SnoopTrade

The Real Time Stock Price Analysis Platform

Pratikkumar Dalsukhbhai Korat
Department of Computer Engineering
San Jose State University
San Jose, USA
pratikkumardalsukhbhai.korat@sjsu.edu

Nevil Padariya
Department of Computer Engineering
San Jose State University
San Jose, USA
nevil.padariya@sjsu.edu

Nagaraj Gireppa Kanni
Department of Computer Engineering
San Jose State University
San Jose, USA
nagarajgireppa.kanni@sjsu.edu

***Abstract*—Insider trading undermines the fairness of financial markets, creating an information asymmetry that leaves retail investors at a significant disadvantage. SnoopTrade addresses this challenge by integrating SEC EDGAR data with real-time stock price feeds to deliver predictive insights into insider trading activities. Leveraging advanced machine learning techniques, such as time-series forecasting with Prophet, the platform forecasts stock price movements influenced by insider trades. SnoopTrade offers a user-friendly interface built with React and a robust FastAPI-powered backend to ensure seamless performance. Deployed on AWS Elastic Beanstalk, it provides a scalable, reliable, and real-time solution. By empowering retail investors with actionable intelligence, SnoopTrade promotes transparency and equity in financial markets.**

CODE REPOSITORY

The source code for this project is available on GitHub: https://github.com/pratikkorat26/CMPE272GenAIProject.git.

I. INTRODUCTION

Insider trading, the unethical practice of leveraging non-public, material information for stock trading, disrupts market integrity and fairness. This behavior disproportionately benefits corporate insiders and institutional investors while leaving retail participants at a disadvantage. Retail investors often lack access to the tools and insights needed to analyze insider trading activities or understand their influence on stock prices, perpetuating a significant information asymmetry in financial markets.

Despite regulatory measures, insider trading remains a persistent challenge. Existing solutions for monitoring these activities fall short in several critical areas. Most tools fail to integrate key data sources, such as SEC EDGAR filings and real-time stock price feeds, into a unified platform. Furthermore, they rarely provide predictive analytics that can empower retail investors to anticipate stock price movements influenced by insider trades. Additionally, these solutions often lack user-friendly interfaces and actionable insights, limiting their accessibility and practical utility.

SnoopTrade addresses these challenges by bridging the gap between data, analytics, and accessibility. The platform combines insider trading data from the SEC EDGAR database with real-time stock price feeds to deliver predictive insights through advanced machine learning techniques, including time-series forecasting with Prophet [1]. SnoopTrade's React-based frontend ensures an intuitive user experience, while its FastAPI-powered [2] backend provides reliable performance. Deployed on AWS Elastic Beanstalk, the platform guarantees scalability and real-time functionality. By democratizing access to insider trading insights, SnoopTrade empowers retail investors with actionable intelligence, promoting transparency and equity in financial markets.

A. Contributions

SnoopTrade redefines financial analytics by providing a comprehensive and user-friendly solution for retail investors to analyze and act on insider trading data. Its unique contributions include:

- **Integrated Platform:**
  - Combines insider trading data from the SEC EDGAR database with real-time stock price feeds into a unified system.
  - Delivers both regulatory filings and live market data seamlessly in one platform.
- **Predictive Analytics:**
  - Uses advanced machine learning models, such as time-series forecasting with Prophet, to analyze the impact of insider trading activities on stock price trends.
  - Provides insights into how insider trades correlate with market behavior by visualization, helping retail investors make informed decisions.
- **User-Friendly Experience:**
  - Offers an intuitive React-based interface designed to make data and analytics accessible for all retail investors.
  - Empowers users with simplified tools to support informed decision-making.
- **Scalable and Reliable Architecture:**
  - Built with a robust FastAPI backend and deployed on AWS Elastic Beanstalk to ensure real-time performance and scalability.
  - Delivers consistent reliability, essential for time-sensitive financial decisions.

