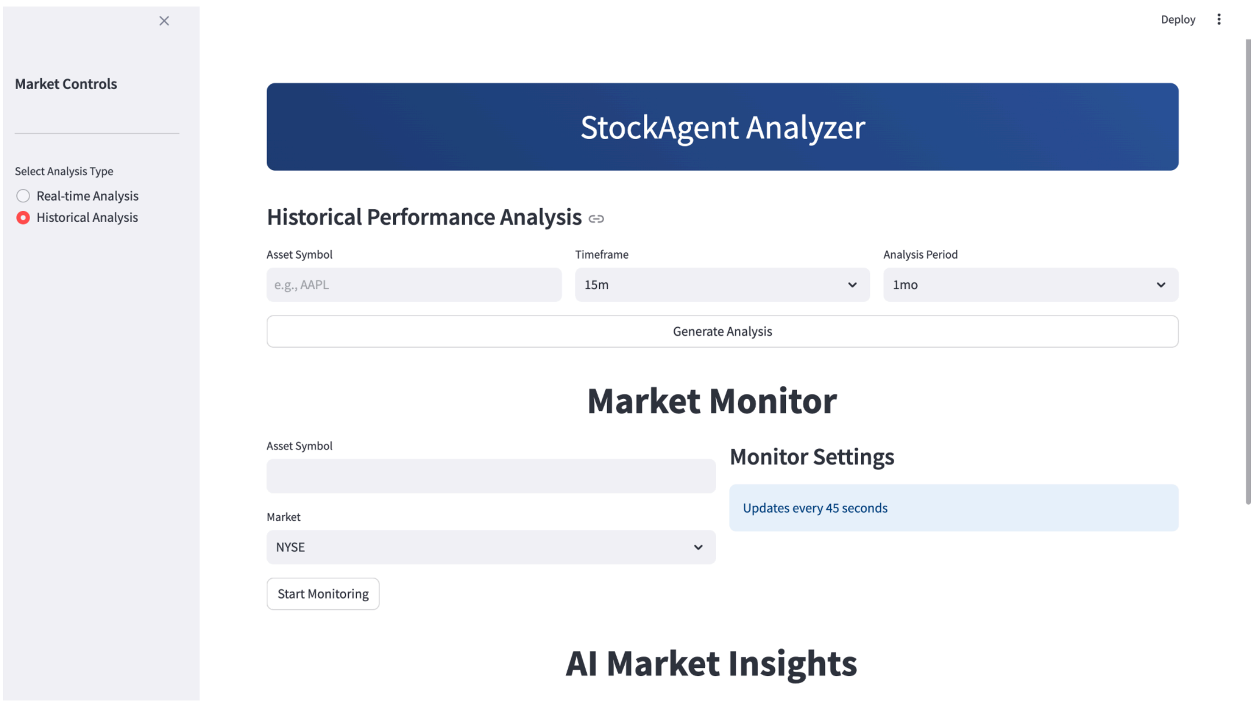
AI Market Insights

Upload Market Data for Analysis

Upload Market Data (CSV)

