CS 307

Design Document

Team 15

Anirudh Kaza, Dev Patel, Praveer Sharan, Swastik Agarwala, Vikhyat Jagini

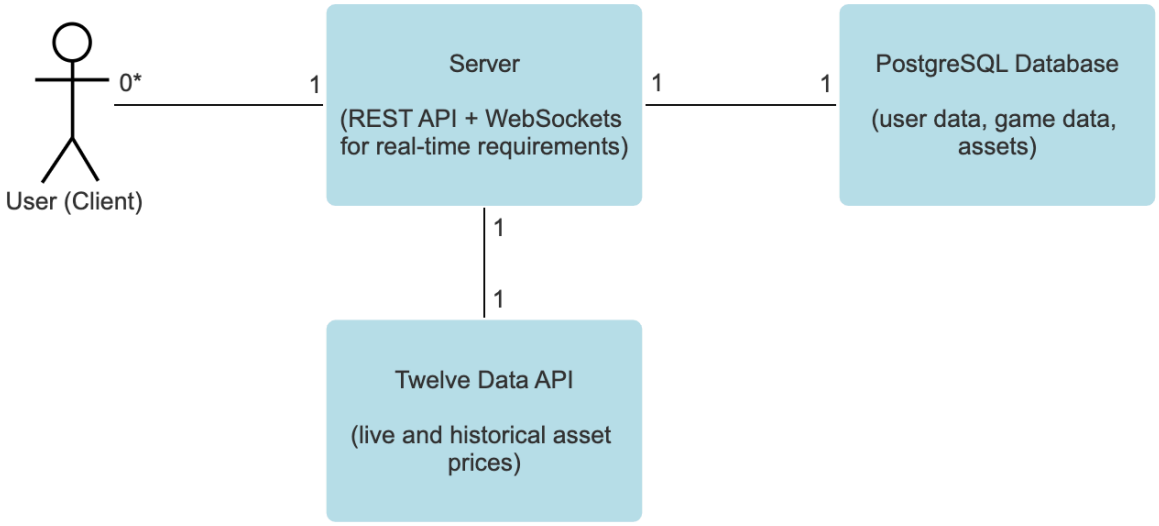
Investify

Purpose

Learning how to invest can be challenging, especially for beginners or those unfamiliar with financial markets. Since there's no one-size-fits-all approach and each person's financial situation is unique, we aim to create an application that helps users explore various investment options like stocks, mutual funds, cryptocurrencies, and bonds, while also making the learning process simpler and more accessible.

Our application will allow users to visualize the differences between investment types by seeing customized projections. Users can also assess the performance of their portfolios or participate in trading competitions to understand how the financial markets function.

Design Outline



Investify follows a client-server model. The frontend is the Investify website (built using React.js) which sends requests to the backend (built using Spring Boot). The server will communicate with the client through a REST API and WebSockets. WebSockets will be used for real-time communication, such as when a user is viewing live price changes for an asset, or when the latest financial updates for an asset need to be sent. The REST API will be used for all other communication, where the user initiates a request. This includes modifying a user profile, starting a game, viewing a portfolio, etc. The server will be connected to a PostgreSQL database. The database will contain all user, game, and asset data. When data for an asset is requested, the server will communicate with a financial API (for example, Twelve Data) for the data, parse it into the appropriate format, and send it back to the client.

Design Issues:

Functional Issues:

1) What Information Is Required For Creating An Account?

- Option 1 - Name, Email, Password
- Option 2 - Username, Email, Password
- Option 2 - Name, Username, Email, Password, Phone Number

Choice: Option 2

Justification: We decided to go with a username system, so that users can find their friends through a chosen, unique username in our system. We won't need the user's phone number or name for the purposes of our project.

2) How should we allow users to import assets for their portfolio

- Option 1 - User inputs assets manually
- Option 2 - Users upload brokerage statements that are parsed
- Option 3 - Let users links their brokerage accounts through brokerage APIs

Choice: Option 1 and 2

Justification: We decided to give users multiple options for adding assets to their portfolios. If users have existing assets from other brokerages, they can import them through statement documents. However, some users may not have any assets and just want to track the performance of some assets, in which case they can just manually input assets of their choice. Option 3 may not be feasible for our project so we will do more research and consider adding this later if time allows.

3) What information do users need to provide to create projections

- Option 1 - Just show basic compounded calculation based on asset they chose
- Option 2 - Users also choose asset type, initial amount, and options to choose best, average or worst case scenarios
- Option 3 - In addition to option 2 selections, users can also choose future market conditions and inflation rates.

Choice: Option 3

Justification: In order to cater to users of different experience levels, we decided to give users more options when they go to create projections. This will take considerably more time than option 1 or 2 but we think it will be worth the extra effort to allow users this flexibility based on their needs. Basic projections will work by including just the asset. More complicated projections will be made based on the variety of input given by the user.

4) What information do users need to provide to create their risk profiles

○ Option 1 - Answer prompt of risk level tolerance (low, medium, high)
○ Option 2 - Require users to answer a questionnaire including age, investment time period, risk tolerance, and investment goals to create their risk profile.
○ Option 2 - Answer questions about whether users would be willing to take certain trades or risks through example questions to assess their risk

Choice: Option 2

Justification: We decided to go with a general risk assessment that was still not too simple. We didn't want to ask specific questions about trading decisions since not all users may have experience trading/investing. Still, we wanted the risk assessment to be comprehensive enough so that it would be more effective in creating the user's risk profile.

Non-Functional Issues:

1) What API should we use for market data?

○ Option 1 - Polygon.io - provides detailed stock data, no bond data
○ Option 2 - Twelve Data - provides the basic data for all asset types we need
○ Option 3 - Alpha Vantage - provides detailed bond and stock data, no other data
○ Option 4 - Yahoo Finance - provides detailed data for most asset types we need, but is very unreliable due to third party services needed

Choice: Option 2

Justification:
We decided to use Twelve Data as the data service API for getting our market data. This seemed to be the best option for us since the free plan was very generous with 800 API credits per day (max of 8 per minute). The free version includes data about all the assets we needed, some of which were premium features in the other APIs. The issue with this API is that we can't request multiple assets at once so it will require using a lot more than 8 API calls per minute in reality but we can deal with this issue easily (upgrading our plan) compared to handling multiple different API services to get the information we need.

