Project: Virtual Stock trading simulator

Notes:

- FE doesn't have to be complex (Simple)
- CloudFront runs on HTTPS, potential issues connecting to API gateway (HTTP) etc.
- HTML,CSS, JS is enough
- Proper async processing i.e. SQS etc
- Caching in dynamoDB, api calls if there is a rate limit
- Gemini API alternative to Amazon Comprehend
- Follow Cloud Native principles (Scalable, Concurrency), Efficiency (Fetching Report, caching etc) <- Grading
- Final submission: Report in IEEE format, all the code, github link - create a release and submit that, recording of the demo in youtube

High-Level Architecture Overview

1. Frontend (Client Layer)

- **React.js / Next.js Web App**
  - User interface for sign-up, trading dashboard, portfolio view, analytics, and news feed.
  - Deployed on **AWS Amplify** or **S3 + CloudFront** for scalability and SSL support.
  - Integrates securely with AWS Cognito for authentication and APIs via API Gateway.

2. Backend (Application Layer)

- **Amazon API Gateway (Mathew)**
  - Entry point for all frontend requests.
  - Routes requests to the appropriate Lambda functions.
  - Handles rate limiting, logging, and authorization via Cognito.
- **AWS Lambda (Microservices)**
  - Stateless backend functions handling:
    1. **User Portfolio Operations** – buy/sell, stop-loss, GTT orders.
    2. **Stock Data Fetching** – pulls from **Yahoo Finance API** or **Alpha Vantage**.
    3. **Financial News Processing** – extracts and summarizes relevant news.
    4. **Performance Analytics** – calculates ROI, volatility, and comparisons.
    5. **Feedback Processing** – consumes user messages from SQS.
    6. **Admin Analytics Service** – prepares metrics for QuickSight.
- **Amazon Comprehend + Lambda**
  - Performs **NLP-based sentiment and summarization** on financial news.

- Optionally integrate **AWS Bedrock (Claude/GPT models)** for enhanced summaries.

3. **Data Layer**

- **Amazon DynamoDB (Rodrigo)**
  - Stores user data, portfolios, transactions, and order history.
  - Uses **partition keys** for userID and **sort keys** for timestamps.
  - TTL configured for temporary orders (e.g., good-till-triggered).
- **Amazon S3**
  - Stores static files, logs, and exported analytics reports.
- **Amazon QuickSight**
  - Provides **interactive BI dashboards** for both user analytics and admin monitoring.
- **Amazon SQS**
  - Message queue for asynchronous feedback and notifications.
  - Decouples user actions from backend processing (feedback, alerts, etc.).

4. **Integration & Data Flow**

(End-to-End User Journey)

1. **User signs up/login** → Cognito validates and issues tokens.
2. **Frontend calls API Gateway** → passes token for authorization.
3. **Lambda (Trading Handler)** performs buy/sell → writes transaction to DynamoDB.
4. **Lambda (Stock Data Fetcher)** periodically pulls prices → updates cache (S3 or DynamoDB).
5. **Lambda (Analytics Engine)** computes performance metrics → stores results.
6. **Comprehend Service** fetches and summarizes financial news → displayed on UI.
7. **QuickSight Dashboards** visualize portfolio growth, market trends, and system KPIs.
8. **SQS + Lambda (Feedback)** handles user suggestions and issue reports asynchronously.