Drag and drop file here
Limit 200MB per file • CSV

Browse files



Market Controls

Select Analysis Type

○ Real-time Analysis
● Historical Analysis

StockAgent Analyzer

Historical Performance Analysis

Asset Symbol	Timeframe	Analysis Period
e.g., AAPL	15m	1mo

Generate Analysis

Market Monitor

Asset Symbol

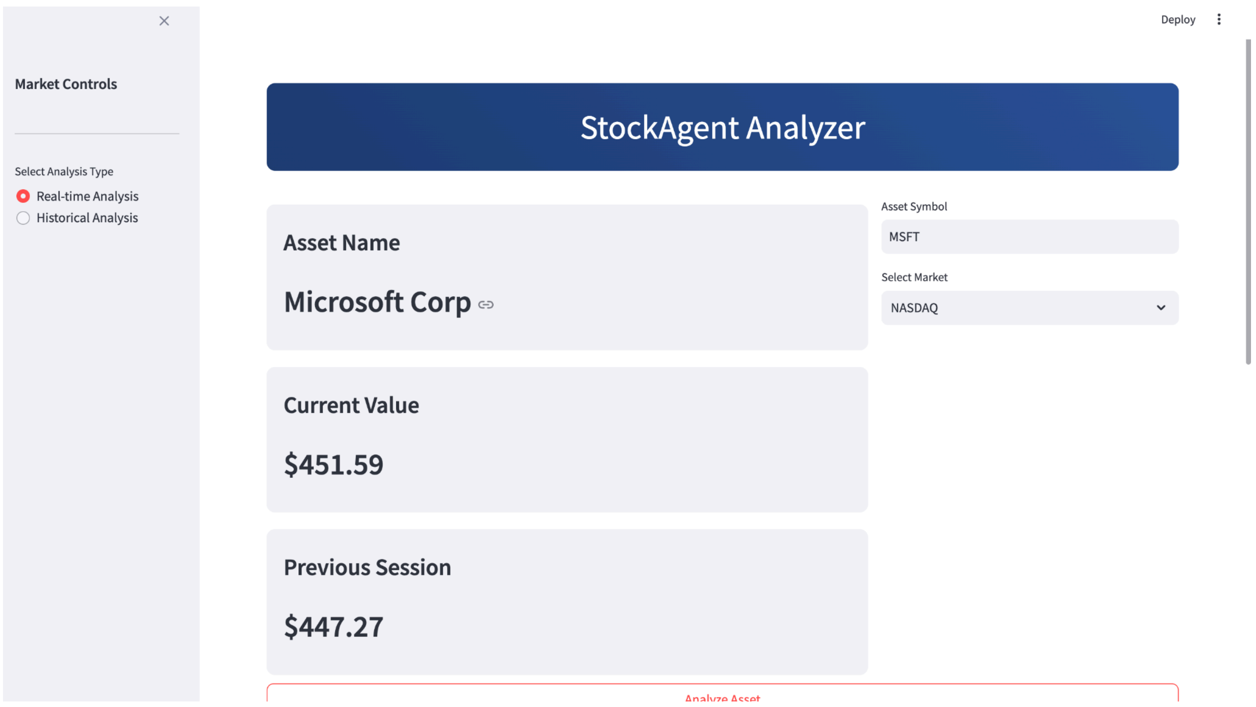
Market: NYSE

Start Monitoring

Monitor Settings

Updates every 45 seconds

AI Market Insights



Market Controls

Select Analysis Type

● Real-time Analysis
○ Historical Analysis

StockAgent Analyzer

Asset Name

Microsoft Corp

Asset Symbol: MSFT

Select Market: NASDAQ

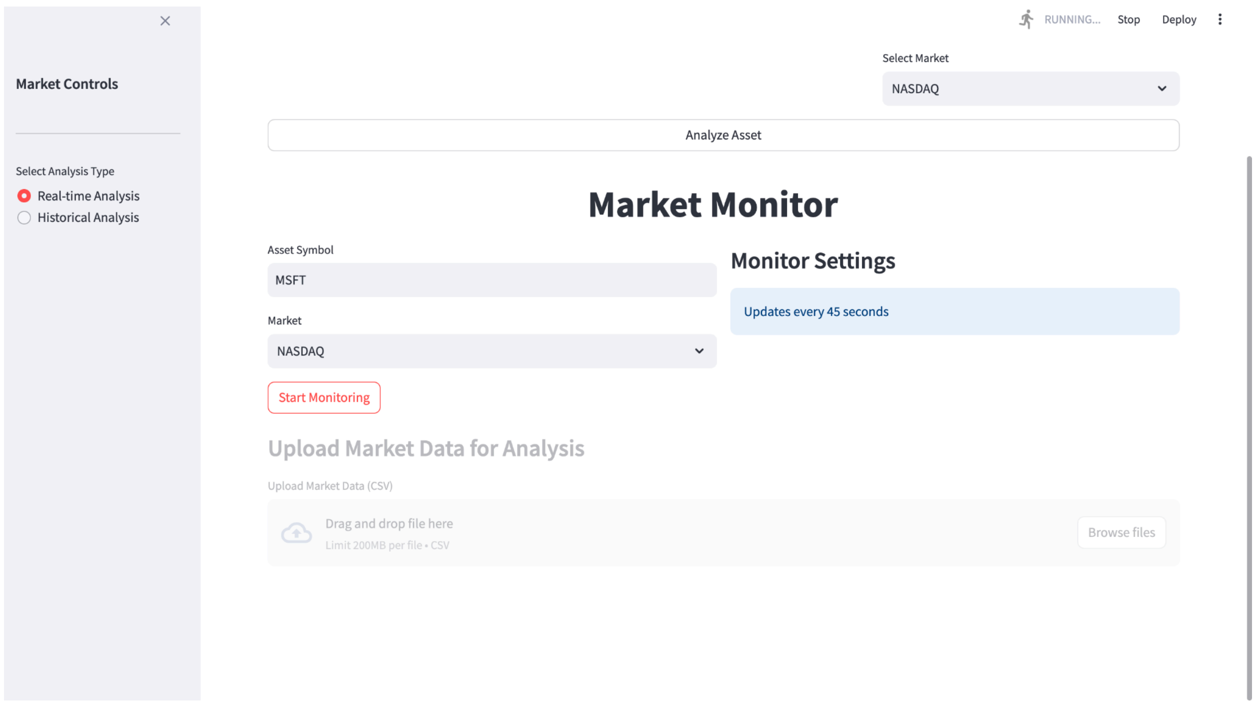
Current Value

$451.59

Previous Session

