Project Title: Stocker: A Cloud-Based Stock Trading Platform

Under the guidance of

Mrs. Sam Sumana , APSSDC
Mr.Anilkumar Varanasi ,APSSDC

- **Tungala Lakshmi Venkat Sai (23981A42I0)** – Raghu Engineering College
- **Koyyana Likhitha (22A51A0531)** – Aditya Institute of Technology and Management
- **Dabbeeru Keerthi (23A51A0484)**– Aditya Institute of Technology and Management
- **Gangu Nikhil Sai (22A51A0581)** – Aditya Institute of Technology and Management
- **Konatham Shree Govardhan Reddy (23A55A0513)** – Aditya Institute of Technology and Management

Course: B.Tech Computer Science .
Batch: 21 ,2022-2026.

Abstract

Stocker is a serverless, cloud-based virtual stock trading platform designed to enhance financial literacy and empower new investors by providing a risk-free environment for trading. In an increasingly complex financial landscape, Stocker serves as an educational tool that allows users to engage with the stock market without the fear of losing real money. The platform simulates real-world trading experiences, enabling users to register, search for stocks, place virtual buy and sell orders, and manage their portfolios effectively.

The user-friendly interface simplifies the complexities of stock trading for beginners, allowing them to navigate various features with ease. Users can access real-time data on stock prices, historical performance, and market trends, which are crucial for making informed trading decisions. Additionally, Stocker includes a feature that sends trade alerts through AWS Simple Notification Service (SNS), keeping users informed about significant market movements and stock price changes. This timely information allows users to react quickly to market conditions, enhancing their learning experience and improving their trading skills.

Built entirely using Amazon Web Services (AWS) Free Tier services, Stocker showcases the power and scalability of modern serverless architecture. The platform utilizes AWS Cognito for user authentication, ensuring a secure registration process. The API Gateway facilitates communication between the front-end and back-end services, while AWS Lambda hosts the Flask API, enabling efficient execution of code in response to events.

Data storage is managed through AWS DynamoDB, providing fast and predictable performance for user data, including portfolios and transaction histories. AWS CloudWatch is utilized for monitoring and logging, offering insights into application performance and user engagement. The use of AWS S3 for storing static assets further enhances the platform's efficiency and scalability.

In addition to its technical features, Stocker emphasizes community and support, including educational resources and forums where users can share experiences and learn from one another. As Stocker evolves, future enhancements may include advanced analytics, machine learning algorithms for personalized trading recommendations, and expanded educational content. By continuously adapting to user needs, Stocker aims to remain at the forefront of virtual trading platforms, making financial education accessible to everyone.

In conclusion, Stocker represents a significant advancement in democratizing financial knowledge and investment opportunities. By providing a safe, engaging, and educational platform for virtual trading, it empowers new investors to take control of their financial futures. The innovative use of serverless architecture and AWS services not only enhances the platform's functionality but also sets a benchmark for future developments in financial technology. Through Stocker, users can embark on their investment journeys with confidence, equipped with the knowledge and skills necessary to navigate the complexities of the stock market.

Introduction

Problem Statement:

The inaccessibility of risk-free environments for beginners to practice stock trading significantly hinders financial literacy and creates a barrier to entry into the investment world. Many novice investors face overwhelming challenges when attempting to navigate the complexities of the stock market, often leading to hesitation and reluctance to engage in trading activities. Traditional educational methods, which typically involve theoretical learning without practical application, fail to equip individuals with the necessary skills and confidence to make informed investment decisions.

Without a safe space to experiment with trading strategies, beginners are left to choose between risking real capital in live markets or refraining from investing altogether. This dilemma not only stifles personal financial growth but also perpetuates broader economic disparities, as individuals from underserved communities are disproportionately affected by the lack of accessible financial education. The absence of comprehensive simulation platforms that replicate real market conditions further exacerbates the issue, leaving users unprepared for the emotional and psychological challenges of actual trading.

