Stock Simulator

Anish Nimbalkar
an4338@nyu.edu

Akshat Shaha
as16655@nyu.edu

Somya Gupta
sg7885@nyu.edu

Yash Amin
yva2006@nyu.edu

***Abstract*—This report outlines the development of a cloud-based stock trading simulator designed to offer users a risk-free environment for learning and practicing stock trading strategies. By leveraging a suite of AWS services, the platform provides features such as real-time stock data integration, virtual trading with advanced order types, portfolio management, and performance analytics. Additionally, the system incorporates financial news summarization and predictive insights to enhance decision-making. This project aims to foster financial literacy, empower users to experiment with trading strategies without monetary risk, and create an engaging learning experience for both beginners and aspiring investors.**

I. Problem Statement

Learning stock trading can be a daunting task, especially for beginners who lack access to real-time data or risk-free environments. Traditional learning methods are often limited to theoretical knowledge or involve real financial risks, which discourage many from exploring the stock market. In addition, understanding the impact of financial news on stock prices and analyzing portfolio performance can be overwhelming without the proper tools. This project addresses these challenges by developing a scalable cloud-based trading simulator that combines virtual trading with advanced analytics and real-time data integration. By offering features such as financial news summarization, portfolio visualization, and performance tracking, the platform equips users with the tools they need to learn, experiment, and grow as informed investors without any financial risk.

II. Motivation

The stock market is one of the most effective tools for wealth creation, yet it remains inaccessible or intimidating to many due to its complexity and inherent risks. This project is motivated by the need to:

- Provide a secure and engaging environment for users to practice trading strategies without monetary loss.
- Encourage financial literacy by offering an interactive platform that simplifies complex concepts.
- Foster investment skill development through hands-on experience with virtual trading.
- Deliver actionable insights by integrating real-time stock data and summarizing the impacts of financial news.
- Build a community of informed investors that enables collaborative learning and shared growth.

By addressing these goals, this project seeks to empower people to take control of their financial futures in a safe and supportive environment.

III. Existing Solutions

Several platforms currently cater to stock market tracking, trading, and portfolio management. While these platforms provide valuable tools for investors and traders, they often lack the comprehensive features necessary for a risk-free learning environment or integrated educational tools. Below is an evaluation of some existing solutions and how this project differentiates itself:

- MoneyControl: MoneyControl is a popular platform for tracking investments and managing portfolios. It offers features like real-time portfolio tracking, stock watchlists, mutual fund tracking, tax planning tools, and performance reports. However, it primarily focuses on investment tracking rather than providing a risk-free trading environment. It lacks virtual trading capabilities, financial news summarization, and advanced analytics for educational purposes. Additionally, its interface can be overwhelming for beginners, limiting its utility as a learning tool.
- eToro: eToro is a well-regarded platform that offers a paper trading simulator with $100,000 in virtual funds. It allows users to practice investing in stocks, ETFs, and cryptocurrencies while testing various strategies. Although eToro provides a robust virtual trading environment, it does not focus on portfolio management analytics or financial news summarization to help users understand stock impacts. Its primary target audience includes traders already familiar with market basics.
- Thinkorswim: Thinkorswim's paperMoney simulator is another strong contender in the market. It provides real-time data for virtual trading and supports multiple asset classes like stocks, options, futures, and forex. The platform also includes advanced technical analysis tools and chart studies. However, its complexity makes it less accessible to beginners. Additionally, it lacks features like financial news summarization and collaborative learning environments.
- Interactive Brokers: Interactive Brokers offers a comprehensive trading platform with features like real-time monitoring, customizable dashboards, advanced charting tools, and options strategy labs. While it is ideal for active traders due to its extensive functionality and global market access, it does not provide a risk-free learning environment tailored for beginners or educational tools to foster financial literacy.

How This Project Differentiates Itself: While existing platforms like MoneyControl focus on portfolio tracking and

others like eToro or Thinkorswim emphasize virtual trading or advanced analytics for experienced traders, this project aims to combine the best of both worlds while addressing their limitations:

- Risk-Free Trading Environment: Unlike platforms that require real investments or cater only to experienced traders, this project offers a completely virtual trading environment where users can experiment with strategies without financial risk.
- Integrated Learning Tools: By summarizing financial news and providing insights into stock impacts, the platform fosters financial literacy in ways that existing solutions do not.
- Comprehensive Portfolio Management: Leveraging AWS DynamoDB for scalability, the platform enables real-time portfolio tracking.