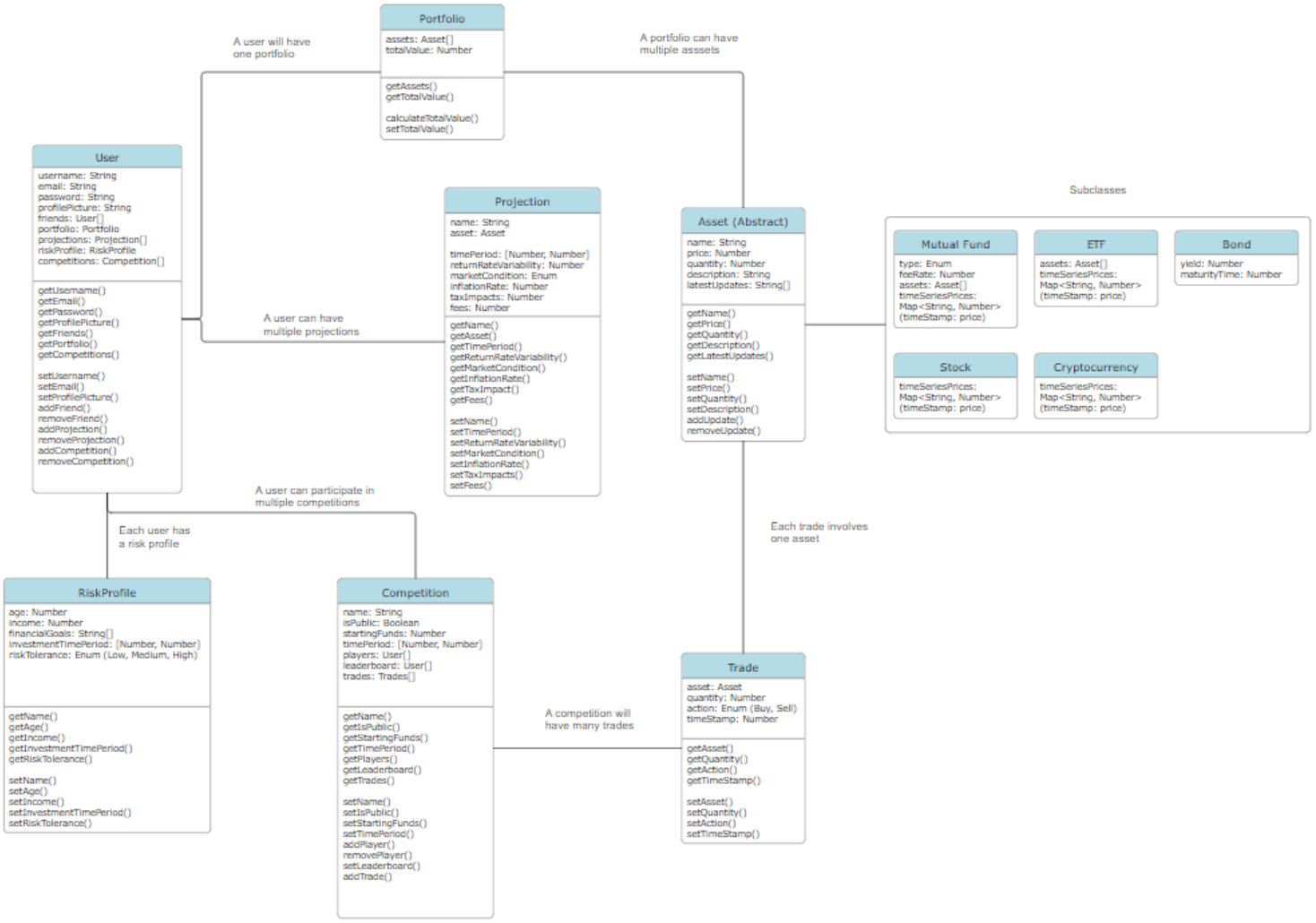
2) How should the server communicate with the client?

○ Option 1 - REST APIs only - simple and easy to implement
○ Option 2 - WebSockets only - more difficult but provides live information
○ Option 3 - Combination of WebSockets and REST APIs

Choice: Option 3

Justification:
We chose to implement most of our communication using REST APIs. This is the simplest way for us to handle all the asset data we will be getting. We plan on adding a web socket for live news feeds that the server will update for the client side to view.

A drawback will be that our implementation will be more time consuming and difficult to implement as we will have to balance handling both types of requests



Design Details

Class level design of the system

Description of Client Classes and their Interactions

The classes are designed based on the objects and tables in our application.

1. **User**
    a. Account is created when users sign up with a username, email and password
    b. Users can update their profile picture through their account settings
    c. Account details for the user can be modified through their account settings
    d. The user can complete their risk profile by taking a risk assessment that will ask questions about age, income, financial goals, investment time period and risk tolerance
    e. The user will be able to search for assets from asset types of stocks, ETFs, crypto, mutual funds, and bonds on the asset page
    f. Users can navigate to the projections page to create an investment projection by selecting an asset

g. Navigating to the portfolio page, users can manually input assets to track in their portfolio or upload brokerage statements to import assets.
h. All users have the ability to participate in ongoing competitions or create their own competitions to play with friends by navigating to the trading page.
i. User can add or remove users from their friends list and invite them to competitions

2. Asset

a. This is an important class that will be used to represent all the information about the asset type, used throughout most classes in our project.
b. Asset types will be created for each asset we support
    i. stocks/ETFs
    ii. Crypto currencies
    iii. Mutual funds
    iv. Bonds
c. When a user searches for an asset to view, it will be stored as this type
d. The data for these assets will be fetched using API calls to TwelveData
e. User portfolio will consist of many different assets
f. User projections will be made using a specific asset
g. Trading competitions will consist of selling or buying specific quantities of these assets

3. Portfolio

a. Stores all the assets that a user added to their investment portfolio
b. Has a total portfolio value which is may be updated by fetching live prices of all assets
c. Can be used by the projection tool to create a portfolio projection
d. Interacts with the risk profile to create the risk-to-return plot for the portfolio
e. Interacts with the assets to create the portfolio heatmaps and sector pie charts
f. Also used by Competition class to assign each user with virtual funds they can spend for buying assets