Moreover, the fear of losing money prevents many potential investors from taking their first steps into the market, resulting in a cycle of inaction that can have long-term consequences on wealth accumulation. The current landscape lacks integrated educational resources that guide users through the complexities of trading, thereby limiting their ability to develop essential skills.

As a result, there is a pressing need for innovative solutions that provide a risk-free, engaging, and educational environment for novice investors. By addressing these challenges, we can empower individuals to build their financial literacy, gain confidence in their trading abilities, and ultimately foster a more inclusive investment landscape that encourages participation from all demographics.

Objectives:

- Create a simulated platform for stock trading.
- Enable virtual transactions with user-friendly interfaces.
- Notify users about trade activities.
- Promote financial awareness through interaction and practice.

Technology Stack Used

Frontend:

- HTML/CSS/JavaScript (hosted on S3 for static assets)

Backend:

- AWS Lambda (Flask API)
- AWS API Gateway
- AWS Cognito (Authentication)

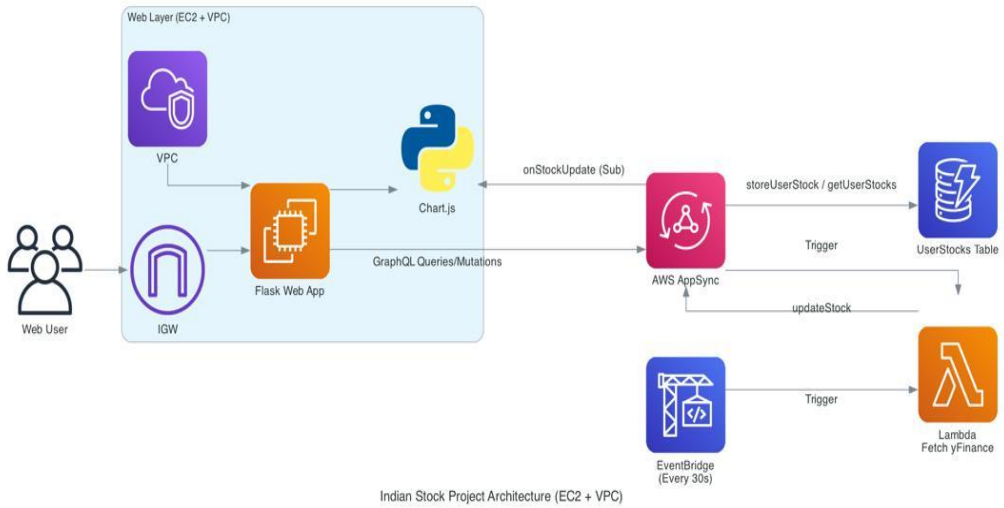
Database & Messaging:

- AWS DynamoDB (Portfolio & Trades)
- AWS SNS (Notifications)

Monitoring:

- AWS CloudWatch (Logs & Alerts)

System Design / Architecture



The platform follows a serverless architecture on AWS Free Tier:

1. **Investor** interacts via the web interface.
2. **Cognito** handles authentication and user sessions.
3. **API Gateway** routes requests securely to the backend.
4. **AWS Lambda** hosts the Flask-based API that processes all logic.
5. **DynamoDB** stores user data like portfolios and transactions.
6. **SNS** delivers trade confirmation alerts.
7. **CloudWatch** captures system logs for monitoring.
8. **S3** hosts static frontend assets (HTML/CSS/JS).

This modular architecture enables cost-efficiency, easy scaling, and clean separation of concerns.