By uniquely integrating real-time data, predictive analytics, and usability, SnoopTrade stands as the most comprehensive and effective solution for retail investors, promoting transparency, fairness, and equity in financial markets.

II. System Architecture

A. Overview

The system architecture outlined here demonstrates a robust, scalable, and secure framework for a financial data processing platform. It combines state-of-the-art technologies like ReactJS for the frontend, FastAPI for the backend, MongoDB for data persistence, and integrates seamlessly with Google Authentication for secure user access. Furthermore, the architecture incorporates automation for data extraction workflows to maintain an up-to-date data repository. The system architecture is as shown in figure 1.

This section meticulously details the system's components, their interactions, and the thought processes behind the architectural decisions, ensuring readers grasp the elegance and efficiency of the design.

B. Architectural Overview

The architecture is divided into four distinct layers, each serving a critical role:

- **Frontend Layer**: A highly interactive user interface designed using ReactJS.
- **Authentication Layer**: Secure user authentication managed via Google Authentication.
- **Backend Layer**: A high-performance API service implemented in FastAPI.
- **Data Layer**: A MongoDB database to store and manage structured and unstructured data.

Each layer is decoupled to ensure modularity, scalability, and maintainability. All layers are hosted on AWS Elastic Beanstalk for robust deployment and optimized scalability.

C. Frontend Application

- **Framework**: ReactJS.
- **Deployment**: Hosted on **AWS Elastic Beanstalk**, configured for high availability.
- **Load Balancer**: Incoming requests are distributed evenly across an autoscaling group of React app instances.
- **Features**:
  - Provides an intuitive user interface for seamless interaction.
  - Manages client-side state and handles API requests with token-based authentication.
  - Authenticates users by interfacing with the Google Authentication service.

This decoupled frontend ensures a responsive, interactive, and reliable experience for users.

D. Authentication

- **Authentication Provider**: Google Authentication.
- **Process**:
  - Users provide their credentials through the frontend, which redirects the authentication request to Google.
  - Upon successful authentication, Google returns an access token that is stored securely in the client session.
  - The token is included in every API request sent to the backend for validation.
  - The backend retrieves the user information associated with the token and checks it against the list of registered users stored in the database.
  - Access is granted only if the user exists in the database, ensuring an additional layer of access control.

This approach guarantees secure, token-based authentication for all user interactions, while ensuring that only pre-registered users in the database are granted access, adhering to industry best practices.

E. Backend

- **Framework**: FastAPI, chosen for its speed and ease of asynchronous operations.
- **Deployment**: Hosted in a **Virtual Private Cloud (VPC)** on AWS Elastic Beanstalk for security and scalability.
- **Load Balancer**: Distributes traffic across multiple FastAPI app instances, ensuring even utilization and high availability.
- **Features**:
  - Handles business logic and API request processing.
  - Authenticates API calls using Google-provided tokens.
  - Interfaces with MongoDB for data retrieval and updates.
  - Processes real-time and batch requests from the frontend and automated workflows.

The backend is designed for resilience and high throughput, meeting the needs of data-intensive operations.

F. Data Storage

- **Database**: MongoDB.
- **Features**:
  - Stores and retrieves financial data, user information, and processed results.
  - Designed to support structured and unstructured data, making it ideal for the diversity of stored information.
  - Facilitates high-speed read/write operations to support concurrent users and workflows.

MongoDB serves as the backbone of the platform's data layer, ensuring reliability and scalability for large datasets.

