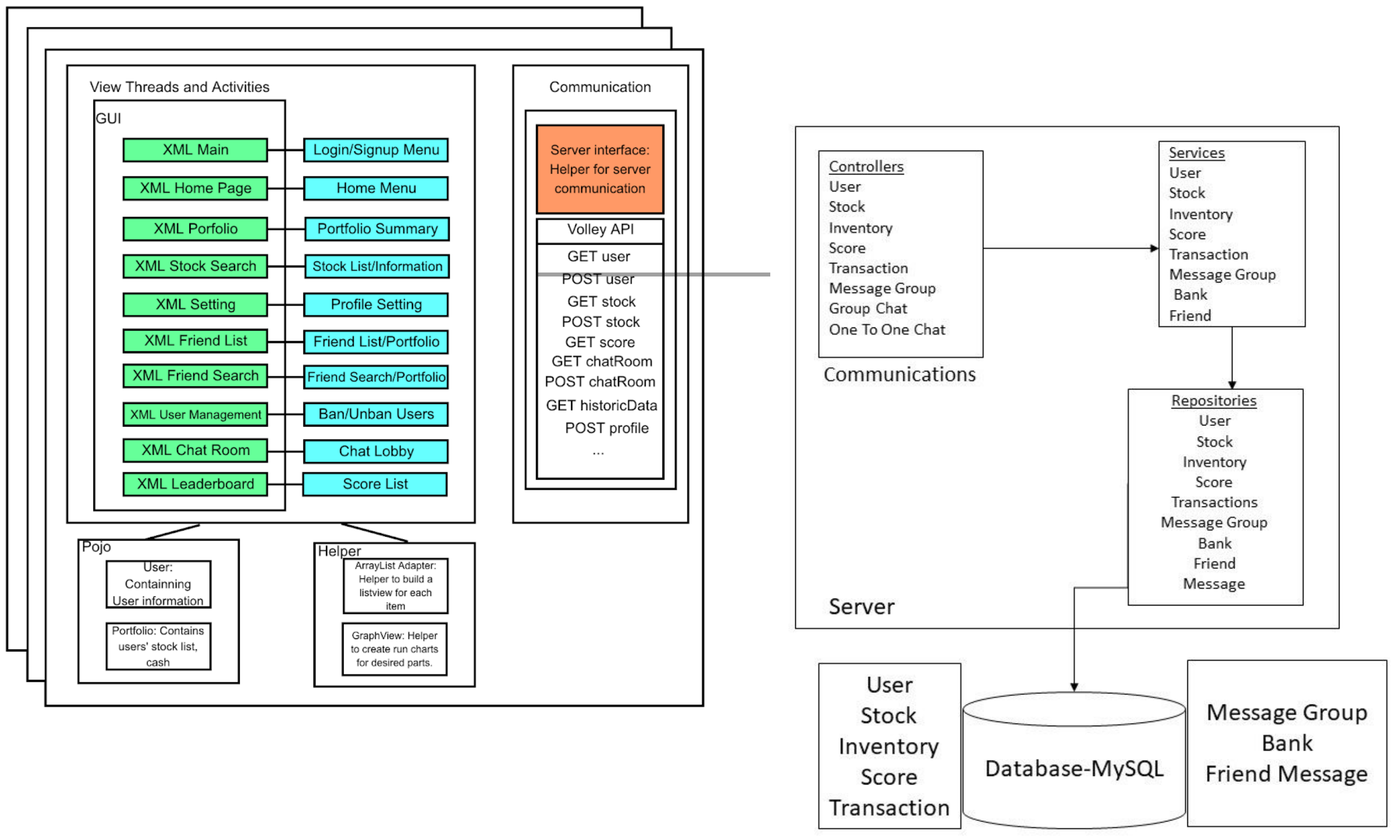
Block Diagram

JR-4

Eric Hameister
Feng Lin
Bennett Ray
Siyuan Zeng

Stock Game

Block Diagram



We have an android front end backed with java code. By connecting to our HTTP server a player may access real time stock data that is displayed on graphs to show the value changing over time. With use of sockets players may be able to chat with one another or in groups. We decided to create our user interface for a mobile device so that players may be able to make real time decisions anywhere, just as if trading on the real market. A player may track themselves over time and compete with their friends to compare strategies and make this a valuable learning experience for the user.