Implementation

Key Features Implemented (MVP):

1. User Registration & Login via AWS Cognito
Secure authentication system for user accounts
Enables personalized trading experiences and data protection

2. Virtual Currency Allocation
Provides users with simulated funds for risk-free trading
Customizable starting balances for different experience levels

3. Stock Search Functionality
Simulated market data with real-world stock symbols
Historical price charts and latest "market" information

4. Virtual Buy/Sell Order Execution
Supports market and limit order types
Realistic order fulfillment with simulated price fluctuations

5. Portfolio Dashboard
Real-time tracking of holdings and performance metrics
Detailed breakdowns of portfolio allocation and gains/losses

6. SNS Trade Notifications
Instant alerts for executed trades and price movements
Configurable notification preferences via email/SMS

Code Snippet Example (Lambda Flask API - Buy Order):

```
@app.route('/buy', methods=['POST'])
def buy_stock():
    data = request.json
    user_id = data['user_id']
    stock_symbol = data['stock_symbol']
    quantity = int(data['quantity'])

    # Example: Deduct virtual cash and add to portfolio
    # DynamoDB update and SNS notification here
    return jsonify({'message': 'Buy order successful'})
```

User Registration and Login (AWS Cognito + Flask):

```
@app.route('/register', methods=['POST'])
def register():
    client.sign_up(
        ClientId=APP_CLIENT_ID,
        Username=request.form['email'],
        Password=request.form['password'],
        UserAttributes=[{'Name': 'email', 'Value': request.form['email']}]
    )
    return redirect('/login')
```

Notification (AWS SNS):

```
sns.publish(
    TopicArn=SNS_TOPIC_ARN,
    Message=f"Booking confirmed for {email}",
    Subject="TravelGo Booking Confirmation"
)
```

Results/Output:

1. Seamless User Login and Registration

Users can easily create accounts and log in, ensuring a smooth onboarding experience. The secure authentication process enhances user trust and data protection.

2. Interactive Stock Data Display

Users can search for stocks and view detailed, interactive data, including historical performance and current prices. This feature enhances user engagement and informed decision-making.

3. Recorded Virtual Transactions in DynamoDB

All virtual buy and sell transactions are accurately logged in DynamoDB, ensuring reliable data persistence. This allows for quick retrieval and analysis of trading history.

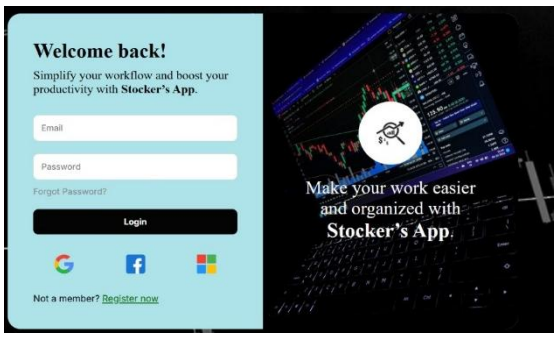
4. Instant Trade Notifications via SNS

Users receive real-time alerts for executed trades and significant price changes through SNS. This feature keeps users informed and enables timely trading decisions.

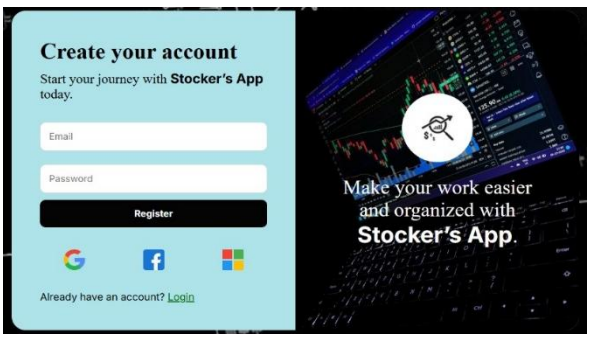
5. Updated Portfolio Balances and Holdings

The portfolio dashboard reflects real-time updates of stock holdings and virtual currency balances. Users can easily track their performance and make informed adjustments to their strategies.