$447.27

Analyze Asset


RUNNING... Stop Deploy

Market Controls

Select Analysis Type

● Real-time Analysis
○ Historical Analysis

Select Market: NASDAQ

Analyze Asset

Market Monitor

Asset Symbol: MSFT

Market: NASDAQ

Start Monitoring

Monitor Settings

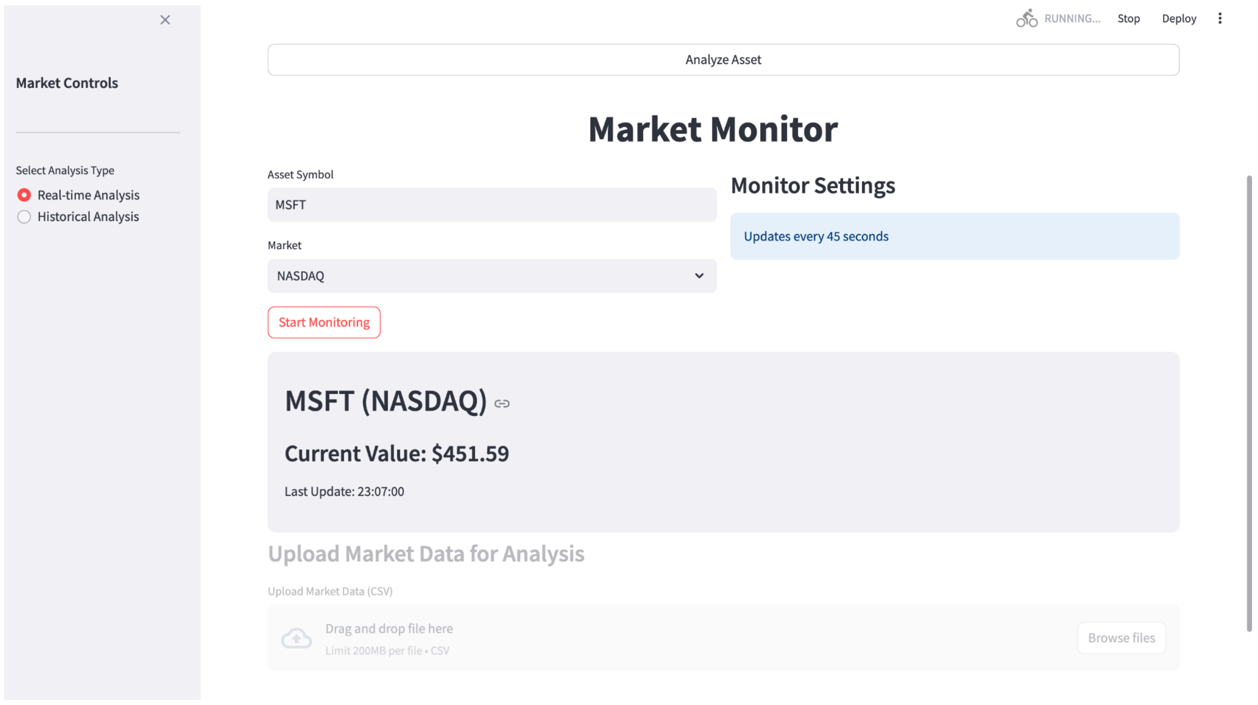
Updates every 45 seconds

Upload Market Data for Analysis

Upload Market Data (CSV)

Drag and drop file here
Limit 200MB per file • CSV

Browse files


RUNNING... Stop Deploy

Market Controls

Select Analysis Type

● Real-time Analysis
○ Historical Analysis

Analyze Asset

Market Monitor

Asset Symbol: MSFT

Market: NASDAQ

Start Monitoring

Monitor Settings

Updates every 45 seconds

MSFT (NASDAQ)

