CS 307 Design Document

Team 11:

Ryan Huff

Sam Kravitz

Akash Lankala

Raziq Raif Ramli

Tyler Stanish

Blake Steel

Index

Purpose

Fortune is a cryptocurrency trading game that is competitive, simple, and fun to play. Current cryptocurrency trading games on the market have severe limitations. Some games are very simple in nature with only the ability to trade one type of cryptocurrency at a time and have no ability to manage the coins the player has. These games quickly become boring and turn users away. Other games are more complex, but are built around events managed by the company of the game itself and have monetary prizes for winning those events. While money is a great incentive to lure in players, it also can drive away many others that would rather simply play a trading game.

Similar in concept, stock market simulation games give players a way to play with virtual money and attempt to grow their imaginary net worth against other players. However, these games are usually intricate with the different lingo of the stock market and are not friendly to newcomers to the genre.

We seek to create a game that is easy for new players to grasp while still maintaining a level of complexity to create a competitive environment for current players. The use of cryptocurrencies removes the intricacies of trading stock while still providing a vehicle for trading. Our players will also be able to create different rounds of games to play with friends as well as compete in an ongoing global game against many different players from around the world.

Functional Requirements

1. General

As a player,

a. I would like to see a landing page that includes the amount of money I currently have in the global game and the current prices of several cryptocurrencies.
b. I would like to have a play page that displays the different types of games that I could play.
c. I would like to have a dedicated game screen for every game that I join.
d. I would like to have a guest player option (optional).
e. I would like to have a notification popup bar where users can get notified of any new updates, friend requests, and game invitations (optional).

f. I would like to have a trophy system to keep track of my achievements and have some major goals to look forward to in this game (optional).
g. I would like to be given a set of goals that I can accomplish every week to motivate me to come back to the game from time to time (optional).
h. I would like to backtest trading bots to test my cryptocurrency trading models (if time permits and obtaining real-time cryptocurrency data is not feasible).

As a developer,

a. I would like to have an admin page to manage players and enforce game policies.
b. I would like to create a service/daemon to fetch live cryptocurrency data.
c. I would like to parse the raw data from (potentially) several APIs.
d. I would like to integrate historical cryptocurrency prices into the platform.
e. I would like to serve live cryptocurrency prices (and potentially other cryptocurrency statistics) to the user with websockets.
f. I would like to calculate a player's net worth, so that a player can see an accurate representation of their standing compared to others.

2. Player account

As a player,

a. I would like to be able to register for a Fortune account.
b. I would like to have a popup page that displays my profile's details.
c. I would like to be able to upload a profile picture (optional).
d. I would like to be able to log in and log out from my account.
e. I would like to be able to change my username (optional).
f. I would like to be able to reset my password (optional).
g. I would like to be able to add friends, so that I can compete with them and compare our progress (optional).
h. I would like to be able to view and manage my friends' list (optional).
i. I would like to link my account to Facebook, so that I can find my friends who also play this game easily (optional).
j. I would like to be able to register with my Google account so that the registration process could be done quicker (optional).

3. Creating a game

As a player,

a. I would like to create a private group game through the play page.
b. I would like a choice of game title.
c. I would like a choice of duration of the game.

d. I would like a choice of which cryptocurrencies (BTC, ETH, etc) are to be traded during the game.
e. I would like a choice of initial cash amount each player has.
f. I would like a shareable link to give to other players to invite them.
g. I would like a 4-digit code to give to other players to invite them.
h. I would like to invite my friends directly by entering their usernames (optional).

4. Joining / navigating to a game

As a player,

a. I would like to be able to join the global indefinite game, so that I can play and compete with strangers.
b. I would like to be able to join the global timed game.
c. I would like to be able to join a private game that has been created.
d. I would like to be able to navigate to any of my currently active games.

5. Playing a game

As a player

a. I would like to see the title of the game.
b. I would like to see a button that reveals the 4 digit code to join the game.
c. I would like to see my current cash within the current game.
d. I would like to see my current net worth.
e. I would like to see the time remaining in a game.
f. I would like to see a time series graph displaying the historical price of a cryptocurrency from the exchange(s) of my choice.
g. I would like to be able to switch the cryptocurrency that is being displayed in the graph.
h. I would like to modify the time span of crypto data to display (min/hr/day/month/yr), so that I may modify the data view to fit my buying and selling needs.
i. I would like to be able to buy mock cryptocurrency and have it attached to my account.
j. I would like to obtain more money if my net balance goes to zero (go into debt)
k. I would like to be able to liquefy all current assets immediately.
l. I would like to be able to buy/sell currency and view data about said currency (Price, history, % change, amount, min/max price on various exchanges)

m. I would like to have an option to buy coins from an exchange that offers the lowest price and sell coins to an exchange that offers the highest price at any given time.
n. I would like to see the current leader of the game.
o. I would like to be able to navigate to the leaderboard for the current game.
p. I would like to filter the leaderboard by friends (optional).
q. I would like to see players' usernames and current net values on the leaderboards.
r. I would like to receive in-app notifications for large movements in crypto price.
s. I would like to receive in-app notifications if there are any significant movements in the leaderboard.
t. I would like to see a widget that displays notification history throughout the game (optional).
u. I would like to chat with other players who are in the same game session with me (optional).
v. I would like to report players who make any form of communication abuse (optional).

Non-Functional Requirements

1. Architecture

a. As a developer, I would like to employ a client-server architecture for Fortune.
b. As a developer, I would like to develop the frontend with React and the server in Python with Flask.
c. As a developer, I would like to store user and cryptocurrency data with a hosted PostgreSQL database.
d. As a developer, I would like to deploy Fortune through Amazon Web Services in order to integrate our frontend and backend easily.

2. Design

a. As a developer, I would like to make an intuitive, straightforward, and interactive interface.
b. As a developer, I would like the players to be able to navigate through the application and execute trades seamlessly.

c. As a developer, I would like to use a pleasing color scheme that will add to the aesthetic of the application. The color palette that I plan to use can be viewed here https://coolors.co/4aa7d6-2a628f-13293d-79b473-db504a.