Screenshot: [Include screenshot of terminal output of Lambda logs]

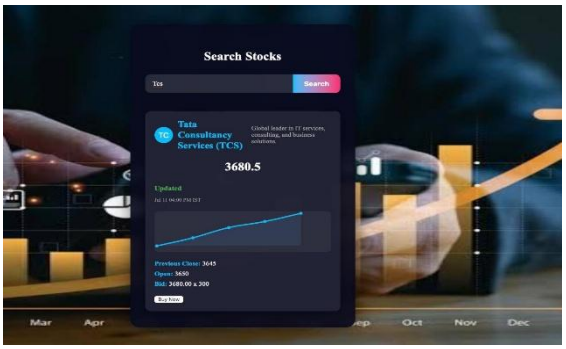


Login Page



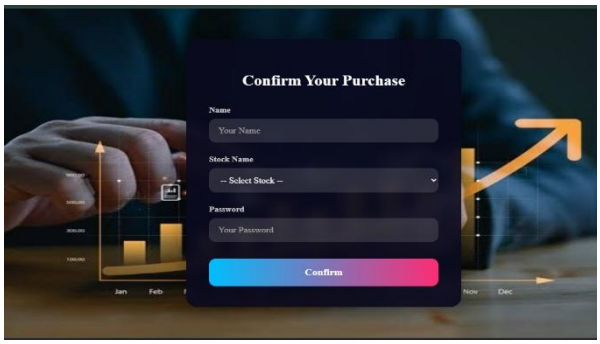
Sign Up Page

Dash Board

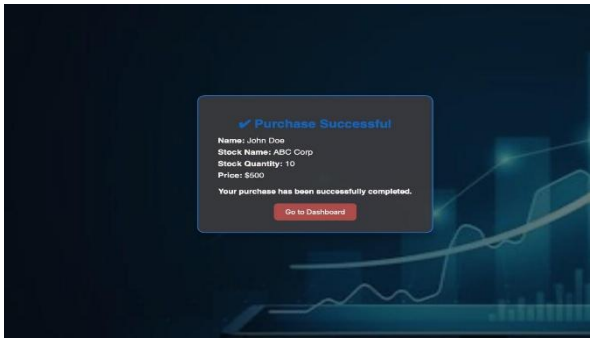


Search Page

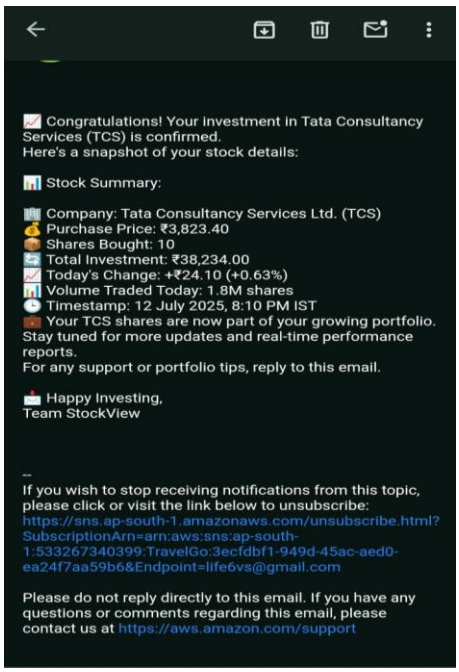
Buying Page



Conformation Page

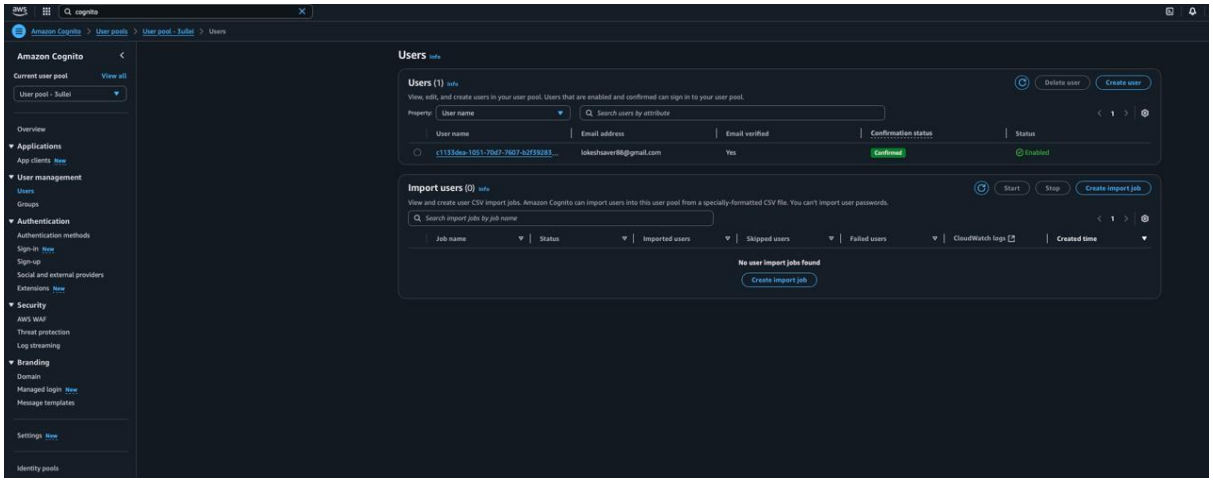


Buying Successful Page

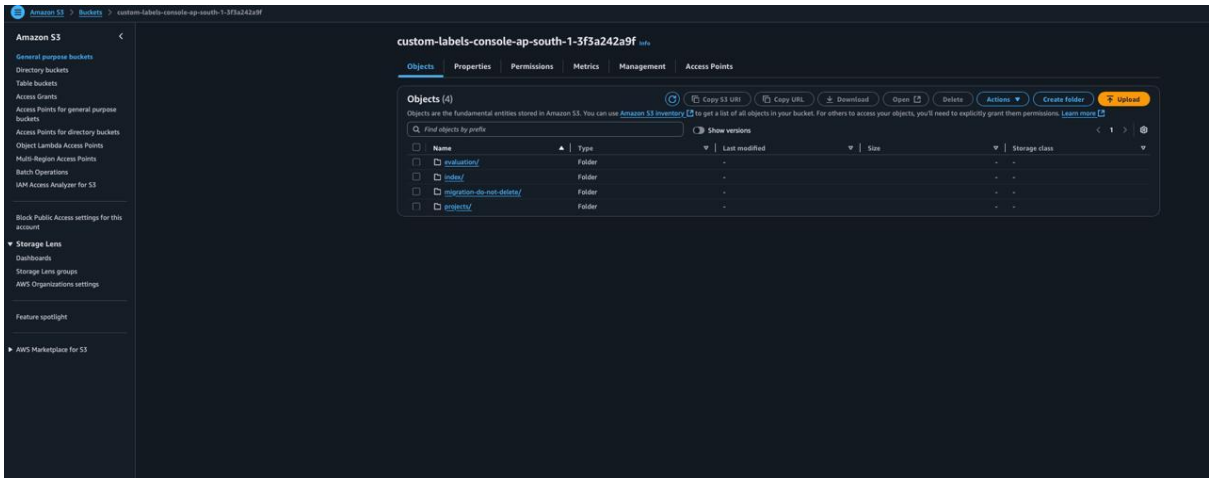


Conformation Email

DynamoDB



Cognito Console Implementation



S3 files

Conclusion & Future Scope

Conclusion: Stocker demonstrates how cloud technologies can enable educational fintech tools. By leveraging serverless architecture, we've built a secure, responsive, and scalable stock trading simulation platform with zero cost using AWS Free Tier.

Future Scope:

- Real-time market data integration (via external APIs)
- Advanced charting with historical price data
- Custom watchlists and news feeds
- Additional order types (e.g., limit, stop-loss)
- Android/iOS mobile applications
- Gamified features for improved engagement

References

- AWS Documentation: https://docs.aws.amazon.com/
- Flask Framework: https://flask.palletsprojects.com/
- DynamoDB Guide: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html
- SNS Guide: https://docs.aws.amazon.com/sns/latest/dg/welcome.html
- CloudWatch: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html