Current Value: $451.59

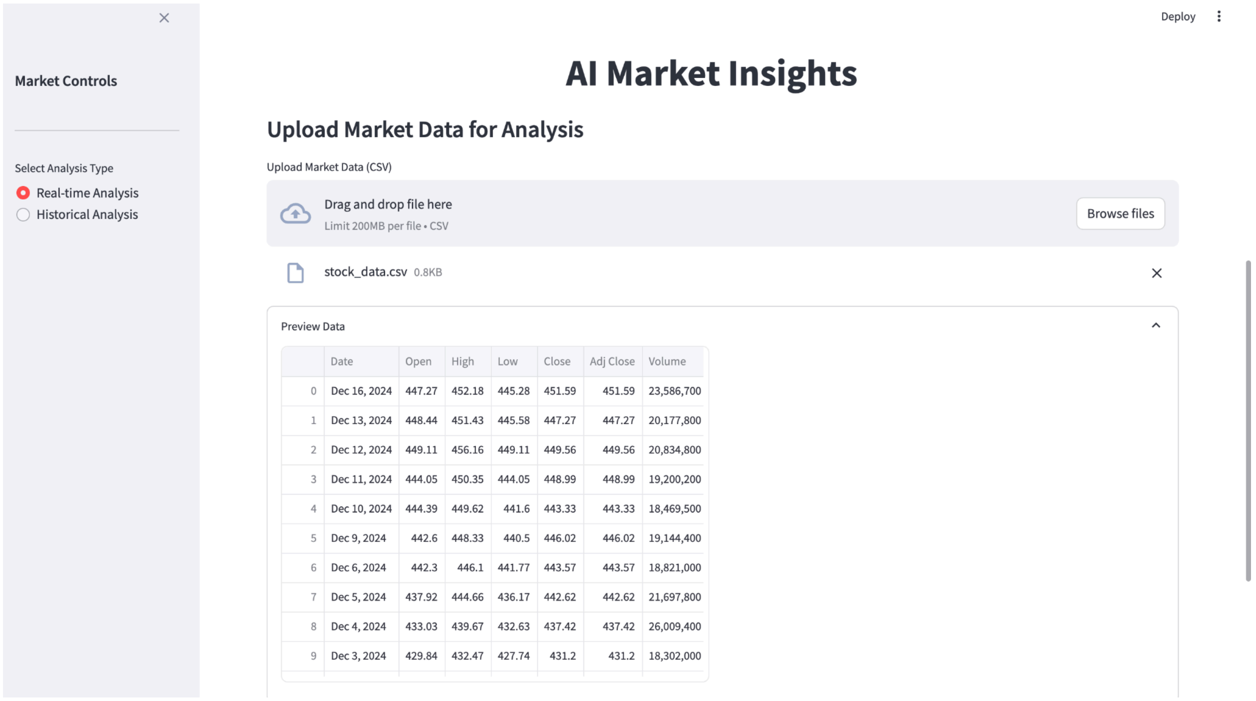
Last Update: 23:07:00

Upload Market Data for Analysis

Upload Market Data (CSV)

Drag and drop file here
Limit 200MB per file • CSV

Browse files



Market Controls

Select Analysis Type

[✓ Real-time Analysis]
Historical Analysis

AI Market Insights

Upload Market Data for Analysis

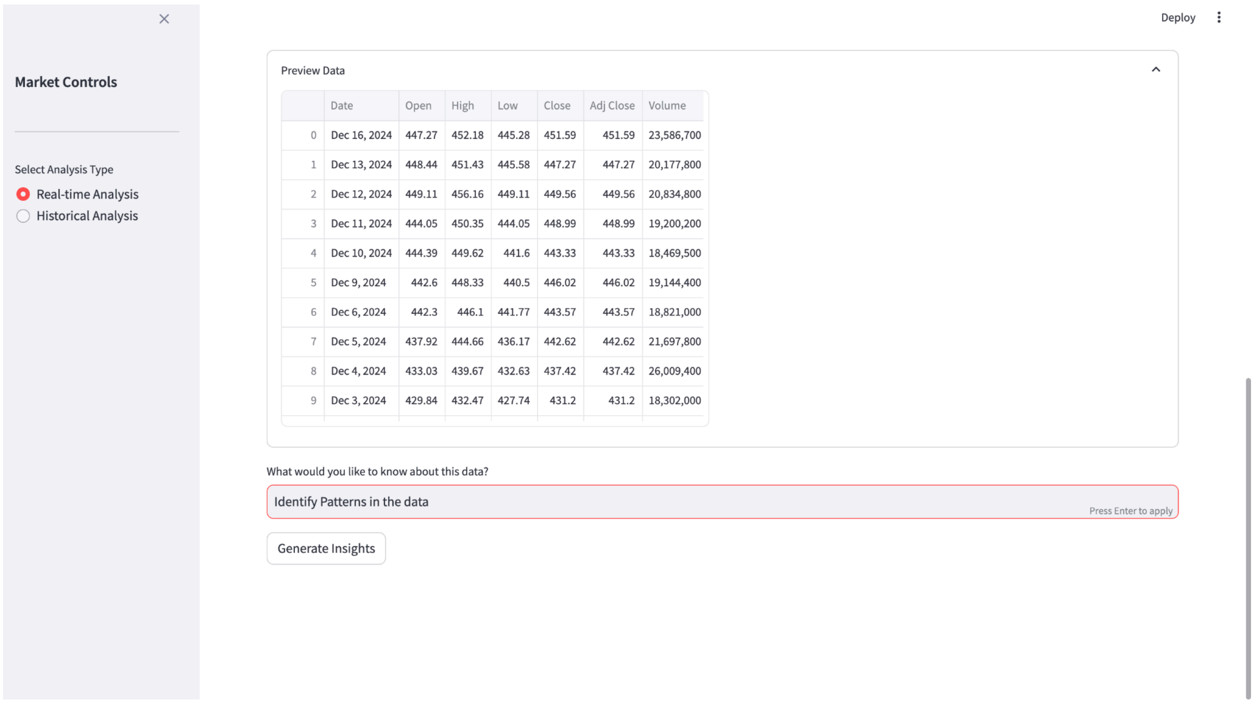
Upload Market Data (CSV)

