Nagamedha Sakhamuri *Panther#: 002828574* *nsakhamuri1@student.gsu.edu*

Project Phase 2
Stock Order Management System (SOMS)

Summary:

The **Stock Order Management System (SOMS)** is an innovative platform that simplifies the process of buying and selling stocks. Developed with **Angular** for the frontend, **Java Spring** for the backend, and **MariaDB** as the database, SOMS focuses on providing a seamless experience for users who want to manage their stock orders efficiently.

Usefulness:

In today's fast-paced financial environment, managing stock orders manually can be tedious and error-prone. SOMS addresses this challenge by automating key processes like user registration, login, placing orders, and canceling them. The application executes orders using **stored procedures** in MariaDB, ensuring that they are handled based on availability and timestamps in a **First Come First Serve (FCFS)** manner.

You might find **similar platforms** like the **NSE** or trading apps like **Zerodha** and **Robinhood**. However, SOMS **stands out** by focusing on database-driven order processing, making it faster and more reliable. Additionally, SOMS goes beyond standard features by incorporating **advanced functionalities**, particularly in data analysis using aggregate functions. This enhancement allows users to gain deeper insights into their stock portfolios and trading history.

Furthermore, the implementation of a **First-Come-First-Served (FCFS)** approach through database locks ensures a fair and efficient stock allocation process. This innovative solution guarantees that the first user to initiate an order will be granted access to the available stocks ahead of others. This preventive measure addresses potential race conditions or conflicts in stock allocation, thereby ensuring a seamless and equitable fulfillment of orders.

Realness: The data used in SOMS will include:

- **Stock Information:** This consists of stock symbols, prices, and availability.
- **User Details:** Collected during the registration process.

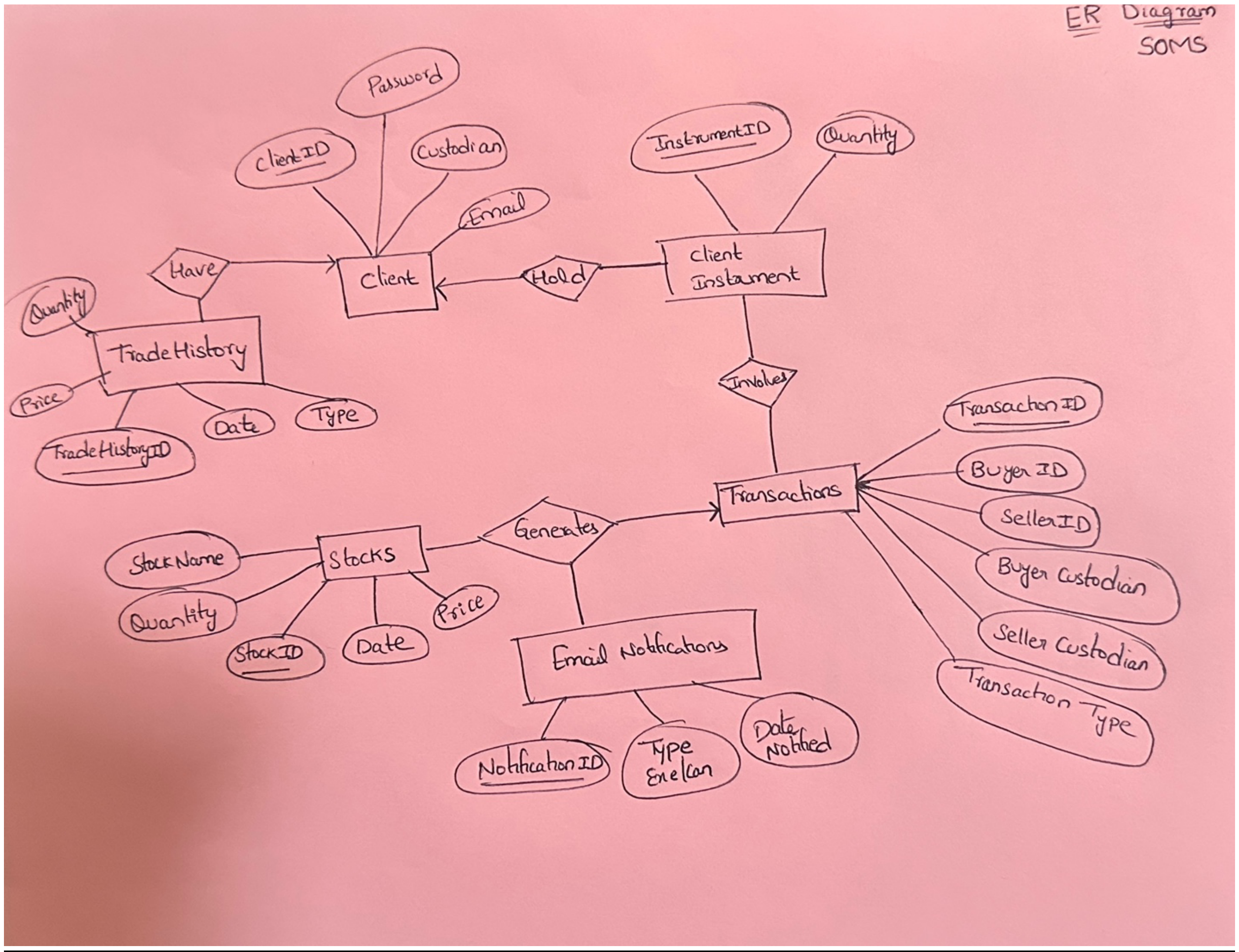
This data will be simulated for demonstration purposes, allowing users to experience the full functionality of the application without needing access to live market data.