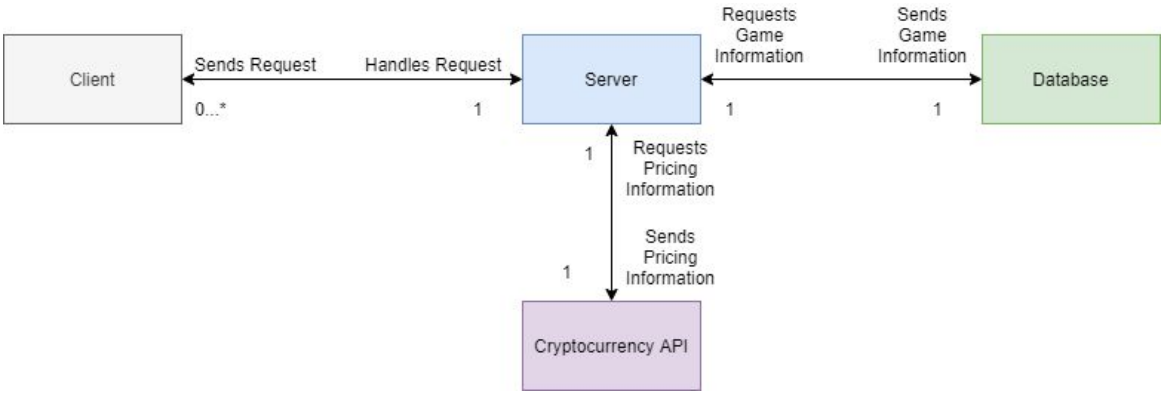
3. Security
    a. As a developer, I would like to secure the sensitive data that is stored in the database.
    b. As a developer, I would like to only allow authenticated users to access some routes in Fortune.

4. Performance
    a. As a developer, I would like the application to be responsive to all requests made by the user.
    b. As a developer, I would like the application to be able to properly handle any raised errors.

Design Outline

High Level Overview

Our project will use the client-server model due to the nature of online web applications/games. A number of clients will connect to the server, requesting data such as game or user profile information. The server then will send a request to the database and the database will respond with that information if possible. Meanwhile, a separate worker component in the server continuously works to keep pricing information of cryptocurrencies accurate by request information from an API and updates the database with that information.

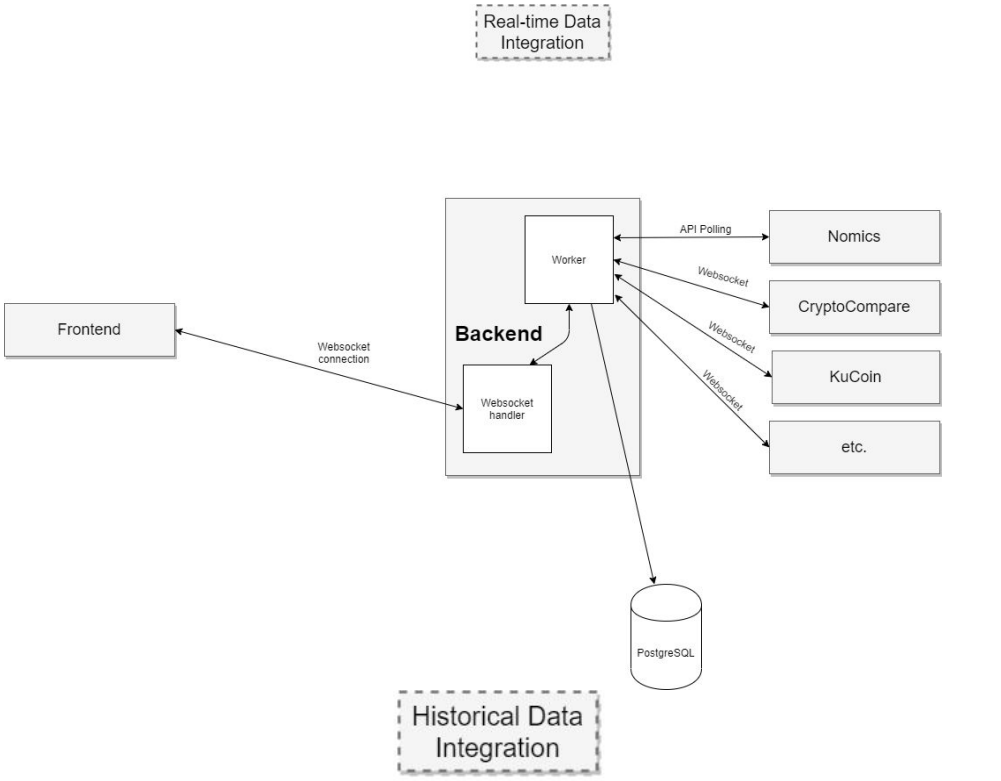


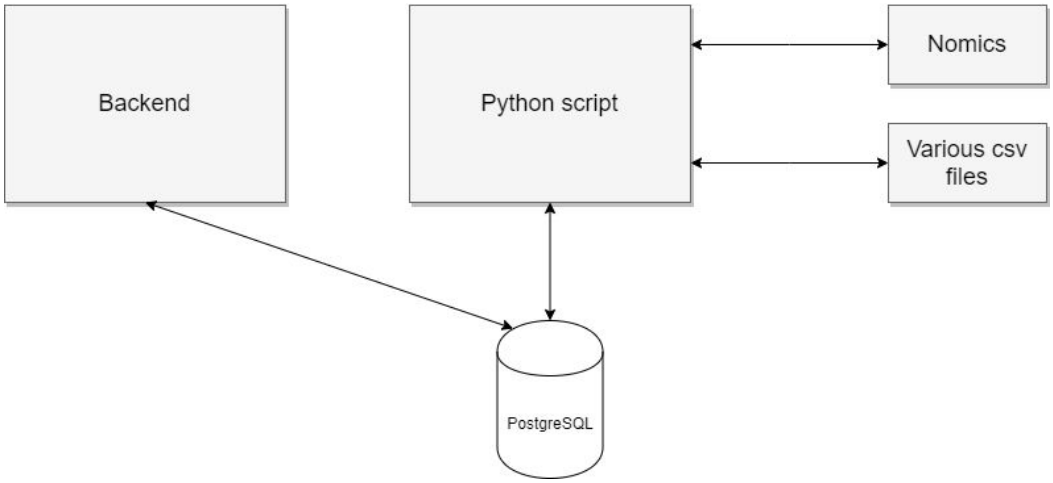
- Client
    - Frontend of the project
    - Serves as the interface for the game
    - Requests information from the server
    - Receives information back from the server and modifies the game screen
- Server
    - Backend of the project
    - Does most of the processing work for the client and API
    - Updates database with cryptocurrency information on a regular basis
- Database
    - Stores user and game information received by the server
    - Sends requested information to the server to handle
- Cryptocurrency API
    - Third party API that provides information such as pricing for various coins and historical data

- Polled by the server at a regular interval to keep prices up to date.

Real-time and Historical Data Integration

Our project will feature real-time data of cryptocurrency pricing information to be used during games. This will be done by polling an API for cryptocurrency information and using that real-time information in games. We will also use historical data provided by the API and in various CSV files to aid in presenting data on-screen.