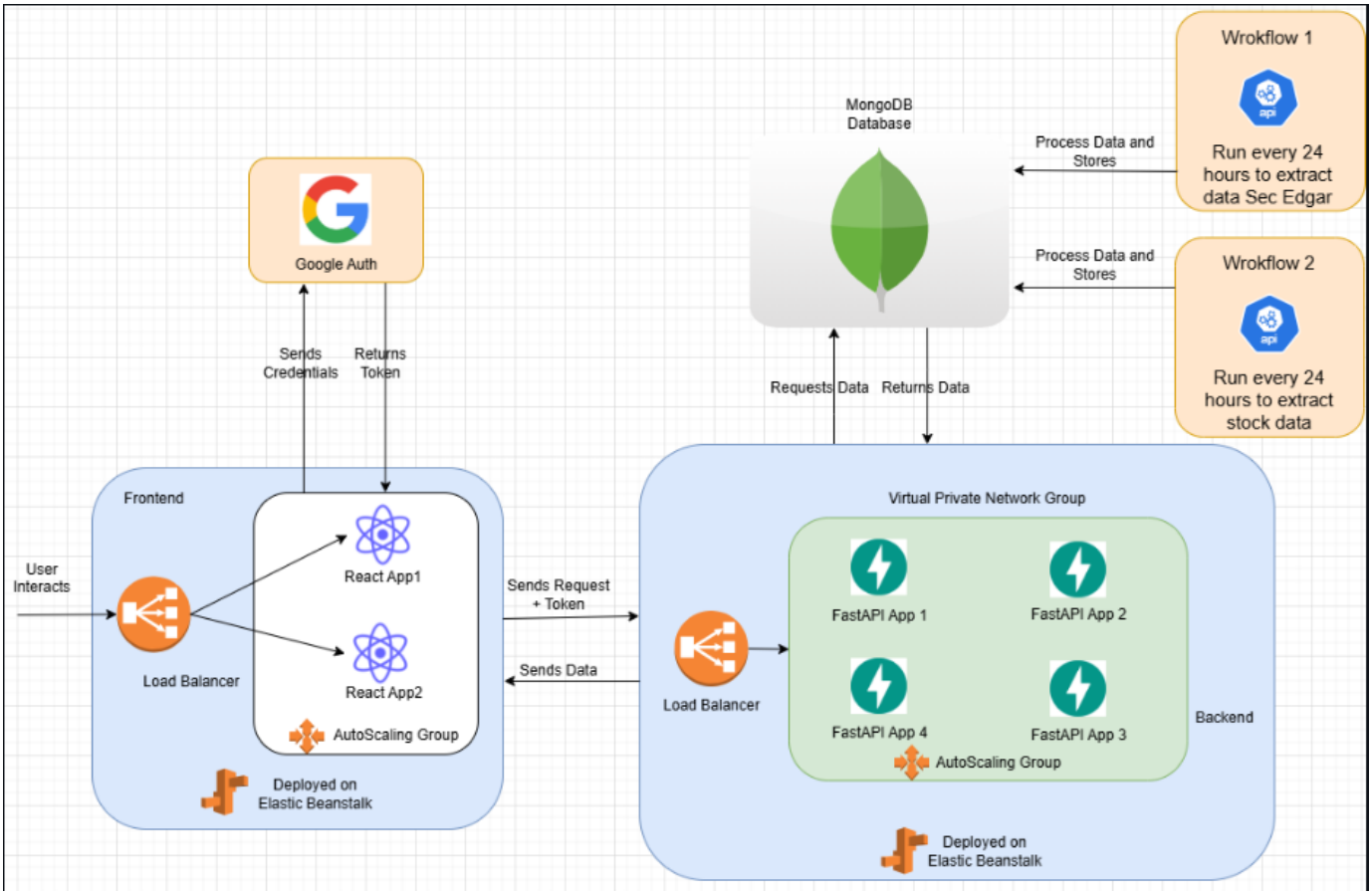

Fig. 1. System Architecture and Workflow Diagram

G. Automated Workflows

The system supports two automated workflows, executed daily to ensure that the latest financial data is available:

- **Workflow 1**: Extracts data from **SEC Edgar**, processes it, and stores the results in MongoDB.
- **Workflow 2**: Fetches stock market data, processes it, and stores it in MongoDB. This data later used for ML model learning and prediction.