IV. System Architecture

The system design for the Stock Simulator leverages a cloud-based architecture to ensure scalability, security, and high availability. The platform is built using AWS services to provide a seamless user experience while managing virtual trading, portfolio management, and analytics. The architecture is divided into multiple components, each responsible for specific functionalities. The system architecture is designed to handle a robust stock market simulation application that efficiently processes real-time data, historical stock information, and news updates while ensuring scalability and reliability. This system includes a front-end application, a load-balanced backend service, external data sources, and a central processing mechanism.

- Components
  - Frontend (React): Provides a user-friendly interface for session management and trading.
  - API Gateway: Routes HTTP requests to specific backend Lambda functions and enforces usage plans to prevent abuse.
  - AWS Cognito: Manages user authentication and issues secure tokens for API calls.
  - Lambda Functions: Implements backend business logic for user registration, session management, and trading operations.
  - DynamoDB: Serves as the database for session information, stock metadata, and news articles.
  - SQS: Handles message queuing for buy/sell stock orders, ensuring decoupled and scalable processing.
  - EC2 Instances:Host backend services, handling tasks such as data aggregation from finance APIs. They work alongside Lambda functions to ensure efficient execution of long-running processes and provide additional computational power when needed.
- Database Design : Three DynamoDB tables are used:
  - Sessions DynamoDB: Stores session details, including portfolios, orders, and wallet balances.
  - Stocks DynamoDB: Contains metadata for stocks like symbols, prices, and volume.
  - News DynamoDB: Stores stock-related news articles, including headlines, URLs, and sentiment analysis.
- Workflow Description
  - User Authentication: Users register or log in via the frontend, which routes requests through API Gateway to Lambda functions. Cognito validates credentials and issues ID and access tokens for secure API calls.
  - Stock and News Data:The frontend fetches stock metadata and news articles through dedicated APIs:
    - Fetch Stocks Lambda queries the Stocks DynamoDB table.
    - Fetch News Lambda retrieves articles from the News DynamoDB table.

    These APIs return data to the frontend for display.
  - Session Management
    - Create Session: Adds a new session to the Sessions DynamoDB.
    - Fetch Session: Retrieves session details, including active trades and portfolios.
    - Delete/Complete Session: Updates or removes session data. Completing a session calculates the remaining balance based on the user's portfolio and wallet balance.
  - Stock Trading: Trading operations are processed asynchronously using SQS:
    - Buy Stock: Validates orders and pushes them to the SQS queue. A Trigger Buy Lambda processes the message, updating the user's portfolio.
    - Sell Stock: Similar to Buy Stock, this operation validates, queues, and processes sell orders. The DynamoDB table is updated to reflect the user's portfolio and wallet balance.
- Security and Scalability
  - Cognito and API Gateway : The system uses Cognito for secure user authentication and API Gateway authorizers to enforce scopes and permissions. Tokens are securely stored on the frontend and passed in the Authorization header for each API call.
  - Throttling and Quotas : API Gateway implements usage plans to enforce request throttling and quotas, ensuring fair usage and protecting backend resources from overload.
  - Scalability with SQS and Lambda : SQS ensures reliable message handling for trading operations, while Lambda automatically scales with traffic. DynamoDB's auto-scaling capabilities ensure low-latency data storage and retrieval.
- Frontend Integration : The React frontend interacts with the backend through REST APIs. Key components include:

- Dashboard.tsx: Displays user sessions and stock data.
- jwtDecode.ts: Decodes Cognito tokens to extract user-specific information.
- apiService.ts: Manages API calls for session and trading operations.