4. Projection

a. Using the asset selected by the user, creates a projection (in the form of future time series data) based on the asset information as well user parameters
b. User parameters include:
    i. Asset information
    ii. Time period
    iii. return rate variability
    iv. Market Condition
    v. Inflation rate
    vi. Tax impacts
    vii. Fees
c. Can interact with portfolio class to make multi-asset projections based on portfolio assets

5. Trade

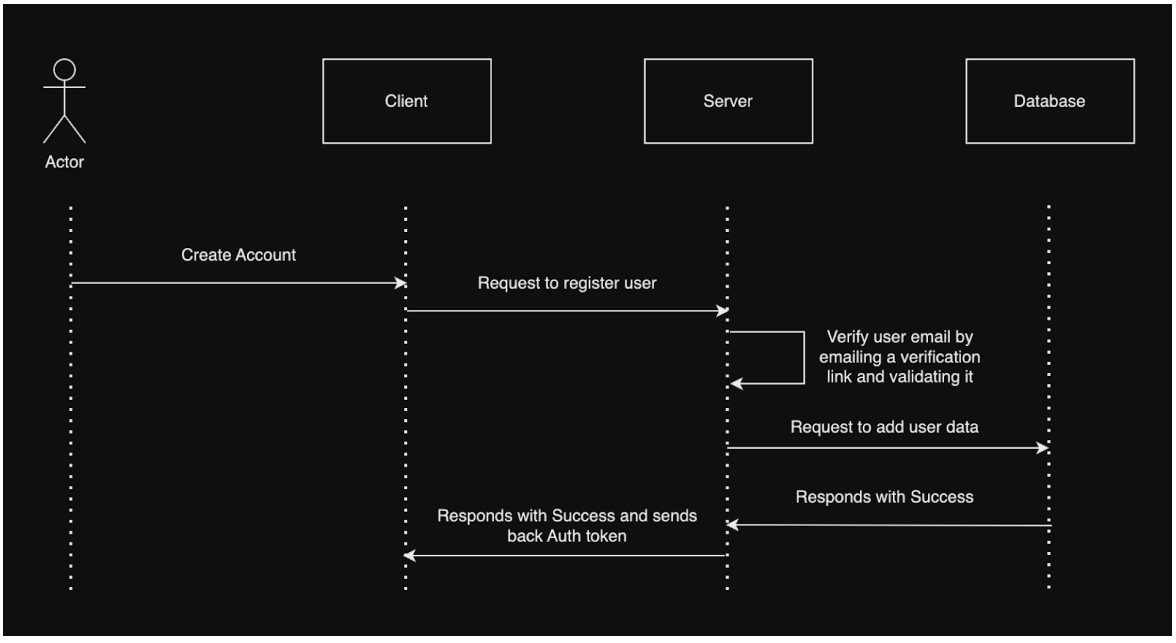
a. Used by the competition for the making trades
b. Has an action of buy or sell

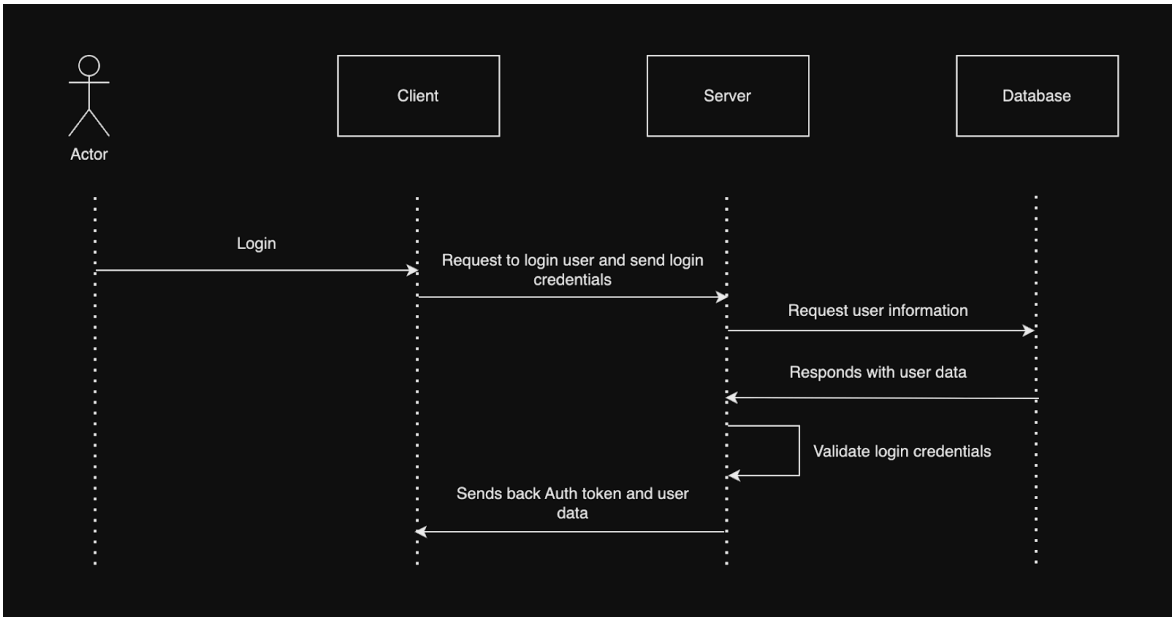
c. Quantity of asset
a. Timestamp of trade action

6. **Competition**
    a. Represents a trading competition and contains all necessary information to run it
    b. Has values for competition duration and initial starting funds amount
    c. Maintains the list of all users participating
    d. Each user is given a portfolio relevant only to the current competition and virtual funds to spend based on competition rules
    e. Maintains a ranked leaderboard of portfolio values
    f. Maintains trades for each user to update their portfolios and total value
7. **RiskProfile**
    a. Stored within the user class
    b. Created using the answers to questions asked about:
        i. Age
        ii. Income
        iii. financial goals
        iv. Investment time period
        v. risk tolerance
    c. Has a risk preference of (high, medium, low) based on risk assessment
        i. Used by portfolio for risk analysis charts
    d. Risk profile will also be used to suggest users investment options