Design Issues

Functional Issues

1. What data do we need to collect for user registrations?
    - Username
    - Password
    - Email
    - Phone number
    - Social media (Facebook / Google)

Choice: Username and password.
Justification: We decided that a username and a password is enough to create an account in our application. However, an email would be necessary for a user to retrieve their password if they forget it. We plan to add a password retrieval feature if time allows as well as the ability to login through social media sites such as Facebook or with accounts such as Google

2. How should we allow users to share a game session that they created?
    - Shareable links
    - 4-digit codes
    - Direct invitations (using other players' usernames)

Choice: All of them
Justification: Initially we were having trouble deciding on which option to choose for sharing a game due to limitations of each one concerning privacy and ease of use. A shareable link would not necessarily be private if it were given out publicly and direct invitation would require a mo mo n additional invitation page that was not in our initial mockup. Our application could potentially run out of 4-digit codes if there were enough games and would require keeping track of current codes being used to prevent any clashes. In the end, we decided to implement each one and leave it up to the users to keep links private, add functionality for 4-digit codes, and add an invitation page feature in our design.

Non-Functional Issues

1. What web service should we use?
    - Amazon Web Service
    - DigitalOcean
    - Heroku

Choice: Amazon Web Service
Justification: We decided on Amazon Web Services because of prior experience we have as well as its free tier having all the options we need for hosting a VPS and database. Using only one service for both our application hosting and database hosting allows for ease of integration between the two.

2. What backend language/framework should we use?
- Flask (Python)
- .NET (C#)
- Node.js (Javascript)

Choice: Flask
Justification: Our group was pretty mixed about which languages we have experience with. Some members had Javascript experience but no Python experience, while others had Python experience but no C# experience. We ended up going with Python because of its well documented and easy to use web frameworks (Flask), as well as having many modules that will allow implementation with some of the more technical aspects of our project (such as implementing a web socket).

3. What frontend language/framework should we use?
- React
- Angular
- Vue

Choice: React
Justification: React is well known among all group members, so it was an easy choice. React is easy to use and powerful enough to make complex yet visually appealing user interfaces. React is also compatible with any backend stack which increases our freedom to choose a backend

4. What database should we use?
- MySQL
- MongoDB
- PostgreSQL

Choice: PostgreSQL
Justification: We chose PostgreSQL because it is a SQL database (as opposed to MongoDB) and free and open source. It also has support for transactional DDL (MySQL does not) which allows for database migrations with guarantees on data and schema integrity if a migration fails halfway through.

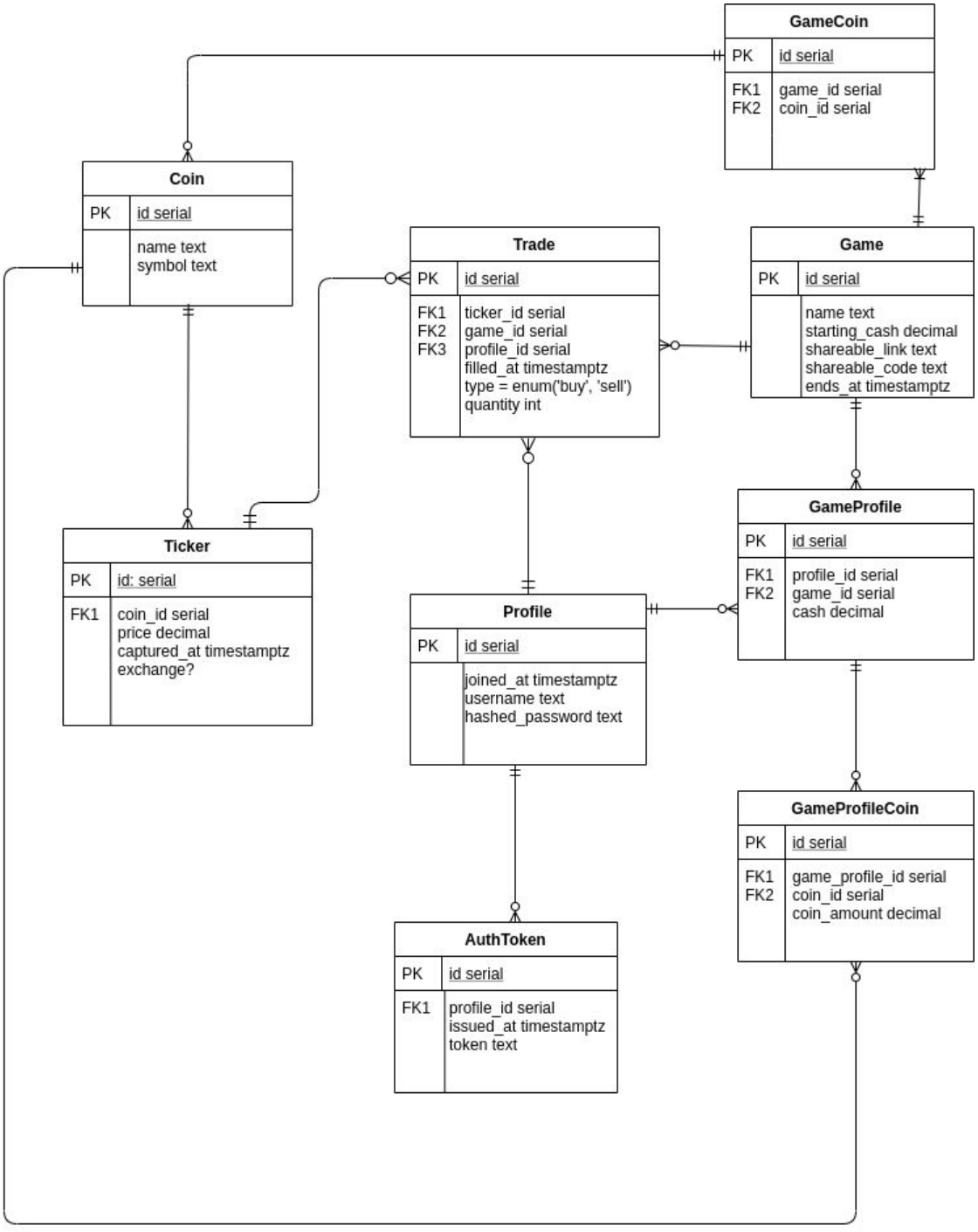
5. What API should we use to get cryptocurrency data?
    - BitcoinWisdom
    - Nomics
    - Bittrex
    - CoinCap
    - CoinCodex
    - CryptoCompare
    - Cryptonator
    - KuCoin
    - CryptoDataDownload

Choice: Nomics, CoinCap, and CryptoDataDownload
Justification: Nomics provides real-time cryptocurrency prices and is cached for only 10 seconds; CoinCap provides a simple, free, and easy to use websocket api. CryptoDataDownload provides hourly (but for some exchanges even minutely) historical prices in csv format which is important in our historical data integration step.