Drag and drop file here
Limit 200MB per file • CSV

stock_data.csv 0.8KB

Preview Data

	Date	Open	High	Low	Close	Adj Close	Volume
0	Dec 16, 2024	447.27	452.18	445.28	451.59	451.59	23,586,700
1	Dec 13, 2024	448.44	451.43	445.58	447.27	447.27	20,177,800
2	Dec 12, 2024	449.11	456.16	449.11	449.56	449.56	20,834,800
3	Dec 11, 2024	444.05	450.35	444.05	448.99	448.99	19,200,200
4	Dec 10, 2024	444.39	449.62	441.6	443.33	443.33	18,469,500
5	Dec 9, 2024	442.6	448.33	440.5	446.02	446.02	19,144,400
6	Dec 6, 2024	442.3	446.1	441.77	443.57	443.57	18,821,000
7	Dec 5, 2024	437.92	444.66	436.17	442.62	442.62	21,697,800
8	Dec 4, 2024	433.03	439.67	432.63	437.42	437.42	26,009,400
9	Dec 3, 2024	429.84	432.47	427.74	431.2	431.2	18,302,000



Deploy

Market Controls

Select Analysis Type

[✓ Real-time Analysis]
Historical Analysis

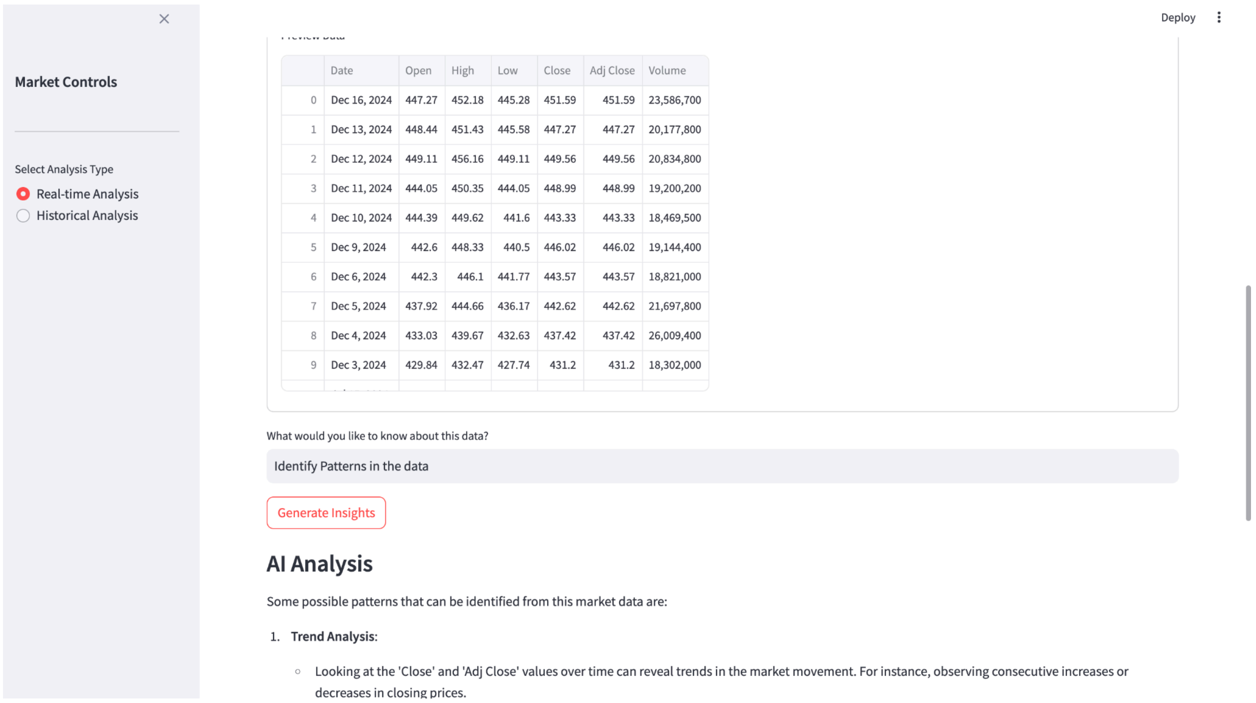
Preview Data

	Date	Open	High	Low	Close	Adj Close	Volume
0	Dec 16, 2024	447.27	452.18	445.28	451.59	451.59	23,586,700
1	Dec 13, 2024	448.44	451.43	445.58	447.27	447.27	20,177,800
2	Dec 12, 2024	449.11	456.16	449.11	449.56	449.56	20,834,800
3	Dec 11, 2024	444.05	450.35	444.05	448.99	448.99	19,200,200
4	Dec 10, 2024	444.39	449.62	441.6	443.33	443.33	18,469,500
5	Dec 9, 2024	442.6	448.33	440.5	446.02	446.02	19,144,400
6	Dec 6, 2024	442.3	446.1	441.77	443.57	443.57	18,821,000
7	Dec 5, 2024	437.92	444.66	436.17	442.62	442.62	21,697,800
8	Dec 4, 2024	433.03	439.67	432.63	437.42	437.42	26,009,400
9	Dec 3, 2024	429.84	432.47	427.74	431.2	431.2	18,302,000