Sequence Diagrams

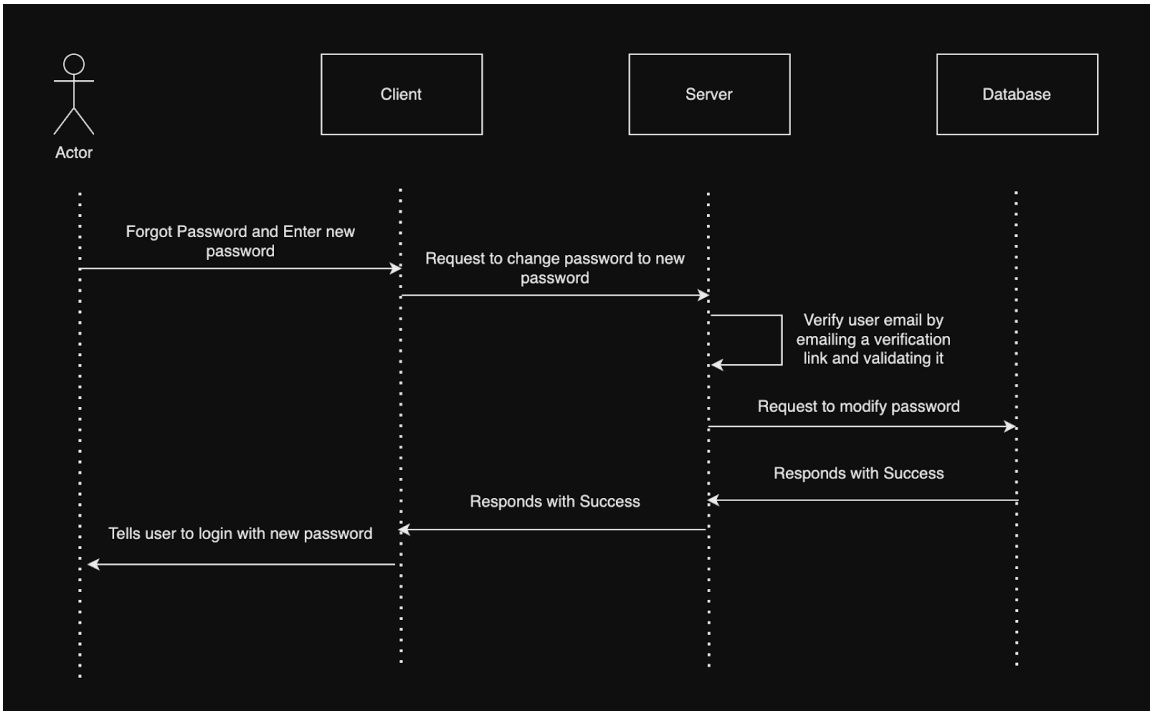
Create / Login / Auth account





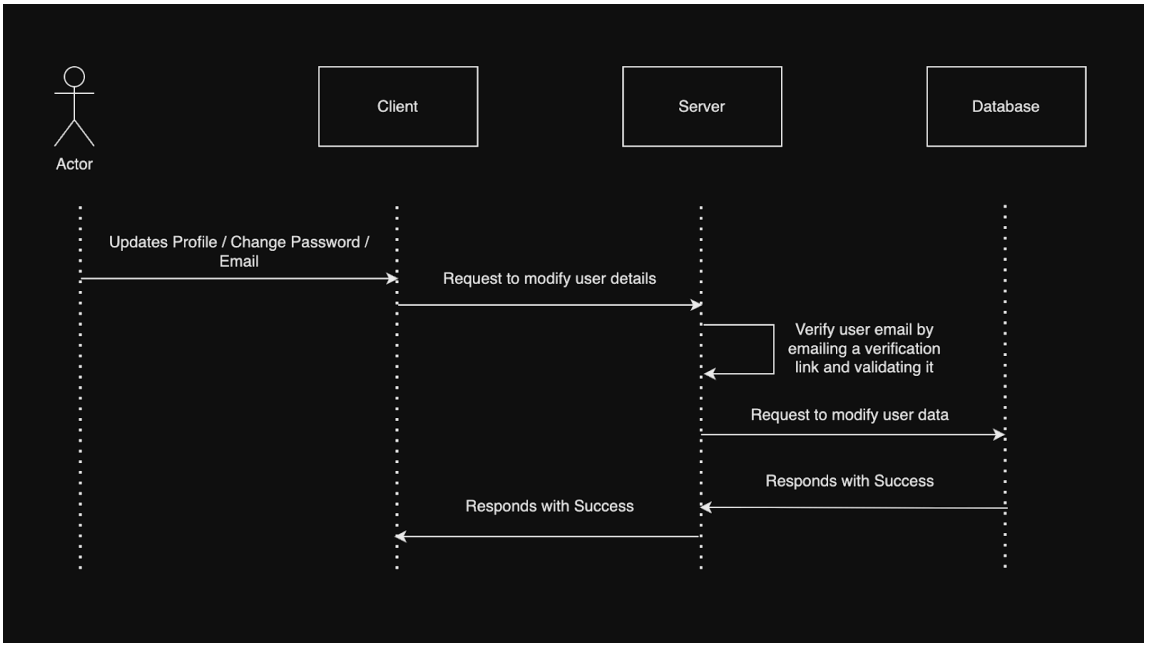
When a user visits the Investify website, they will have the option to either login or create an account. If the user selects the “Create Account” option, they will be linked to a page requiring them to enter a username, email, and password. When they hit submit, the user will be sent to the server which will validate the input fields (valid email address, not a duplicate username, not a duplicate email address, and password is secure). A verification email will be sent to the email address by the server, and the user will have to verify it. If there are no issues, the information will be saved to the database as a new user, the server will generate a session authentication token, and the user will receive a response (which contains the session token) with an HTTP success code. If a user instead selects “Login”, the user will be taken to a page that will require them to enter in their account’s email address and password. After hitting submit, the input fields will be sent to the server and validated (checks if the user exists in the database). If the input is valid and the correct credentials are provided, a session authentication token will be attached to a response with an HTTP success code and sent to the user.

Forgot user password



If the user forgets their password, they can click on the “Forgot Password” link on the login page. Then, they can enter the email address associated with their account, and hit submit. The server will then verify that the user’s account exists, and will send a link to the user’s email address. The link leads to a page where the user can enter a new password and hit submit to the server. The user’s password will then be updated in the database.