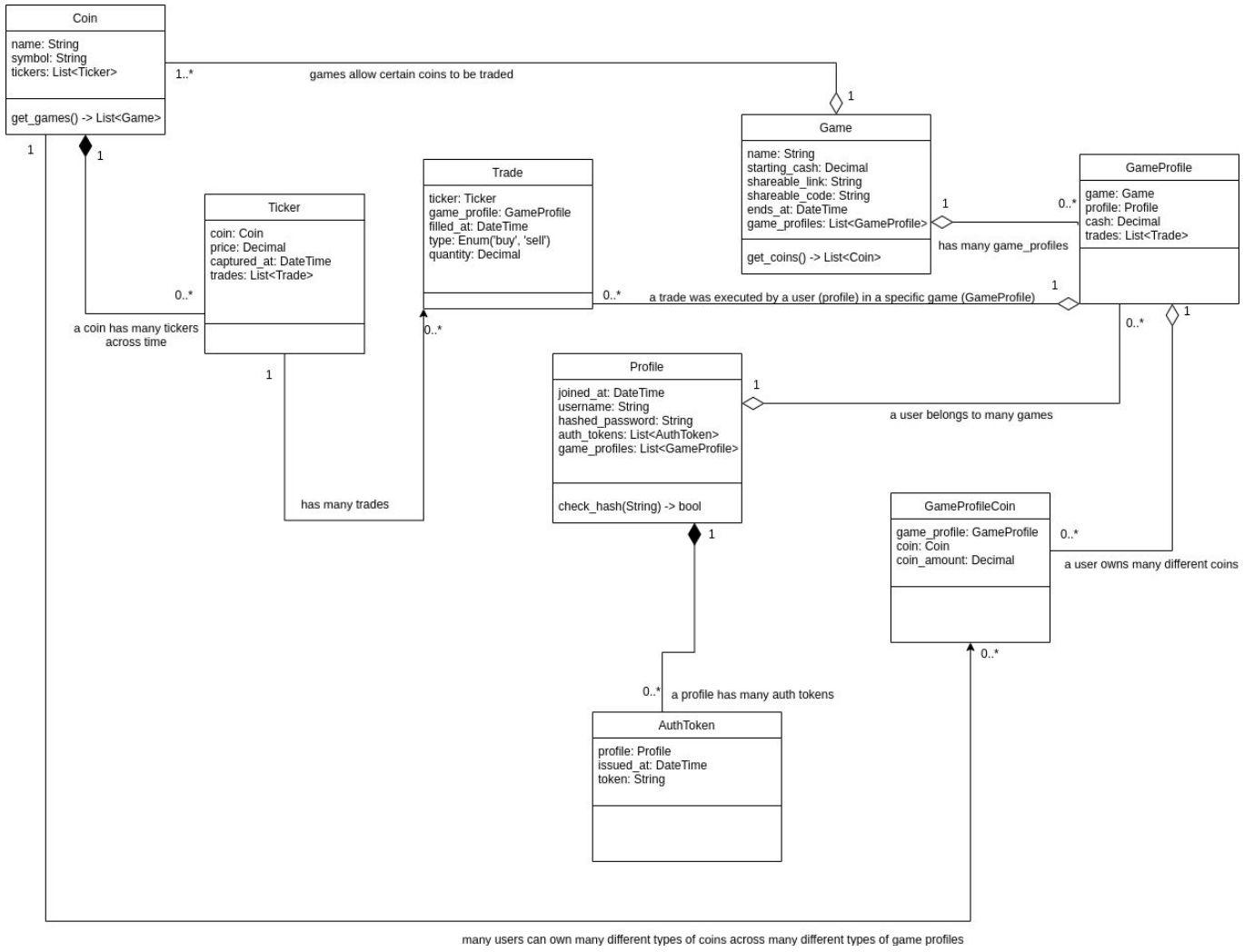
Design Details

Class and Database Design

Database Design:



Class Design:



Coin

- Represents a tradeable coin.
- A coin has a unique id.
- A coin can be uniquely identified by its symbol (e.g. BTC or ETH).

Ticker

- Data structure representing a coin's price at a specific point in time.
- A ticker will have a unique id.
- Can optionally be associated with an exchange (although exchange-specific data may not be viable).
- This data type also will be used frequently in our data integration step both historical and real-time to show users in the future to provide better looking and more refined graphs of historical data.

Profile

- A user's profile.
- A profile will have its own unique id.
- Stores user data such as hashed password, account creation date and time, and the user's username.
- Contains a list of associated AuthTokens and GameProfiles.

AuthToken

- An authenticatable token the user can use to access API endpoints.
- An auth token will have its own unique id.
- An auth token will have a globally unique token field with which the user can authenticate for future API calls from the client.
- Generated upon every sign in/sign up request.
- The token is non-expiring and there can exist multiple AuthTokens per user Profile.
- Deleted upon a user's request to log out.

Trade

- Represents a transaction to buy/sell a coin at a ticker price.
- A trade will have its own unique id.
- A trade will have an execution time.
- A trade will have an associated user who executed the trade.
- A trade must also belong to a specific game.
- The type of trade will also be included in this data structure (e.g. buy or sell)

Game

- A representation for a game created by a user.
- A game will have its own unique id
- Along with its unique id, a game will have unique shareable code and link
- A game will have an ending time and date after which no more trades will be allowed to be executed.

GameProfile

- Used to represent a many-to-many relationship between a game and a profile.

- Each GameProfile will have a unique id.
- A game can have multiple profiles and a profile can be participating in many games.
- A GameProfile will store the balance for a profile (user), their associated trades, and the coins that they own (via GameCoin)