Functionality: SOMS provides a range of features designed to enhance user experience:

1. **User Registration**: Clients register with information such as ClientID, Custodian, and Email.
2. **Portfolio Management**: Each client can manage multiple stocks, with details stored in **Client Instrument**.
3. **Buy and Sell Stocks**: Clients can initiate **buy** or **sell** transactions, which are recorded in the **Transactions** entity.
4. **Trade History Logging**: All transactions are logged in **Trade History**, allowing for easy tracking of past trades.
5. **Email Notifications**: Clients receive notifications regarding their transactions through **Email Notifications**.
6. **Real-Time Stock Availability**: The system tracks the availability and details of stocks using the **Stocks** entity, updating it with every transaction.

Team Members:
Nagamedha Sakhamuri – 002828574 - nsakhamuri1@student.gsu.edu
Ttanvi Tummapudi – 002843956 - ttummapudi1@student.gsu.edu

ER Diagram:



Assumptions:

Entities and Attributes:

1. **Client**
   - ClientID (Primary Key)
   - Custodian
   - Email
   - Password
2. **Client Instrument**
   - InstrumentID (Primary Key)
   - Quantity
   - ClientID (Foreign Key to Client)
3. **Transactions**
   - TransactionID (Primary Key)

- Buyer Custodian
- Seller Custodian
- Transaction Type
- BuyerID (Foreign Key to Client)
- SellerID (Foreign Key to Client)
- InstrumentID (Foreign Key to Client Instrument)

4. **Stocks**
   - StockID (Buy/Sell) (Primary Key)
   - ClientID (Foreign Key to Client)
   - InstrumentID (Foreign Key to Client Instrument)
   - Buy/Sell Date
   - Price
   - Stock Name
   - Quantity

5. **Trade History**
   - TradeHistoryID (Primary Key)
   - ClientID (Foreign Key to Client)
   - InstrumentID (Foreign Key to Client Instrument)
   - Trade Date
   - Trade Type (Buy/Sell)
   - Quantity
   - Price

6. **Email Notifications**
   - NotificationID (Primary Key)
   - TransactionID (Foreign Key to Transactions)
   - Notification Type (Order Execution/Order Canceled)
   - Notification Date

Relationships:

- **Client - Client Instrument:** 1:M (One Client can hold multiple Instruments)
- **Client Instrument - Transactions:** M:M (Multiple Transactions can involve multiple Instruments)
- **Transactions - Stocks:** 1:M (Each Transaction involves multiple Stocks)
- **Transactions - Email Notifications:** 1:M (Each Transaction can generate multiple Email Notifications)
- **Client - Trade History:** 1:M (One Client can have multiple Trade History records)

Team Members:
Nagamedha Sakhamuri – 002828574 - nsakhamuri1@student.gsu.edu
Ttanvi Tummapudi – 002843956 - ttummapudi1@student.gsu.edu