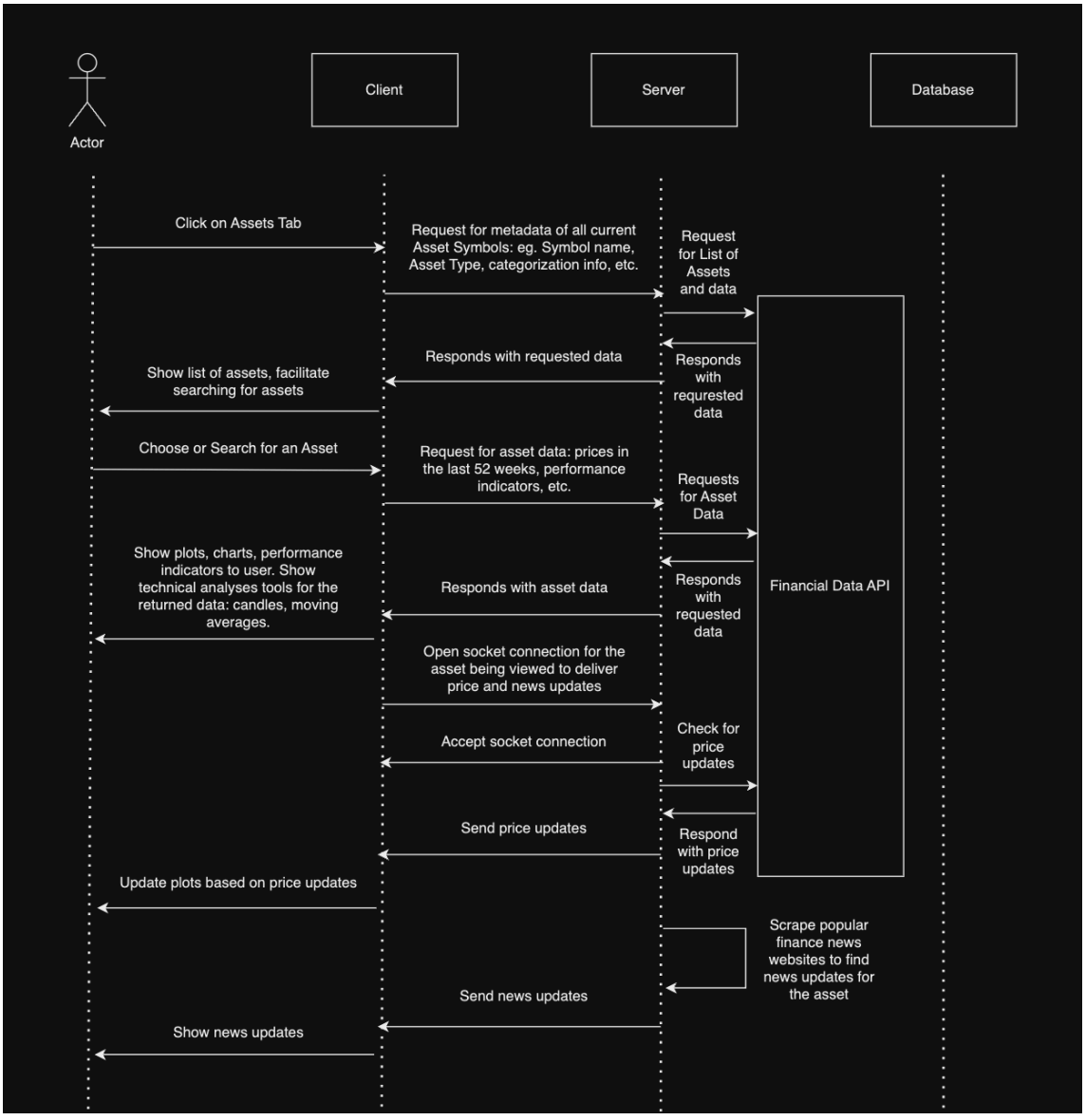
Edit user profile



Once a user is logged in, they can modify profile details such as their username, email, and password. The user can click on their profile icon in the navbar which opens a dropdown menu including a link to the edit profile page. On that page, the user can modify any of the previously mentioned fields and hit submit. The server

will validate the inputs by checking the following: the username is not a duplicate in the database, the provided email is not already tied to an existing account and is valid, and the password is secure. If those conditions are met, the changes are processed and a success message is sent to the user. If the user edits their email address, a verification email will be sent to the new address by the server, and the user will have to verify it.

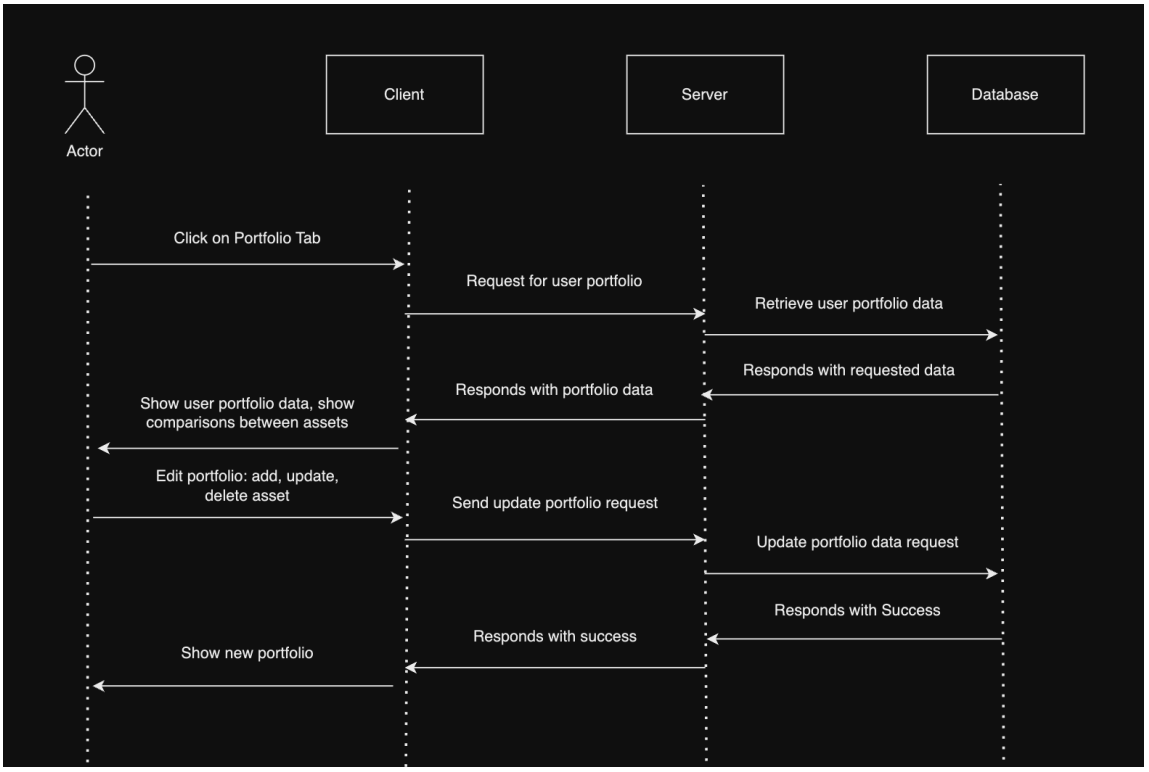
Query asset(s) for asset page, technical analysis