GameCoin

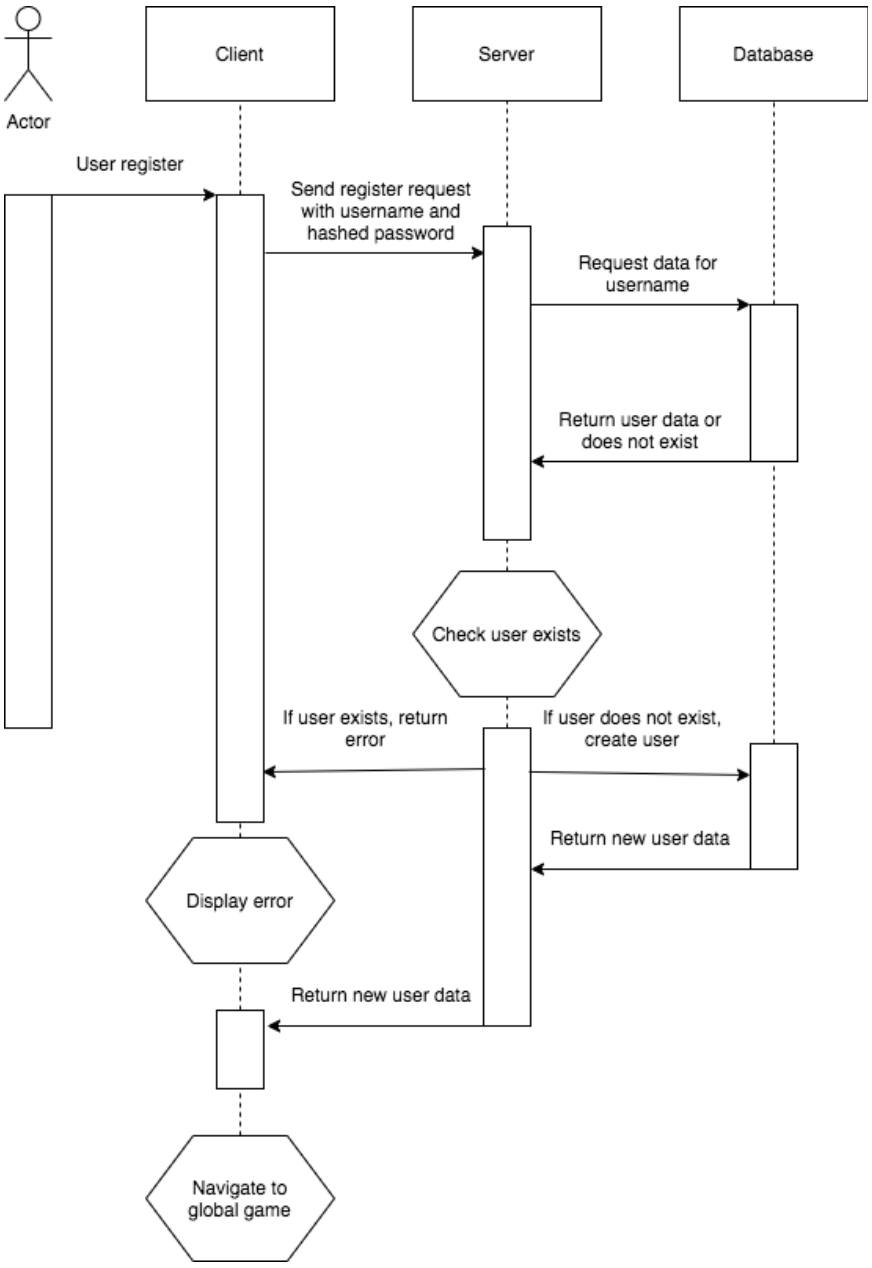
- *This is an internal class and is not included in the public-facing API and therefore not the UML class diagram, however it is in the database schema.*
- This must be an “internal class” because the idea of a GameCoin really doesn’t exist, rather it is used to associate Games and Coins that are allowed to be traded in a specific Game. That is, to represent a many-to-many relationship between a coin and a game.
- Each GameCoin will have its own unique id
- The reason behind this is that a game can optionally allow certain tokens to be traded and disallow others to not be traded.
- By allowing a coin to be traded it will have a GameCoin which has the id of the coin that is allowed to be traded and the game_id which points to the game that a GameCoin is being traded under.
- At the class level, an attempt to fetch the games a coin belongs to (or the coins a game has) will go through this mapping class.

GameProfileCoin

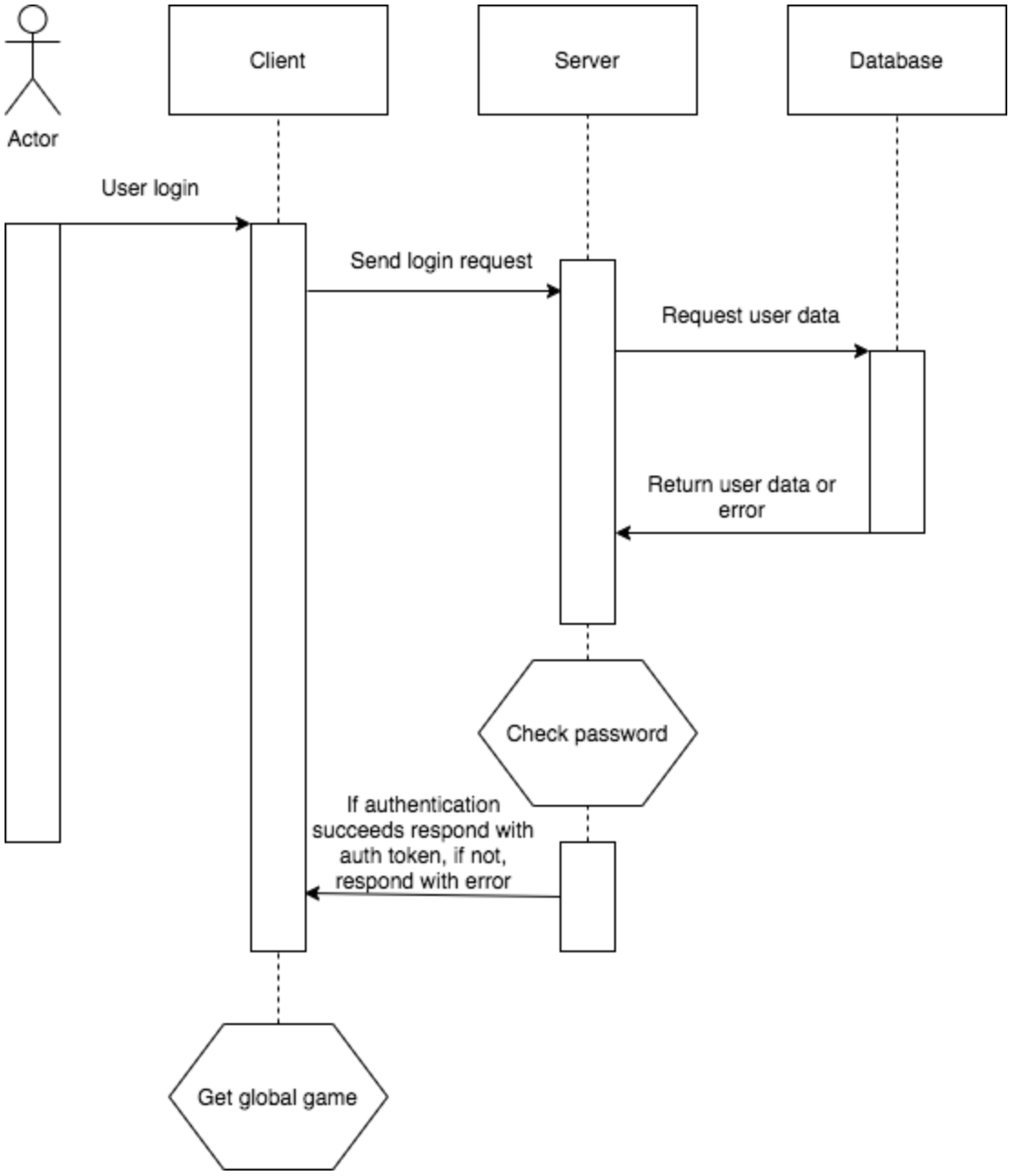
- This represents an ownership or holding of a cryptocurrency asset by a player for a specific game.
- A GameProfileCoin will have its own unique id
- This is the owner of a many-to-many relationship between a GameProfile and a Coin.
- The reason for this relationship is because a GameProfile can own multiple types of coins.
- For example, a user can have holdings of BTC and ETH with various amounts.
    - Therefore in this scenario a user’s GameProfile will have multiple GameProfileCoins.