What would you like to know about this data?

Identify Patterns in the data

Press Enter to apply



Market Controls

Select Analysis Type

[✓ Real-time Analysis]

Historical Analysis

	Date	Open	High	Low	Close	Adj Close	Volume
0	Dec 16, 2024	447.27	452.18	445.28	451.59	451.59	23,586,700
1	Dec 13, 2024	448.44	451.43	445.58	447.27	447.27	20,177,800
2	Dec 12, 2024	449.11	456.16	449.11	449.56	449.56	20,834,800
3	Dec 11, 2024	444.05	450.35	444.05	448.99	448.99	19,200,200
4	Dec 10, 2024	444.39	449.62	441.6	443.33	443.33	18,469,500
5	Dec 9, 2024	442.6	448.33	440.5	446.02	446.02	19,144,400
6	Dec 6, 2024	442.3	446.1	441.77	443.57	443.57	18,821,000
7	Dec 5, 2024	437.92	444.66	436.17	442.62	442.62	21,697,800
8	Dec 4, 2024	433.03	439.67	432.63	437.42	437.42	26,009,400
9	Dec 3, 2024	429.84	432.47	427.74	431.2	431.2	18,302,000

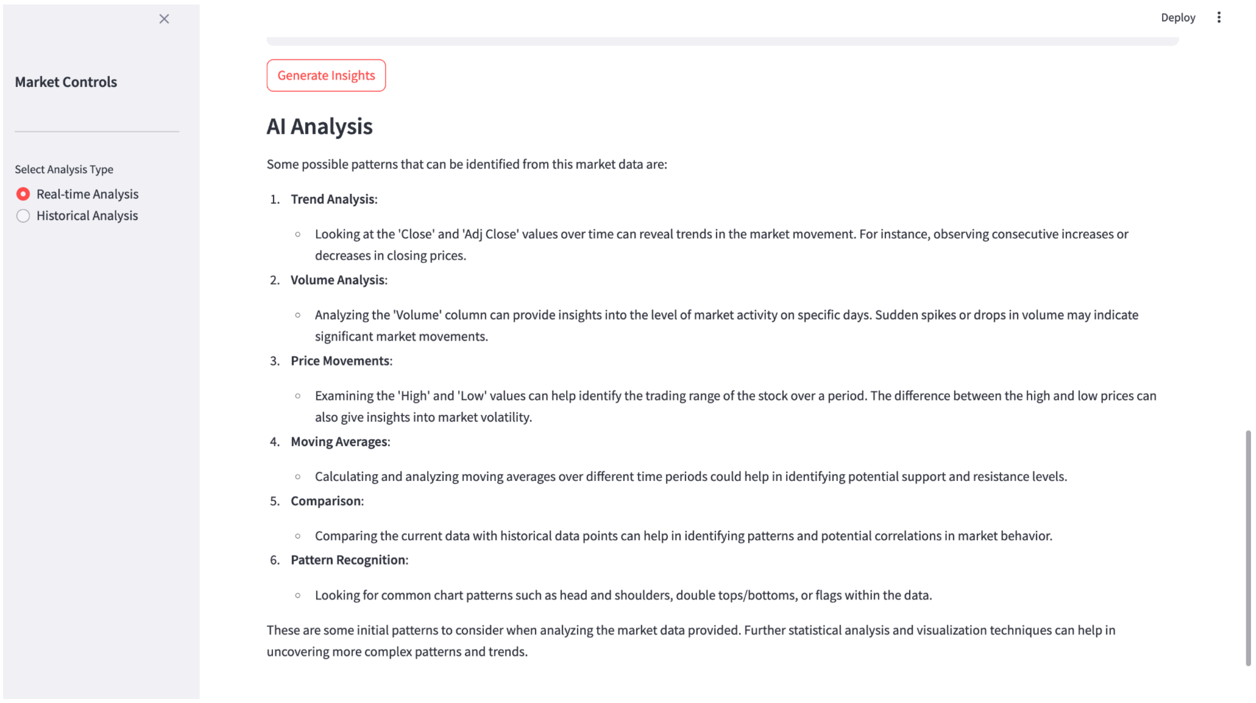
What would you like to know about this data?