The website's navbar will have a button called "Assets" which will send the user to a page where they can search for various assets (types include, stocks, mutual funds, ETFs, bonds). There will be a search bar where the user can enter a specific asset symbol and its type. Once they hit submit, the input will be sent to the server. The server will send an API request to Twelve Data with the asset symbol and type, and we will parse the response to get the current asset price and the price history from the past 52 weeks. We will also web scrape popular financial news sources for

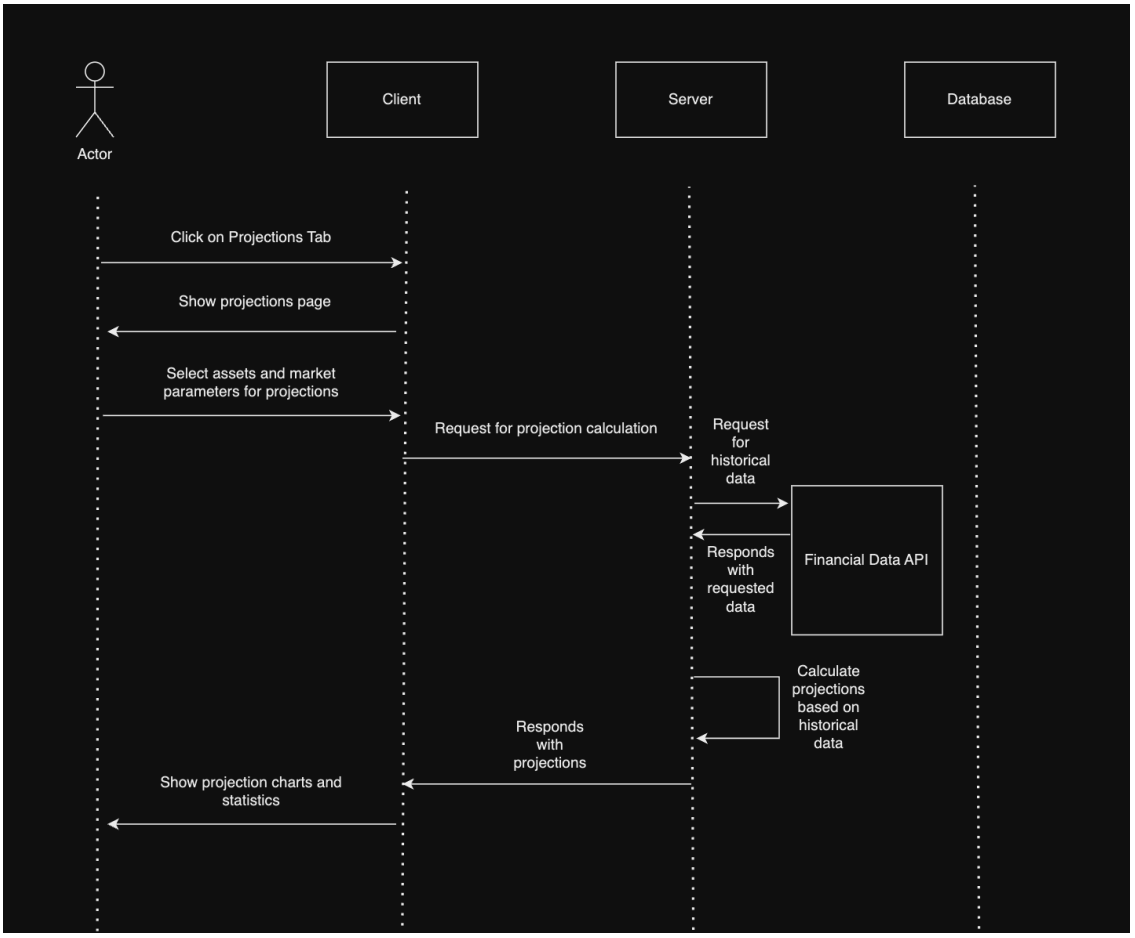
the latest updates about the asset. This data will then be sent in a response from the server to the client. On the client, candlestick charts will be displayed to represent the changes in asset prices and the current asset price will be displayed.