Sequence Diagrams

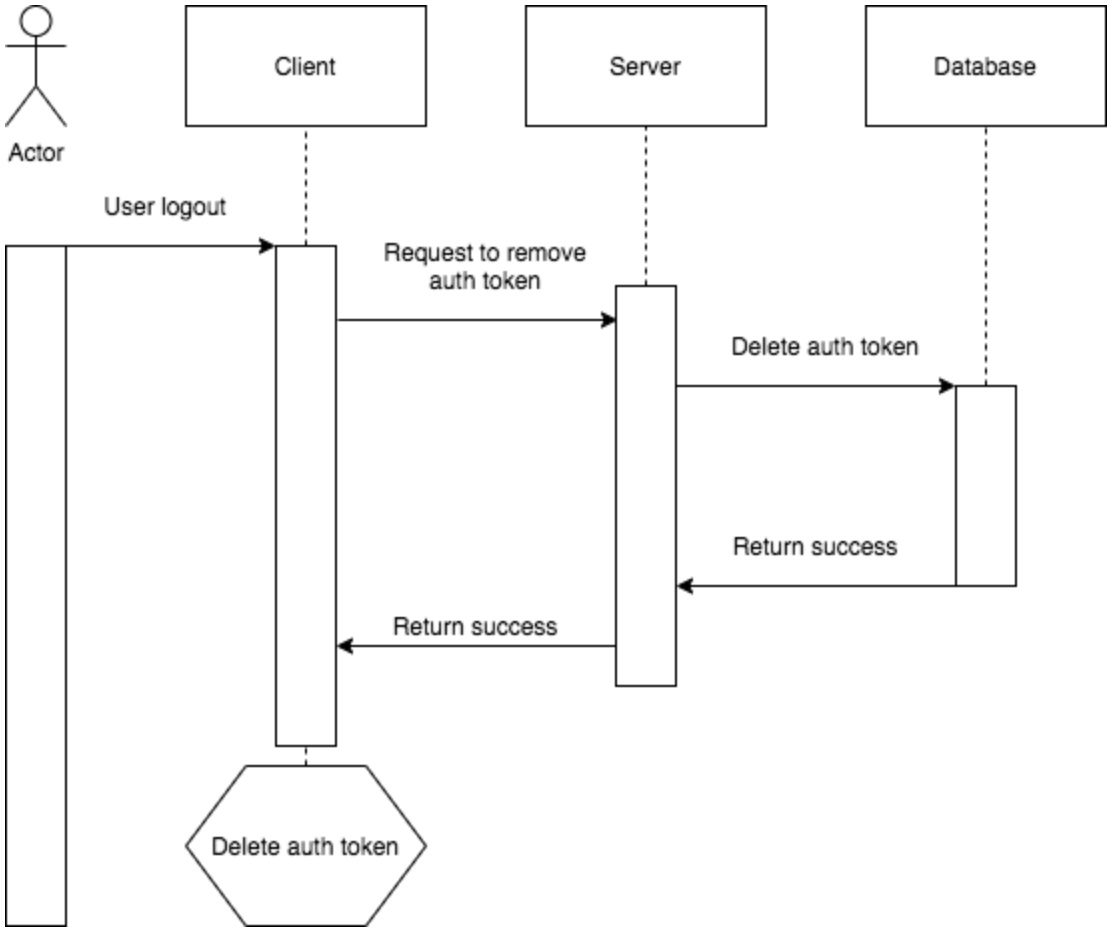
1. Register



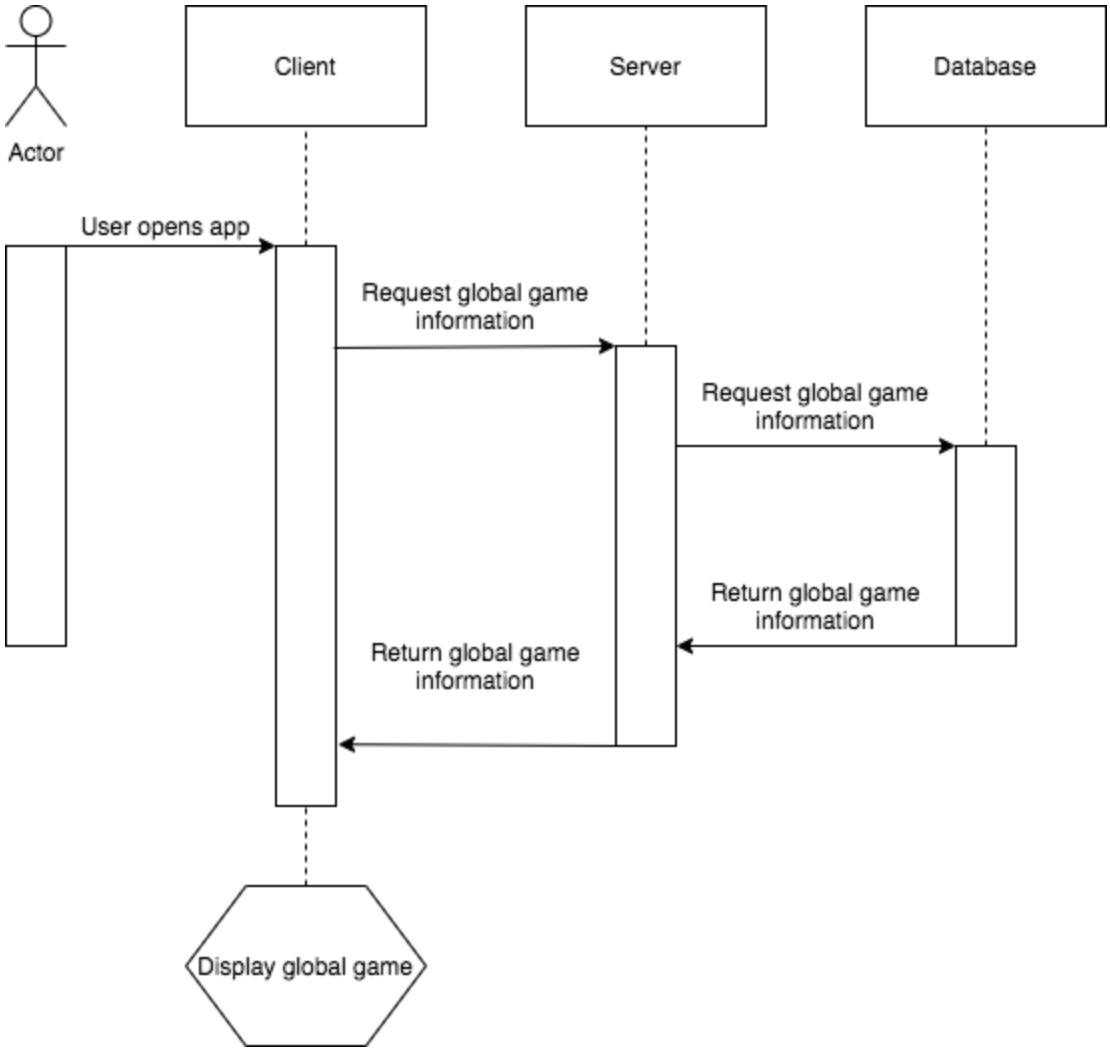
2. Login


Actor
Client
Server
Database
User login
Send login request
Request user data
Return user data or error
Check password