Both workflows run independently of user activity, ensuring the platform remains up-to-date with minimal manual intervention.

III. Infrastructure and Deployment

- **AWS Elastic Beanstalk**:
  - Frontend and backend layers are deployed in separate Elastic Beanstalk environments, each configured with **autoscaling groups** for elasticity.
  - Autoscaling dynamically adjusts the number of instances based on traffic, optimizing resource utilization.
- **Load Balancers**:
  - Frontend: Balances user requests across React instances, ensuring reliability during peak loads.
  - Backend: Handles API requests by distributing them among FastAPI instances within the VPC.
- **Virtual Private Cloud (VPC)**:
  - Backend applications and MongoDB are hosted within a private network, ensuring isolation from external threats and unauthorized access.
  - Communication between components is restricted to private subnets for added security.

IV. Data Flow and Request Workflow

A. Frontend to Backend Workflow

1) Users interact with the **ReactJS frontend**, which manages state and sends authenticated API requests.
2) API requests include a Google Authentication token, which is decoded in the backend to verify user information against the database. If the user is not found, they are added to the database and granted access to the services.
3) Requests are routed to the backend through the load balancer, where they are processed by the FastAPI application.
4) The backend leverages a machine learning model (Prophet) for dynamic stock price predictions. The

model is trained or updated on-the-go using user-provided data and enhanced with engineered features and custom regressors.
5) The backend queries the MongoDB database as needed and integrates the ML predictions with existing data to generate enriched, processed results.
6) The processed and predicted data is returned to the frontend for user visualization and interaction.

B. Automated Workflow

1) Both workflows run at scheduled intervals (every 24 hours). We have tested this workflow runs seamlessly across 1 month. As of now, we kept it them as manual runs to avoid additional cost.
2) Workflow 1 interacts with SEC Edgar APIs, processes insider trade stock data, and updates MongoDB.
3) Workflow 2 fetches stock data using Yahoo finance API, processes it, and stores it in MongoDB.
4) Processed data becomes available for frontend requests, ensuring users always access the latest information.

V. Testing and Validation

Testing was conducted at multiple levels to ensure the system's reliability, accuracy, and maintainability.

A. API Testing

The API endpoints were thoroughly tested using **Swagger UI**. Each endpoint was validated for:

- Proper authentication and token handling.
- Accuracy and consistency of responses, including ML prediction outputs.
- Handling of edge cases, such as invalid inputs and missing data.

B. Frontend Testing

The ReactJS-based frontend underwent manual testing to validate user interface functionality and seamless interaction with the backend services. This included:

- Verifying token-based login and secure communication with the backend.
- Testing visual elements, responsiveness, and user workflows.
- Ensuring proper rendering of ML prediction outputs and data visualizations.

C. Static Analysis

Static analysis was performed on the backend codebase to evaluate code quality, detect potential vulnerabilities, and measure maintainability. The analysis revealed a quality score of **5.78** out of 10, indicating areas for improvement in coding standards and performance optimization.

D. ML-Based Performance Testing

The performance of the machine learning model was validated using real-world stock price data to assess its accuracy and reliability. Key aspects of this evaluation include:

- **Prediction Accuracy:** The model was tasked with forecasting stock prices for the next 7 days. The predicted prices consistently fell within a range of **±10 to ±15 dollars** of the actual stock prices, demonstrating its practical reliability for short-term forecasts.
- **Confidence Intervals:** The model outputs included confidence intervals for each prediction, offering additional insight into the prediction's uncertainty. For the majority of cases, actual stock prices fell within the 95% confidence interval, showcasing the robustness of the predictions.
- **Use Case Validation:** The predictions provided actionable insights for end-users, such as detecting trends influenced by insider trading activities. The model's ability to integrate additional regressors (e.g., insider trades, volatility, and RSI) enhanced its predictive performance and practical applicability.