Add asset to portfolio



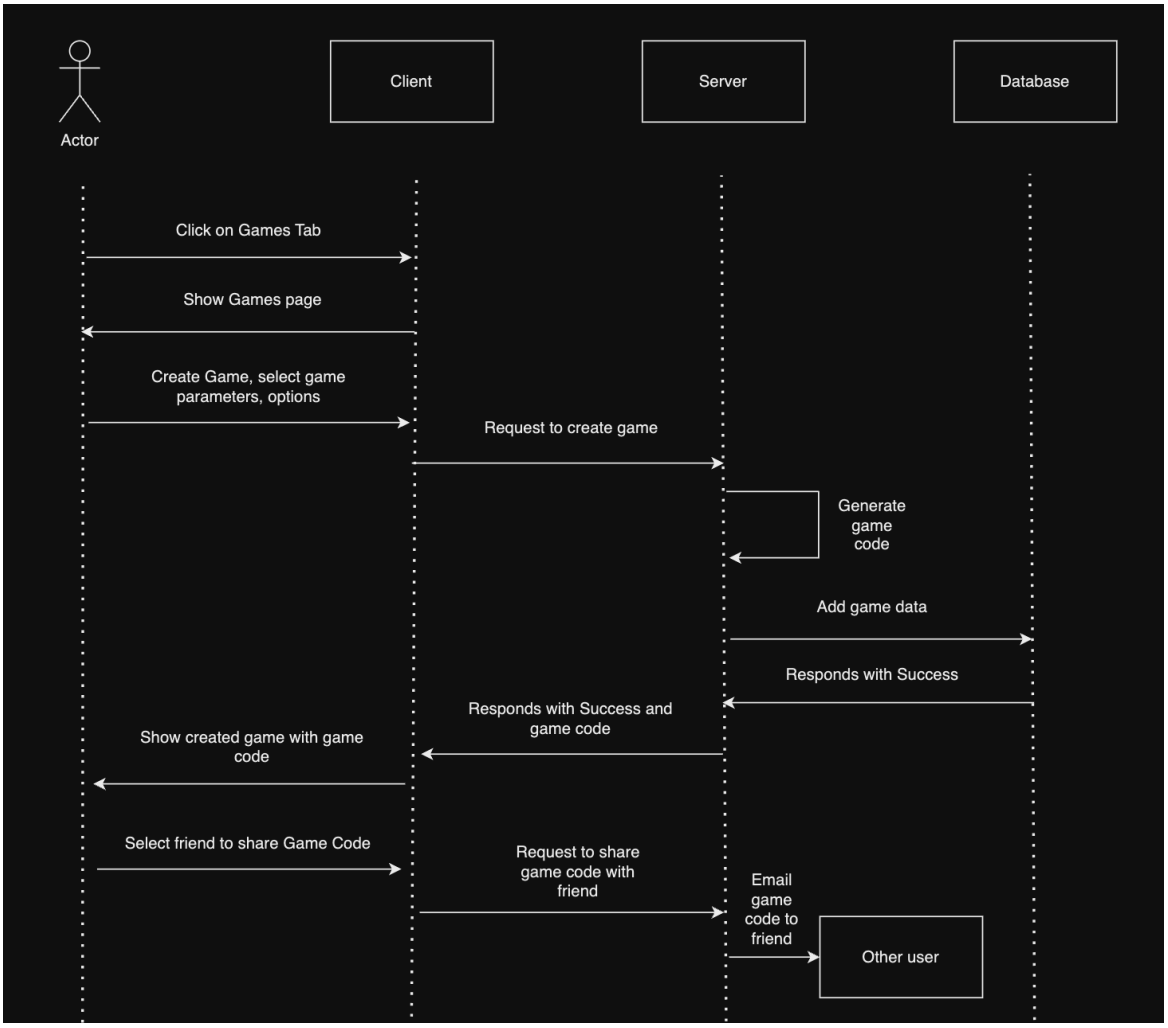
On the assets or portfolio page, the user can search for an asset (as described above) and add it to their portfolio. The adding process includes selecting the asset and specifying the number of shares of that asset to add. This data will be sent to the server, where a request will be made to the database to add the asset and quantity to the table. From the database, this information can either be added or retrieved. Within the client and server interactions, the user can create a portfolio, add/update/delete an asset. Essentially, the server is a middleman for storing persistent data for the user's portfolio.

Create projection



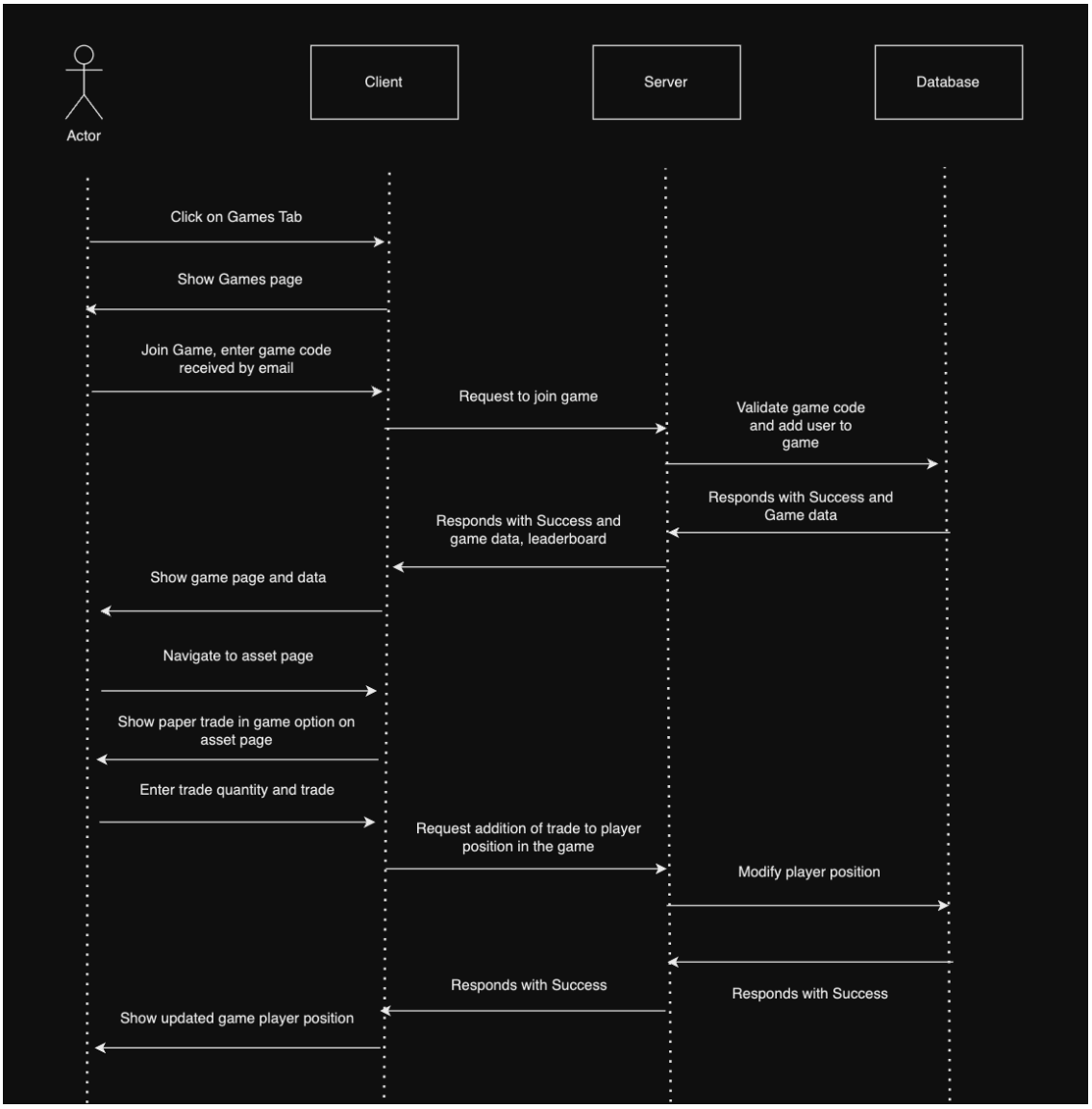
Within the user and the client-side, the user can click on the projections tab which will show frontend charts and statistics for projections. The user can also select a specific projections page as well as assets and market parameters. The client and server side communicate through a request for a projection calculation and a response. Within the server and the database, the server communicates with a financial data API with a request to fetch historical data. Then, within the server, it calculates projections and sends it to the client side for chart manipulation and for user access.