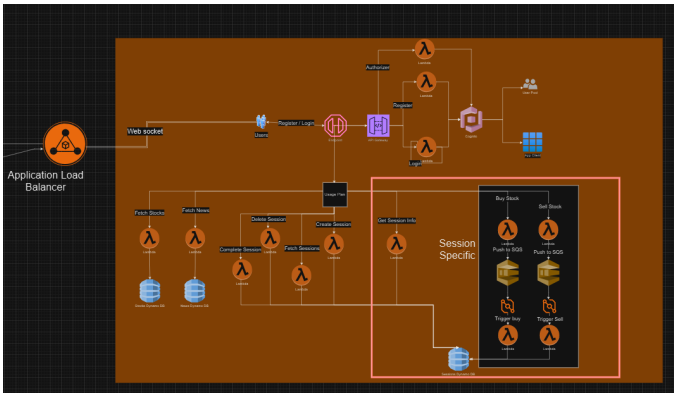

Fig. 1. User interaction with the frontend

- Backend Integration
  - Application Load Balancer (ALB):
    - The ALB distributes incoming traffic evenly among the EC2 instances within the Auto Scaling Group.
    - It ensures seamless WebSocket connections for users to receive real-time stock updates and interact with the backend application.
    - ALB dynamically manages traffic by routing requests to healthy backend instances.
  - Backend Services (Auto Scaling Group):
    - The backend services consist of multiple EC2 instances managed by an Auto Scaling Group. This ensures elasticity, allowing instances to scale up or down based on metrics like CPU utilization.
    - Each backend instance is responsible for handling user requests and emitting real-time stock updates via WebSockets.
    - Backend services are synchronized with data stored in DynamoDB for consistent state management.
  - Central Process:
    - The central process establishes a WebSocket connection with Yahoo Finance to fetch real-time stock market data. This data is continuously written to DynamoDB.
    - Historical stock data and related news are also fetched at scheduled intervals using the Polygon API and pushed into DynamoDB.
    - The central process acts as the single source of truth for the backend instances, ensuring consistent and synchronized data across the system.
  - DynamoDB:
    - DynamoDB serves as the primary database for storing real-time stock updates, historical data, and stock-related news.
    - It is optimized for high availability and low latency, supporting the real-time nature of the application.
  - Advantages of DynamoDB in This Use Case:
    - Low Latency: DynamoDB ensures millisecond response times, which is critical for real-time stock data updates. For example, as new stock prices are received from Yahoo Finance, the database can instantly store and retrieve the data for backend instances.
    - Scalability: DynamoDB scales automatically to handle large volumes of data, such as high-frequency stock price updates during market hours.
    - High Availability: DynamoDB's distributed architecture ensures that the application remains operational even during infrastructure failures, making it suitable for mission-critical stock market applications.
    - Flexible Data Models: DynamoDB supports key-value and document data models, which makes it ideal for storing diverse types of data like real-time stock prices, historical price trends, and news articles.
    - Integration with AWS Ecosystem: DynamoDB integrates seamlessly with other AWS services like Lambda, SQS, and CloudWatch. For example, stock data updates can trigger a Lambda function for further processing or analytics.
  - Example Use Case:
    - Real-time stock updates are written to DynamoDB, and backend instances query the latest prices to emit to connected clients.
    - News articles fetched from the Polygon API are indexed with stock symbols and stored in DynamoDB. When a user queries for a specific stock, the backend can quickly retrieve relevant news articles.
  - External Data Sources:
    - Yahoo Finance: A WebSocket connection is established to retrieve real-time stock data.
    - Polygon API: Used to fetch historical stock data and periodic stock-related news.
- Data Flow and Synchronization
  - Real-Time Data Flow:
    - The central process retrieves real-time data from Yahoo Finance and writes it into DynamoDB.
    - Backend instances establish independent WebSocket Connections with Yahoo Finance and emit these updates to connected clients over WebSocket connection established between the WebClient and the Backend instances. This ensures high

availability of real-time stock market data and overcomes single point of failure.

  - Historical and News Data Flow:
    * The central process periodically fetches historical data and news from the Polygon API.
    * This data is pushed into DynamoDB, where lambda functions can retrieve it to serve user queries.
  - Load Balancing and Scalability:
    * The ALB ensures that traffic is distributed among backend instances.
    * The Auto Scaling Group dynamically adjusts the number of backend EC2 instances based on the application's load and resource utilization.
  - WebSocket Synchronization:
    * Each backend instance establishes its own WebSocket connection with Yahoo Finance.
- Scalability and Fault Tolerance
  - The use of an Auto Scaling Group ensures the system can handle high traffic volumes by scaling horizontally.
  - DynamoDB provides a reliable and highly available data storage solution.
  - The ALB automatically routes traffic to healthy backend instances, ensuring minimal downtime and enhanced fault tolerance.

This architecture is designed to handle the high demands of real-time stock data processing while ensuring seamless scalability, high availability, and a consistent user experience. It leverages AWS services and industry best practices to provide a robust and efficient stock market simulation platform. The integration of DynamoDB allows for fast, reliable, and scalable data handling, which is pivotal for the system's performance and reliability.

V. Future Work

While the current implementation focuses on creating a solid foundation for virtual trading and portfolio management, there are several opportunities for future improvements.

