Real-time Portfolio Dashboard:
Final Project Report

Authored By: Beenaa Motiram Salian

Project's Function

The Real-time Portfolio Dashboard leverages a cloud-based data pipeline to monitor stock performance and conduct technical analysis. The system integrates historical data processing with live streaming capabilities, computes technical indicators dynamically, and delivers interactive visualizations via a modern web interface. This project enables users to gain actionable insights into their portfolio's performance and market trends efficiently and in real time.

Dataset

The project utilizes datasets from AlphaVantage API, with a focus on stocks from the technology sector: AAPL, NVDA, AMD, and TSLA.
(https://www.alphavantage.co/documentation/)

1. Historical Daily Data :

- 20 years of daily OHLCV data (Open, High, Low, Close, Volume).
- Stored in Azure Blob Storage for batch processing. This will be adhoc batch ingestion.

2. Intraday Data:

- 1-minute OHLCV data simulated at 3-second intervals for real-time updates.
- Processed via Azure Event Hub and consumed by the Flask backend.

3. Technical Indicators Data :

- Metrics include Simple Moving Average (SMA), Cumulative Moving Average (CMA), and Volume Moving Average (VMA).
- Computed in real time using Azure Stream Analytics.

2.2 Pipeline Architecture (M2 - Setup)

1. Data Ingestion Layer :

- AlphaVantage API Integration: Fetches historical and intraday stock data.
- Local Producer (simulated real-time data to event hub) (per-stock basis)
- Azure Event Hubs: Streams real-time data from producers.
- Azure Blob Storage: Stores historical data files for batch processing.

2. Processing Layer :

- Azure Stream Analytics: Calculates technical indicators (SMA, CMA, VMA) from live data.
- Azure Data Factory: Processes historical data for ingestion into Azure SQL Database.
- Flask Backend: Streams processed data to the frontend via WebSockets.

3. Storage Layer :

- Azure SQL Database: Serves as a centralized repository for historical data.
- Azure Blob Storage : Real-time processed data archival
- In-memory Caching: Ensures low-latency real-time updates for the frontend.

4. Presentation Layer :

- Interactive Web Dashboard: Built with Flask, HTML, CSS, and JavaScript.
- Real-time Visualization: Provides dynamic candlestick charts, technical indicator graphs, and portfolio performance metrics.