If authentication succeeds respond with auth token, if not, respond with error
Get global game

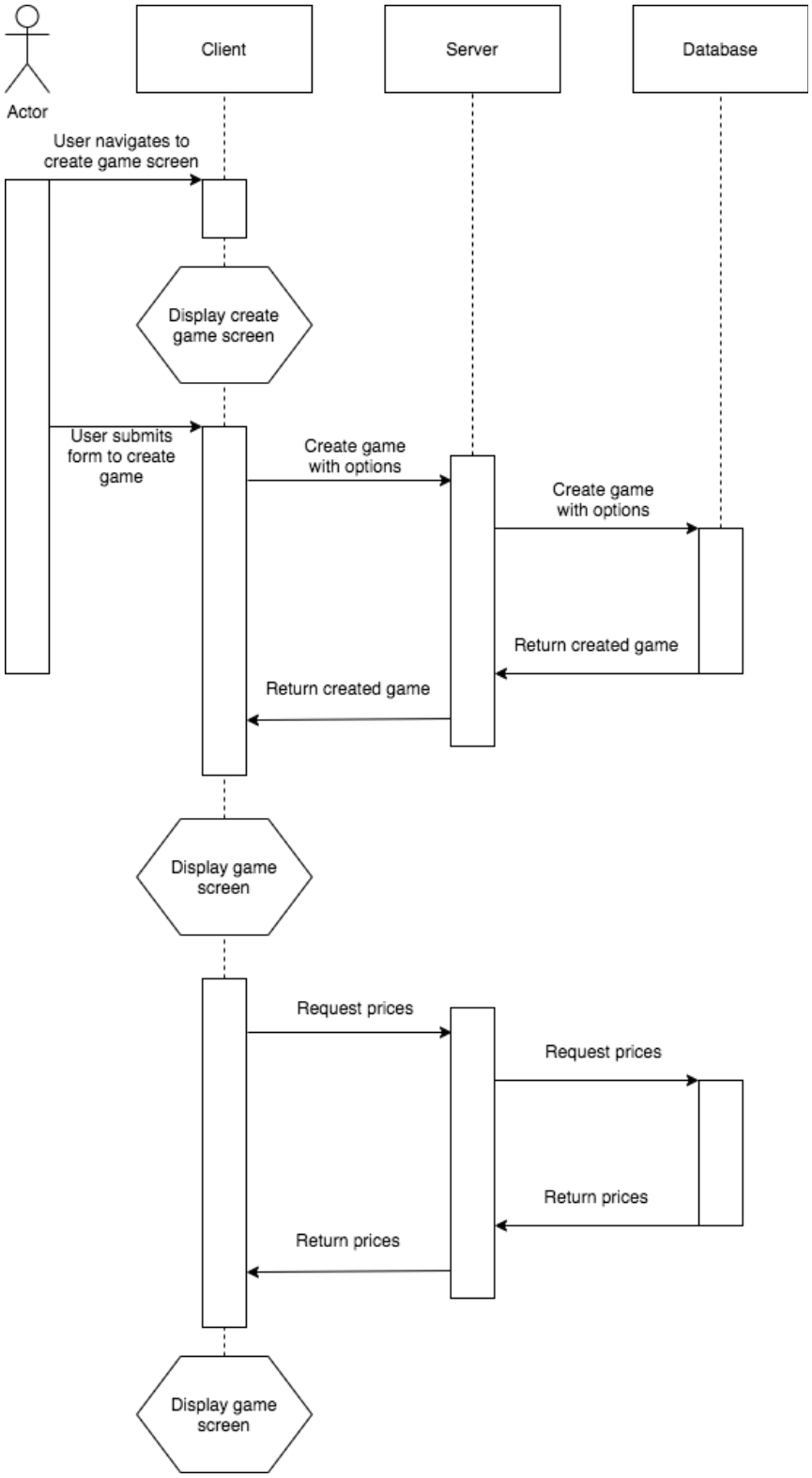
3. Logout


Actor
Client
Server
Database
User logout
Request to remove auth token
Delete auth token
Return success
Return success
Delete auth token