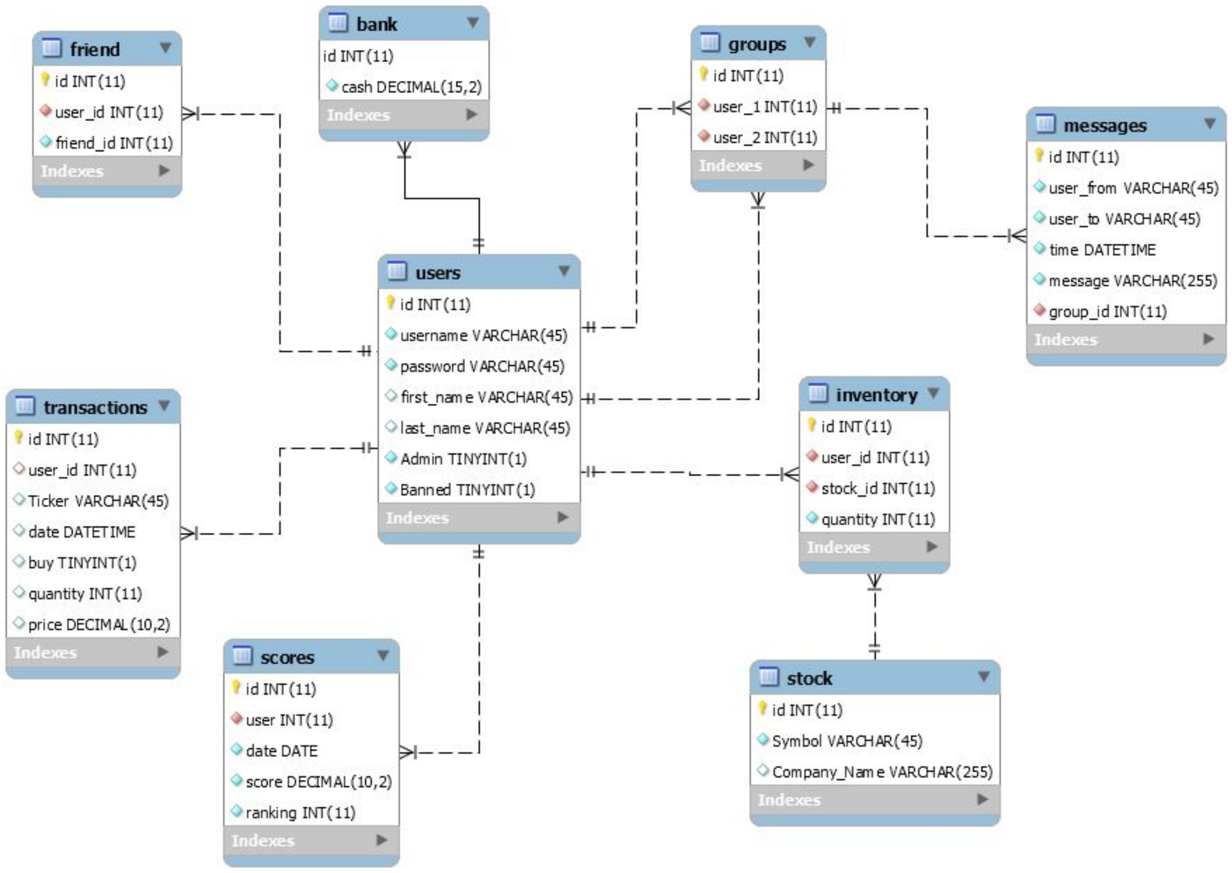
We are using Spring Boot and the Spring Framework to facilitate our back end development. With Spring we can easily create HTTP endpoints as well as create socket connections with multiple simultaneous users. We use a HttpURLConnection in order to get live data from an external API whenever the data is requested. Spring automatically handles conversion of JSON objects from the frontend to useable Java Objects and vise versa.

We have a MySQL Server in order to persist and retrieve required data about users and the stocks that they have purchased. Our Spring backend is connected to this server and handles converting and persisting of Java Objects to rows in the database.

The three user types are player, admin, and guest. We really wanted to make the game accessible and easy to get into so we wanted a guest type so that they may checkout the game, before making an account. Players have full access to the game allowing them to buy and sell stocks and track their portfolio over time. Admins have an additional screen allowing them to ban and unban users, if required.

Not storing historical stock data: We decided that it was best to put more strain on the server computationally than to try to control and optimize the database. The api that we are using allows retrieval for historical data so we decided to retrieve it from the api as required, even if redundant.
Score Calculation: We thought it would be best if the score was just the total networth of a player's portfolio. This allows the best investors to climb their way to the top gradually even if starting later than other players. So the score is the sum of all values of stocks as well as remaining cash.
Not including FINRA/transaction fees: Fees are extremely varied depending on the method of transaction. We considered adding a player by player option to add fees for their own portfolios so that they may get a more realistic picture of investing. But as a consequence would make the game unfair for those with higher fees and therefore we decided to just leave them out entirely.

Tables



friend
id INT(11)
user_id INT(11)
friend_id INT(11)
Indexes
bank
id INT(11)
cash DECIMAL(15,2)
Indexes
groups
id INT(11)
user_1 INT(11)
user_2 INT(11)
Indexes
messages
id INT(11)
user_from VARCHAR(45)
user_to VARCHAR(45)
time DATETIME
message VARCHAR(255)
group_id INT(11)
Indexes
users
id INT(11)
username VARCHAR(45)
password VARCHAR(45)
first_name VARCHAR(45)
last_name VARCHAR(45)
Admin TINYINT(1)
Banned TINYINT(1)
Indexes
transactions
id INT(11)
user_id INT(11)
Ticker VARCHAR(45)
date DATETIME
buy TINYINT(1)
quantity INT(11)
price DECIMAL(10,2)
Indexes
inventory
id INT(11)
user_id INT(11)
stock_id INT(11)
quantity INT(11)
Indexes
scores
id INT(11)
user INT(11)
date DATE
score DECIMAL(10,2)
ranking INT(11)
Indexes
stock
id INT(11)
Symbol VARCHAR(45)
Company_Name VARCHAR(255)
Indexes