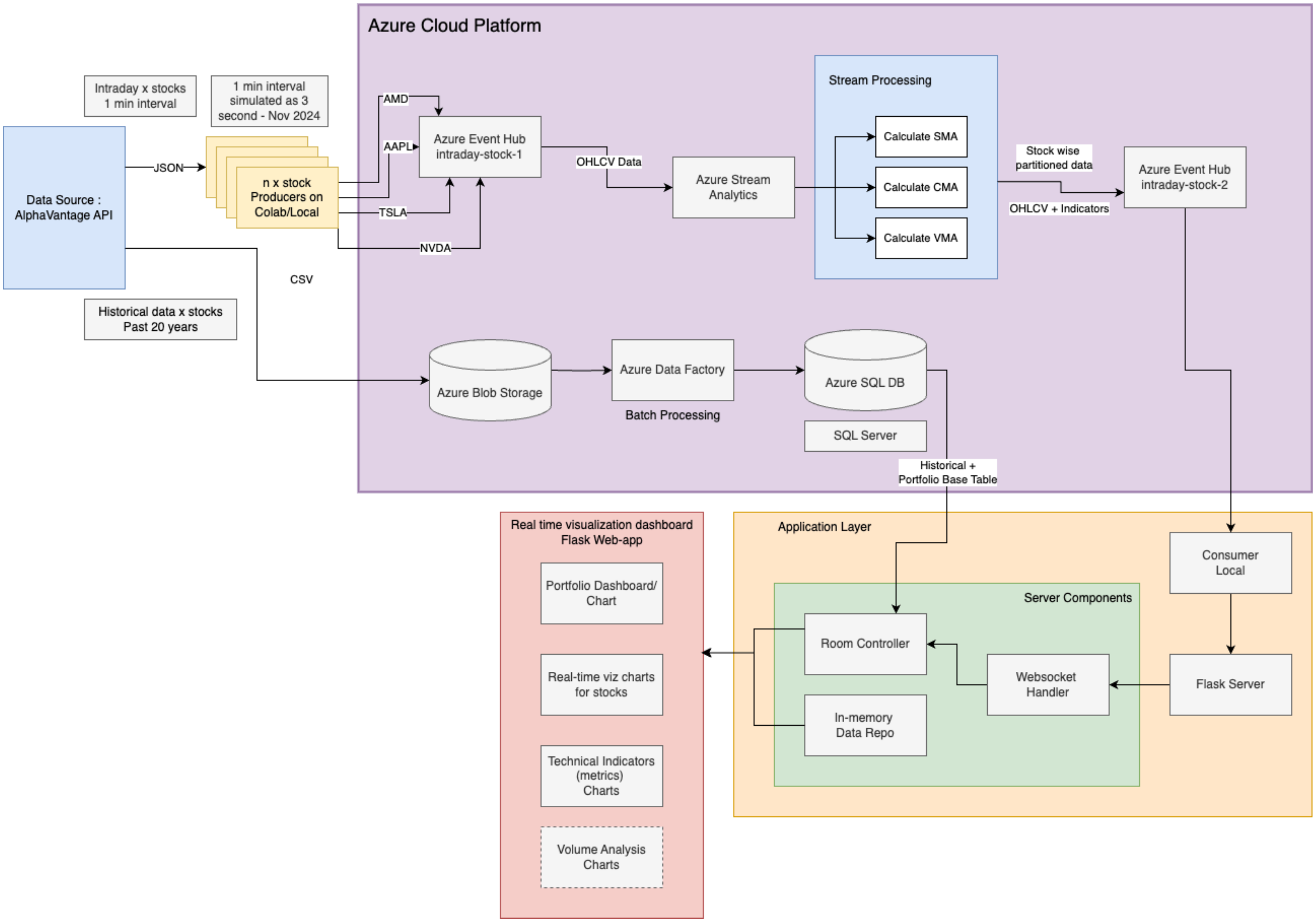

Fig : System Architecture

3. Data Exploration & Quality Assessment (M3)

a. Completeness:
   i. Verified timestamp alignment across datasets to prevent inconsistencies in real-time updates.
   ii. Addressed missing data points due to non-trading days or API limitations through interpolation.

b. Data Format Consistency:
   i. Standardized formats for timestamps, numerical values, and column structures.

ii. Ensured consistent units across metrics for seamless integration and visualization.

-

These preprocessing steps are crucial for consistent, uninterrupted visualizations.

4. Data Transformation & Provenance (M5)

1. Data Transformation Steps :
   a. Real-time Processing Pipeline:
   Raw Data (colab producer) → Event Hub 1 → Stream Analytics → Technical Indicators → Event Hub 2 → Consumer → WebSocket
      - Moving Average Calculations: Computes SMA, CMA, and VMA dynamically for each stock (Based on 30 second windows).
      - Volume Metrics: Analyzes trading volume trends for insights.

   b. Historical Data Pipeline :
   CSV Files → Blob Storage → Data Factory → SQL Database → Flask Server
      - Cleaning and Standardization: Ensures uniform formats across dates, decimals, and structures.
      - Data Ingestion: Loads cleaned data into SQL for adhoc batch processing.

2. Data Provenance and Lineage:

The data provenance for this project includes the origin, structure, and transformation of datasets used in the real-time portfolio dashboard. Key elements include:

1. Source Data: Historical (stored as CSV files) and intraday (JSON) stock data originally sourced from Alpha Vantage.
   a. Historical (2 decades/stock)
   b. Real-time (streaming from 11/04/2024 - 1 min interval intraday)
2. Transformation Details: Historical data undergoes a one-time batch ingestion into Azure SQL, while intraday data is processed as simulated real-time data, streamed through Azure Event Hub → Azure Stream processing → Websocket server.
3. Frequency and Coverage: Daily data provides a 20-year historical perspective, while intraday data is generated at 1-minute intervals but visualized every 3 seconds for the sake of this project.
4. Documentation of Simulated Data: Real-time data is simulated on Colab notebooks via Azure event-hub producers to local event-hub consumers to avoid API limitations and latency issues, ensuring seamless data flow for visualization.

5. Business Analytics Implementation (M7)

1. Portfolio Analysis : Tracks performance metrics i.e portfolio gains, losses, and overall returns.
2. Technical Analysis :
    a. Trend Identification: Charts moving average (SMA: Simple Moving Average & CMA : Cumulative Moving Average) crossovers for potential buy/sell signals.
    b. Volume Analysis : Charts Volume trades against VMA (Volume Moving Average) to understand trading activity for informed investment decisions.

6. Thorough Investigation

1. Project Viability

The Real-time Portfolio Dashboard demonstrates strong potential as a scalable and innovative solution for real-time stock analysis. Its modular, cloud-native architecture and successful integration of real-time streaming with dynamic visualizations validate its feasibility for deployment in financial analytics.

Key innovations include:

- Real-time processing with technical indicators (SMA, CMA, VMA).
- Cloud-native, scalable design leveraging Azure services.
- Simulated real-time data streaming to address API limitations.

Challenges and Limitations

- API Rate Limits: Dependency on AlphaVantage restricts scalability.
- Latency Issues: Real-time responsiveness may falter under high workloads.
- Cost: Scaling Azure services may become expensive.
- Maintenance Complexity: Multi-service integration requires careful monitoring.

Recommendations

- Incorporate predictive analytics (e.g., ML-based forecasting).
- Optimize backend processing for reduced latency.
- Explore cost-efficient alternatives to Azure services.
- Add automated alert systems for enhanced user experience.
- By addressing these limitations, the project can be scaled to handle larger datasets and deliver greater value.