The combination of API testing, manual frontend testing, static analysis, and ML-based performance validation ensures a robust and secure system capable of handling user interactions and dynamic data processing efficiently. The ML model's real-world validation highlights its reliability in delivering actionable insights to retail investors. We plan to conduct more comprehensive and rigorous testing and validation of this platform to ensure its robustness, reliability, and accuracy across a wider range of scenarios.

VI. Performance Monitoring

For performance monitoring, we utilize a combination of metrics, logs, and traces to ensure smooth operation and quick debugging. Amazon CloudWatch, integrated with Elastic Beanstalk, provides real-time insights into system performance, including CPU utilization, memory usage, request latency, and error rates. These metrics help monitor the health of the backend and identify bottlenecks or anomalies proactively.

Custom alarms are configured for critical thresholds, triggering automated scaling actions through Elastic Beanstalk to handle increased traffic or resource demand. Additionally, detailed logs and traces allow us to analyze application behavior, troubleshoot issues, and optimize resource usage efficiently. The combination of CloudWatch's advanced monitoring capabilities and Elastic Beanstalk's automation ensures a reliable, high-performance infrastructure that can scale dynamically to meet user demands.

VII. Challenges and Mitigations

The development of SnoopTrade posed several challenges, which were addressed as follows:

- **Handling Noisy Data**: SEC EDGAR data often contained incomplete or unstructured information. Advanced preprocessing techniques, such as feature engineering and missing value imputation, were implemented.

- **Model Optimization**: Prophet's default parameters were insufficient for stock price prediction. Custom regressors and hyperparameter tuning significantly improved accuracy.
- **Scalability**: Initial performance issues during peak traffic were resolved by leveraging AWS Elastic Beanstalk's autoscaling capabilities.

VIII. Conclusion and Future Work

SnoopTrade stands out as a pioneering platform, uniquely offering real-time training and prediction of stock price movements influenced by insider trading activities. By integrating SEC EDGAR data, real-time stock information, and advanced machine learning models within an intuitive and scalable system, SnoopTrade addresses longstanding gaps in financial market transparency. Retail investors are empowered with actionable intelligence, promoting equitable access and informed decision-making.

IX. Future Work

To further enhance the platform, we aim to:

- **Increase Security:** Implement advanced encryption standards, multi-factor authentication, and additional data integrity measures.
- **Enhanced Customization:** Provide users with greater flexibility to adjust and fine-tune machine learning models for personalized insights.
- **Expand Coverage:** Support more companies and datasets, broadening the platform's applicability and user base. Include data from different sector to check influence on the companies' stock prices.
- **Optimize Latency:** Reduce response times further to ensure seamless real-time performance, even during peak traffic.

With these planned improvements, SnoopTrade will continue to push the boundaries of financial analytics, offering unmatched features and scalability for retail investors. This vision positions SnoopTrade as a transformative tool for promoting fairness and transparency in financial markets.

X. Acknowledgment

We would like to express their sincere gratitude to Professor Rakesh Ranjan for their invaluable guidance and support throughout this project. Special thanks to the AWS team for providing cloud computing resources and tools that significantly enhanced the implementation of this work. The authors also acknowledge the use of SEC EDGAR and Yahoo Finance for data acquisition, and MongoDB Atlas and AWS for database and frontend hosting solutions. We, the authors and creators of this application, declare that we have no conflicts of interest. Additionally, we affirm that all team members contributed equally to the development and completion of this work.

References

[1] S. J. Taylor and B. Letham, "Forecasting at Scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, January 2018. [Online]. Available: https://ideas.repec.org/a/taf/amstat/v72y2018i1p37-45.html

[2] S. Ramírez, "Fastapi," https://fastapi.tiangolo.com, n.d., fastAPI framework, high performance, easy to learn, fast to code, ready for production. [Online]. Available: https://github.com/fastapi/fastapi