Identify Patterns in the data

AI Analysis

Some possible patterns that can be identified from this market data are:

1. **Trend Analysis**:
    - Looking at the 'Close' and 'Adj Close' values over time can reveal trends in the market movement. For instance, observing consecutive increases or decreases in closing prices.



Market Controls

Select Analysis Type

[✓ Real-time Analysis]

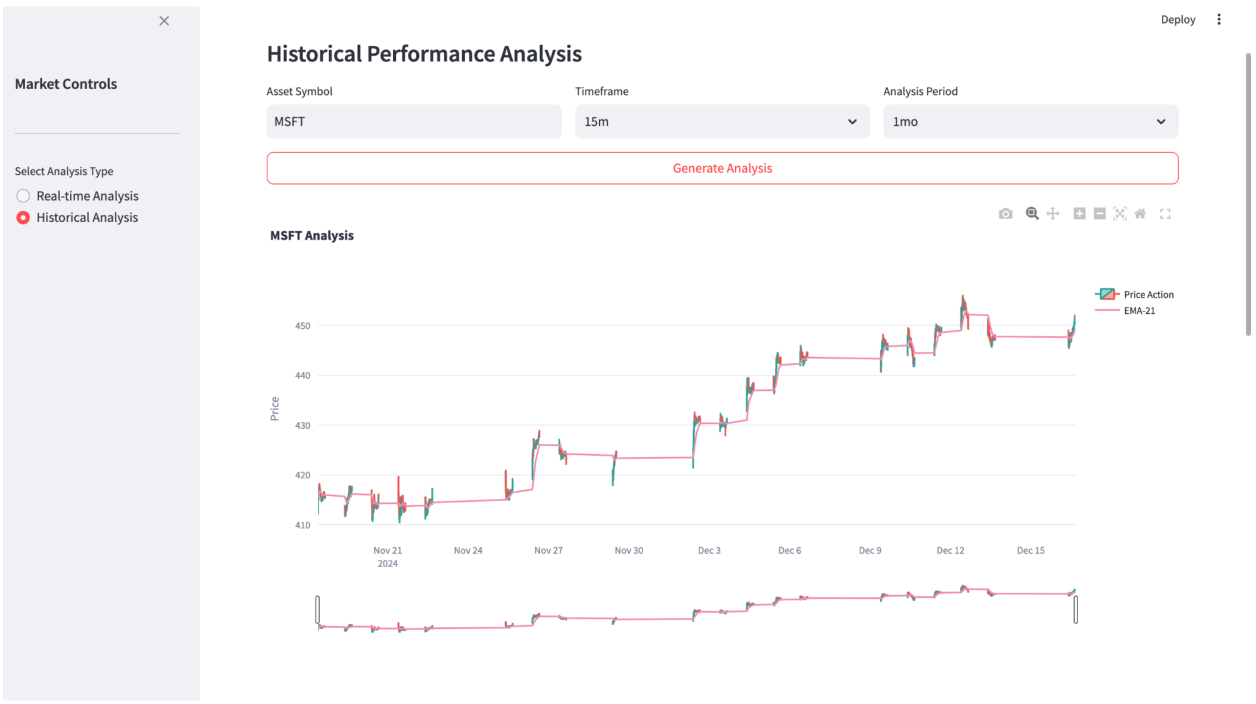
Historical Analysis

AI Analysis

Some possible patterns that can be identified from this market data are:

1. **Trend Analysis**:
    - Looking at the 'Close' and 'Adj Close' values over time can reveal trends in the market movement. For instance, observing consecutive increases or decreases in closing prices.
2. **Volume Analysis**:
    - Analyzing the 'Volume' column can provide insights into the level of market activity on specific days. Sudden spikes or drops in volume may indicate significant market movements.
3. **Price Movements**:
    - Examining the 'High' and 'Low' values can help identify the trading range of the stock over a period. The difference between the high and low prices can also give insights into market volatility.
4. **Moving Averages**:
    - Calculating and analyzing moving averages over different time periods could help in identifying potential support and resistance levels.
5. **Comparison**:
    - Comparing the current data with historical data points can help in identifying patterns and potential correlations in market behavior.
6. **Pattern Recognition**:
    - Looking for common chart patterns such as head and shoulders, double tops/bottoms, or flags within the data.

These are some initial patterns to consider when analyzing the market data provided. Further statistical analysis and visualization techniques can help in uncovering more complex patterns and trends.