- Machine Learning Integration:
  - Implement predictive analytics using Amazon SageMaker to forecast stock price trends based on historical data.
  - Develop personalized investment recommendations that are tailored to user behavior and preferences.
- Gamification Features:
  - Introduce leaderboards, challenges, and rewards to encourage user engagement.
  - Add educational modules or quizzes on investment strategies.
- Collaborative Features:
  - Create forums or chat rooms where users can discuss strategies or share insights.

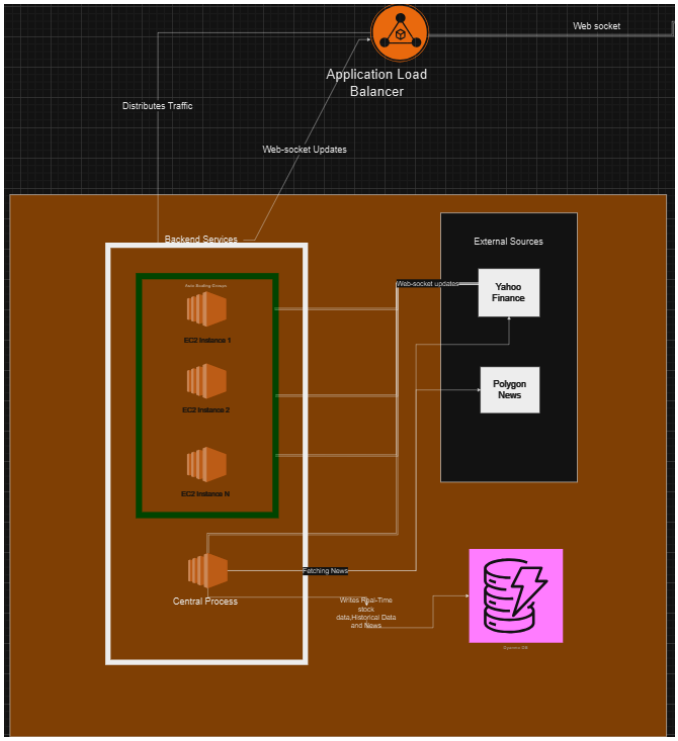

Fig. 2. Backend servers and APIs

  - Enable group-based simulations where users can compete or collaborate in virtual investment scenarios.
- Expanded Data Sources:
  - Integrate additional financial APIs (e.g., Bloomberg Finance or Google Finance) to provide more comprehensive market coverage.
  - Include global market data for international trading simulations.
- Advanced Order Types:
  - Add support for more complex order types like trailing stops or conditional orders.
  - Provide users with the ability to simulate trades in options and futures markets.
- Enhanced Administrative Tools:
  - Expand the administrative dashboard with more granular analytics on user behavior and platform performance.
  - Use Amazon QuickSight's advanced visualization capabilities for deeper insights into platform usage trends.

VI. Results

The implementation of the cloud-based stock trading simulator has yielded promising results, demonstrating the platform's potential to provide an engaging and educational experience for users. Below are the key outcomes achieved during the development and testing phases:

- Functional Virtual Trading System
  - Users can perform virtual trades successfully executed through AWS Lambda functions.
  - The system processes transactions while simulating real-world, ensuring a seamless trading experience.
- Real-Time Data Integration
  - Stock price data is fetched from financial APIs (Yahoo Finance), providing users with up-to-date market information.
  - Real-time portfolio updates ensure that users can monitor their investments accurately.
- Financial News Summarization
  - The simulator uses NEWS APIs which allows for the extraction and summarization of financial news, offering users insights into potential stock impacts.
- Portfolio Management and Analytics
  - User portfolios are stored in Amazon DynamoDB, enabling fast and scalable data retrieval.
  - Performance metrics, such as portfolio growth rate, profit/loss analysis, and risk exposure, are calculated using AWS Lambda and are available to the user to make informed decisions.
  - Stock charts and graphs provide users with a clear understanding of their trading performance.
- Scalability and Security
  - The use of AWS services ensures that the platform is both scalable to accommodate growing user bases and secure through Amazon Cognito's authentication mechanisms.

These results demonstrate the platform's ability to meet its objectives of fostering financial literacy, providing a risk-free trading environment, and offering advanced analytics for portfolio management. Future iterations will build on this foundation to incorporate additional features, enhance user experience, and expand the platform's capabilities.

VII. GITHUB URL

Github Link