Project Phase 3

Stock Order Management System (SOMS) - Development Plan

1. Relational Schema of SOMS Database:

PFB the relational schema design for SOMS database, including all keys and dependencies.

Client

- ClientID (Primary Key)
- Custodian
- Email
- Password

Client Instrument

- InstrumentID (Primary Key)
- Quantity
- ClientID (Foreign Key to Client)

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)

Stocks

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

Trade History

- TradeHistoryID (Primary Key)
- ClientID (Foreign Key to Client)
- InstrumentID (Foreign Key to Client Instrument)
- Trade Date
- Trade Type (Buy/Sell)

Team Members:

Nagamedha Sakhamuri – 002828574 - nsakhamuri1@student.gsu.edu

Ttanvi Tummapudi – 002843956 - ttummapudi1@student.gsu.edu

- Quantity
- Price

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)

2. Choice of Databases and Software Platforms/Languages:

The system uses **MariaDB** for the database, **Java with SpringBoot** for backend development, and **HTML, CSS, TypeScript, AngularJS** for the frontend. These technologies ensure a robust, secure, and interactive platform for managing stock transactions.

- ***Database:*** **MariaDB -** An open-source relational database that handles large volumes of stock data and transactions efficiently, providing reliability and scalability.

- ***Backend:*** **Java with SpringBoot -** A powerful server-side framework that processes user requests and stock transactions securely, offering a scalable infrastructure.

- ***Frontend:*** **HTML, CSS, TypeScript, AngularJS -** These tools create a modern, interactive user interface. AngularJS allows for dynamic, real-time updates, while HTML/CSS provide structure and styling, ensuring the application is intuitive and visually appealing for users managing their stock orders.

3. Data Source:

The data for the application will primarily come from user input and interactions within the application. The raw data is taken from the National Stock Exchange of India.

- **User Input:** Clients provide data through registration, portfolio management, and buy/sell transactions within the application.

- **Stock Market Data from the National Stock Exchange of India (NSE):** This includes stock symbols, real-time prices, and stock availability to support accurate trading and portfolio management.

4. Labor Division:

- **Database Design**: Both members will collaborate on designing the database schema, ensuring proper relationships and dependencies between entities.
- **Backend Development**: Responsibilities will be divided. One member will focus on user authentication and input validation, while the other will handle stock transactions and stock recommendations.
- **Frontend Development**: Frontend tasks will be split, with one member creating user interfaces and data presentation, and the other handling notifications and making the interface responsive.
- **Data Entry**: Both members will work together to enter the initial data required for the system.
- **Testing and Quality Assurance**: Both members will participate in testing, debugging, and ensuring the application is robust and functional.

5. Project Timeline with Milestones:

PFB weekly project timeline with clear milestones:

- **Oct 18 – Oct 24 (1 week): Database Design**
  - Design the database schema and define relationships between tables.
  - Set up the database structure in MySQL/MariaDB and populate initial test data.
- **Oct 25 – Oct 29 (5 days): Initial Data Entry**
  - Gather and input all required stock and user information into the database.
  - Conduct data validation to ensure accuracy and completeness.
- **Oct 30 – Nov 6 (1 week): Backend Development - User Authentication**
  - Develop user authentication features, including registration and login.
  - Implement input validation to enhance security.
- **Nov 7 – Nov 13 (1 week): Backend Development - Stock Transactions**
  - Build functionalities for buying and selling stocks.
  - Implement portfolio tracking for users.
- **Nov 14 – Nov 17 (4 days): Backend Development - Stock Recommendations**
  - Develop a recommendation engine for personalized stock suggestions based on user data.
- **Nov 18 – Nov 22 (5 days): Frontend Development - User Interface**

- Design and develop the user interface, focusing on usability and real-time updates for stock transactions.

- **Nov 23 – Nov 25 (3 days): Frontend Development - User Notifications**
  - Implement a notification system to alert users of transaction statuses and system updates.

- **26 – Nov 28 (3 days): Integration of Frontend and Backend**
  - Connect frontend components with backend APIs to ensure seamless functionality.

- **Nov 29 – Dec 1 (3 days): System Testing and Debugging**
  - Conduct comprehensive testing of the entire system to identify and resolve issues.

- **Dec 2 (1 day): Final Quality Assurance and Documentation**
  - Perform final checks and prepare project documentation for submission.

*****