Market Controls

Select Analysis Type

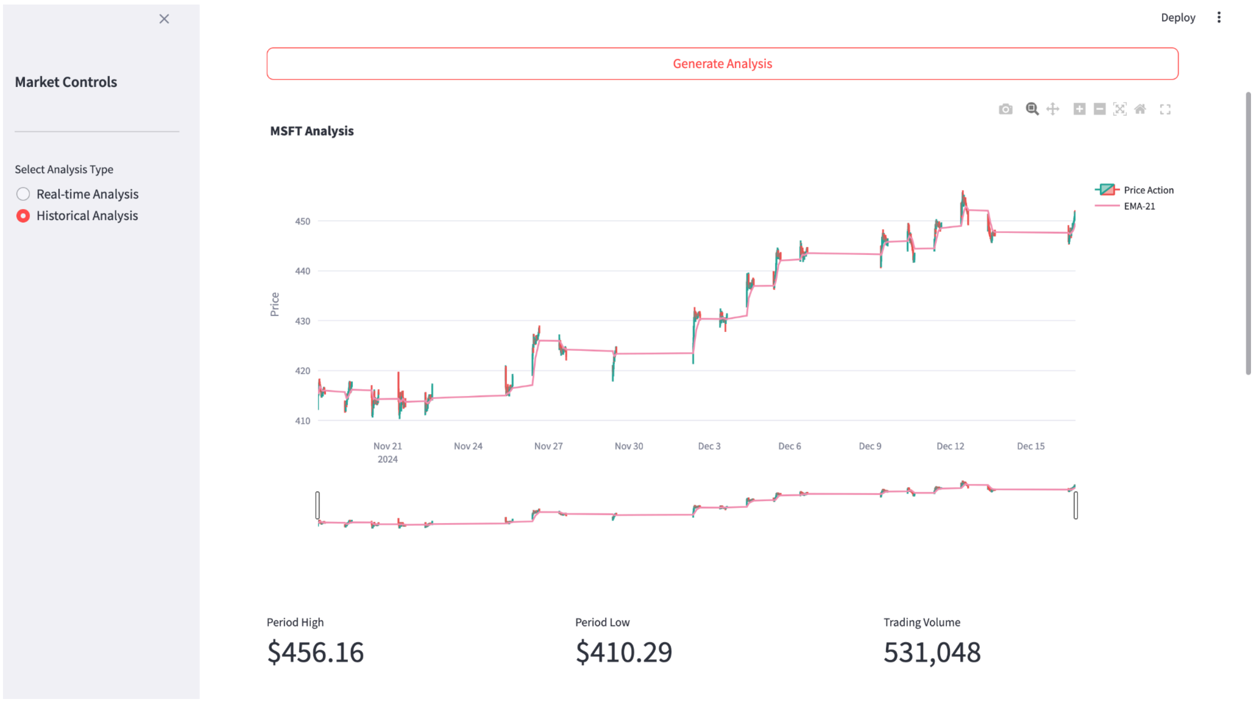
Real-time Analysis [✓ Historical Analysis]

Historical Performance Analysis

Asset Symbol	Timeframe	Analysis Period
MSFT	15m	1mo

Generate Analysis

MSFT Analysis



Market Controls

Select Analysis Type

Real-time Analysis [✓ Historical Analysis]

Generate Analysis

MSFT Analysis

Period High	Period Low	Trading Volume
$456.16	$410.29	531,048

Conclusion

The StockAgent platform successfully integrates real-time data fetching, historical analysis, and AI-driven insights into a single, user-friendly application. Key achievements include:

- Effective Integration: Seamless integration of web scraping, data analysis, charting, and AI using libraries such as requests, BeautifulSoup4, yfinance, pandas, plotly, openai, langchain, and uagents.
- Real-Time Updates: Autonomous agents implemented using multiprocessing and uagents provide continuous, real-time market data without affecting UI responsiveness.
- Interactive Charting: Interactive candlestick charts with EMA provide users with the tools necessary to carry out technical analysis.
- AI-Driven Insights: Successful integration of AI models to generate predictive insights, analyze patterns, and support complex user queries about market data.

Key Learnings:

- The importance of concurrent processing for developing real-time data applications.
- The power of AI when combined with real-time market data.
- How to efficiently build applications using streamlit that provide user friendly UI's for complex systems.
- The importance of error handling for preventing unexpected application failures.

Future Extensions:

- Expanded AI Capabilities: Incorporate more advanced AI models (e.g., RNNs) for more accurate predictive analysis and more complex trading strategies.
- Integration with More Data Sources: Expand data sources to incorporate APIs from other financial platforms to widen the scope of analysis.
- User Authentication and Data Storage: Implement user authentication and data storage features, to provide persistent user settings and preferences.
- Backtesting: Enable users to test various strategies using historical data.
- Automated Alerts: Build automated alert systems based on specific price movements and/or predicted market trends.

- Mobile App Version: Develop a mobile application version for users to access market data on the go.

The StockAgent platform lays a solid foundation for future development and has the potential to evolve into a more sophisticated and comprehensive financial tool. The successful implementation of AI for data interpretation, with clear and concise answers to questions asked by users, is a major stepping stone towards a fully autonomous AI driven trading platform.