Create Game / Invite Friends



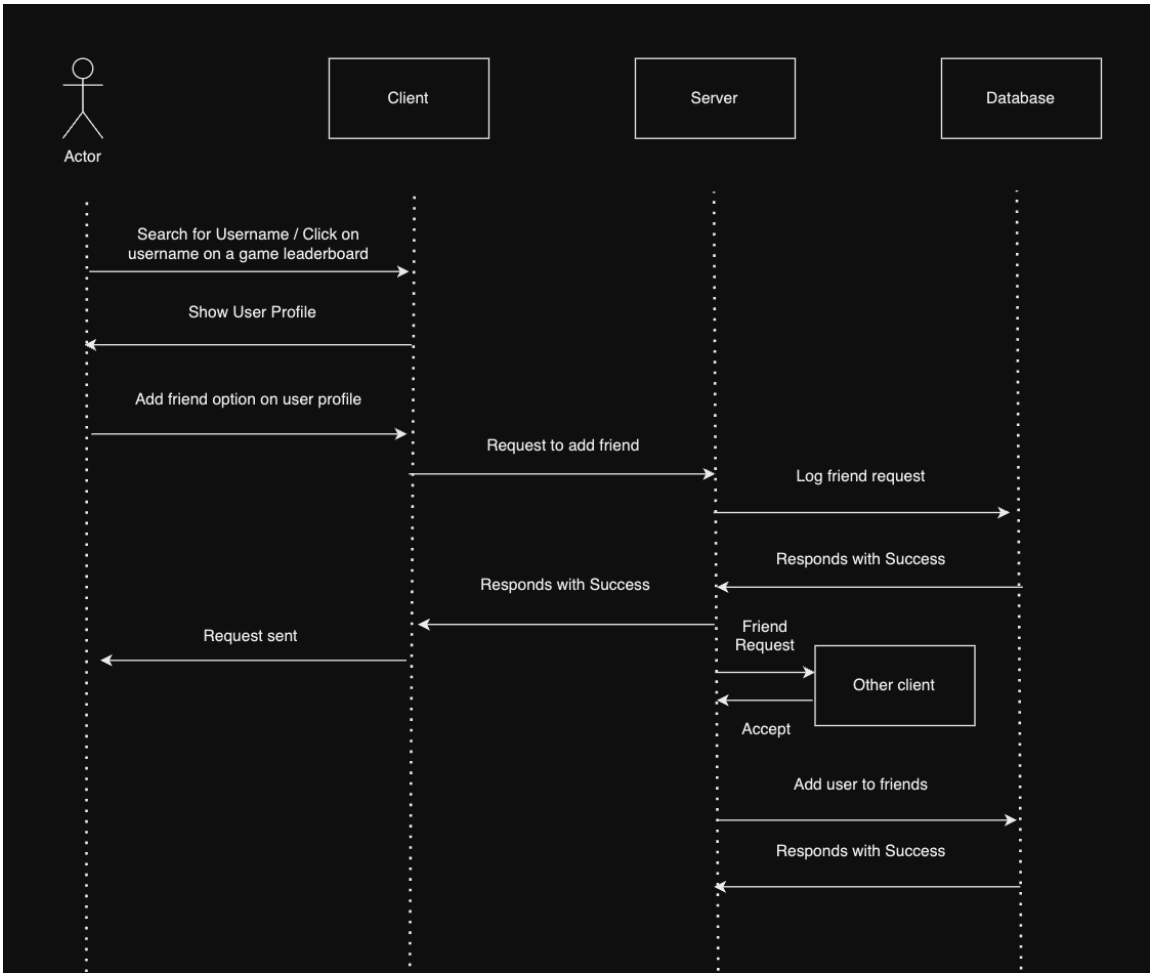
Accessible to the user from the client side, there is the “games” tab which displays a games page where the user can create a game with parameters and options. The created game with its respective code appears which the user can then use to share this code with other prospective players. Within the backend, the server can handle two requests: creating a game and sharing a game code. The server can place relevant game data in the database, such as game code, creator, etc. based on existing game codes, and verifies success. This is how the game code can be sent back to the user for sharing. Additionally, the backend sends the game code to the other user client by email.

Join / Play Game



Made available to the user from the client side is the games tab and an option to join a game from a game code which would be sent through email. This box, when filled and submitted, will send a request to the backend which will resolve a request with the same code if it exists, and then add the user to the game within the database. Upon this response, the game page and data will load. The user may also navigate to the asset page which will show paper trades in game options, where they may enter the trade and its respective quantity. This will be sent to the backend as a request which will calculate the new parameters for the player and update the player's ranking within the game if necessary, which will carry over to the frontend.

Add Friend



The user can access a textbox that will allow them to search for the username or click on the username from a game leaderboard. Upon this click, it will show the user profile. Within this profile is an option to send a friend request. The backend will carry a request to add this friend. The server will log this request in the database, send a URL with this friend request to the client based on the unique code generated by this request. When the invited user clicks on the URL, a request will be made to the backend to resolve the invite in the database according to the code and log a friendship instead.