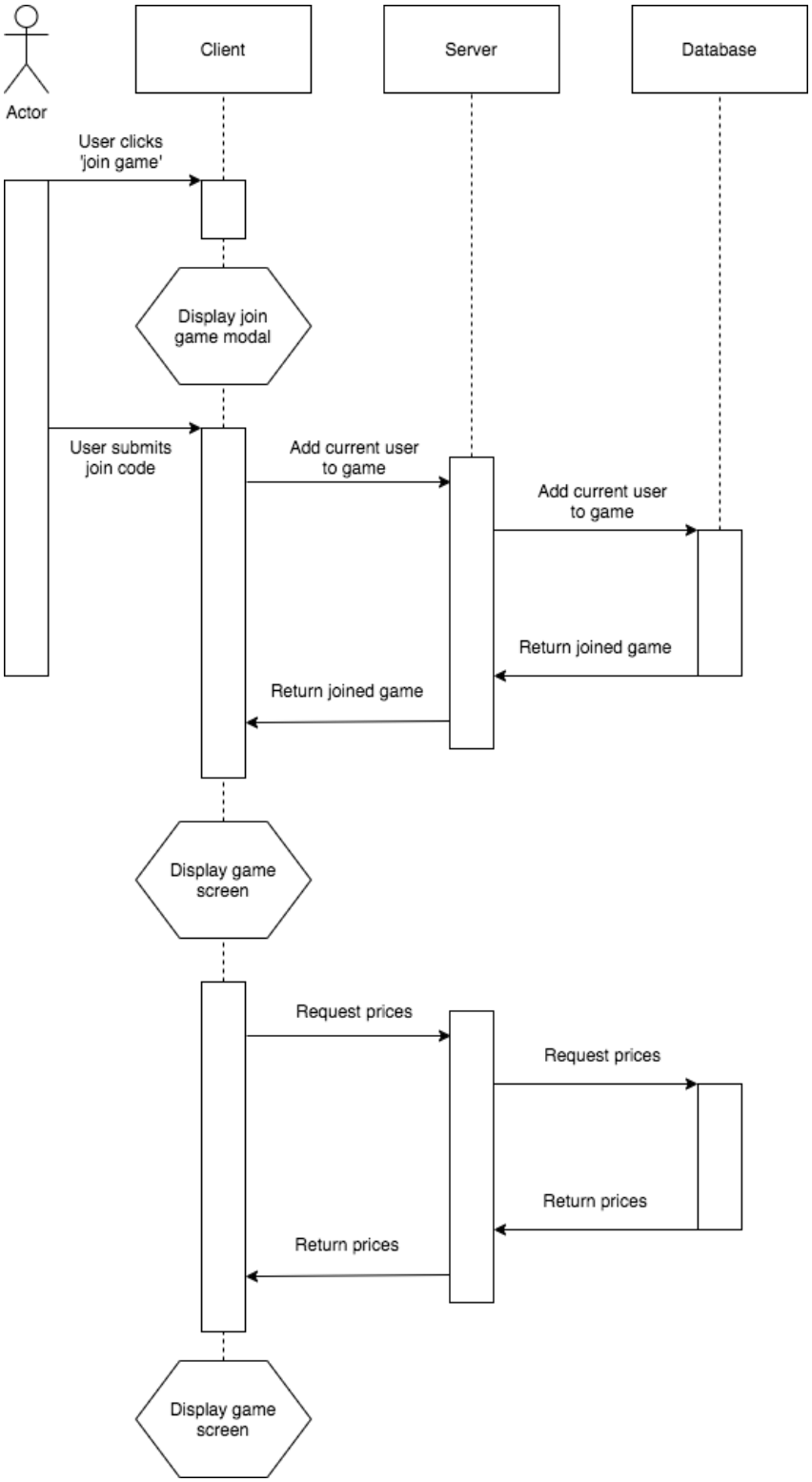
4. Open App


Actor
Client
Server
Database
User opens app
Request global game information
Request global game information
Return global game information
Return global game information
Display global game

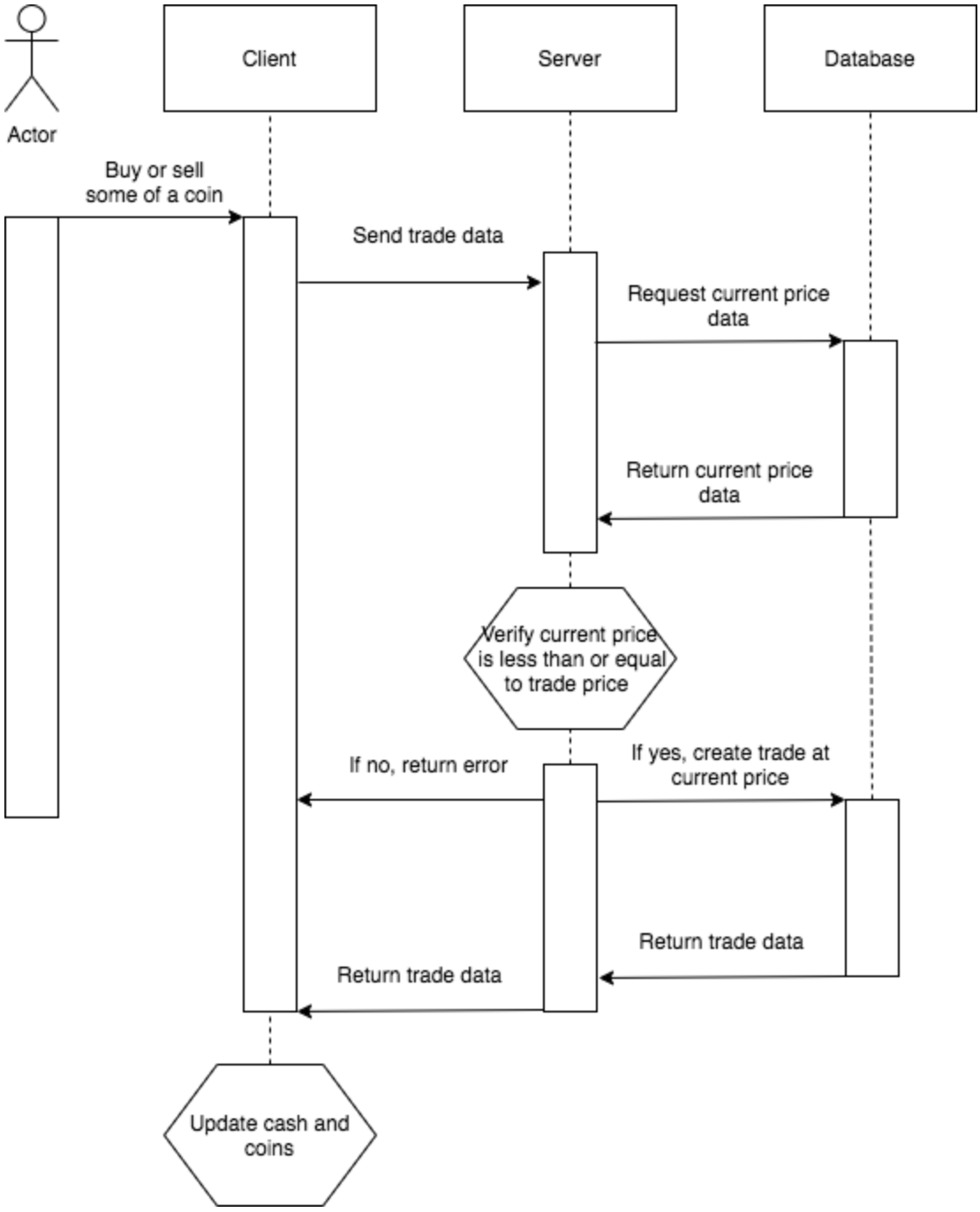
5. Create Game



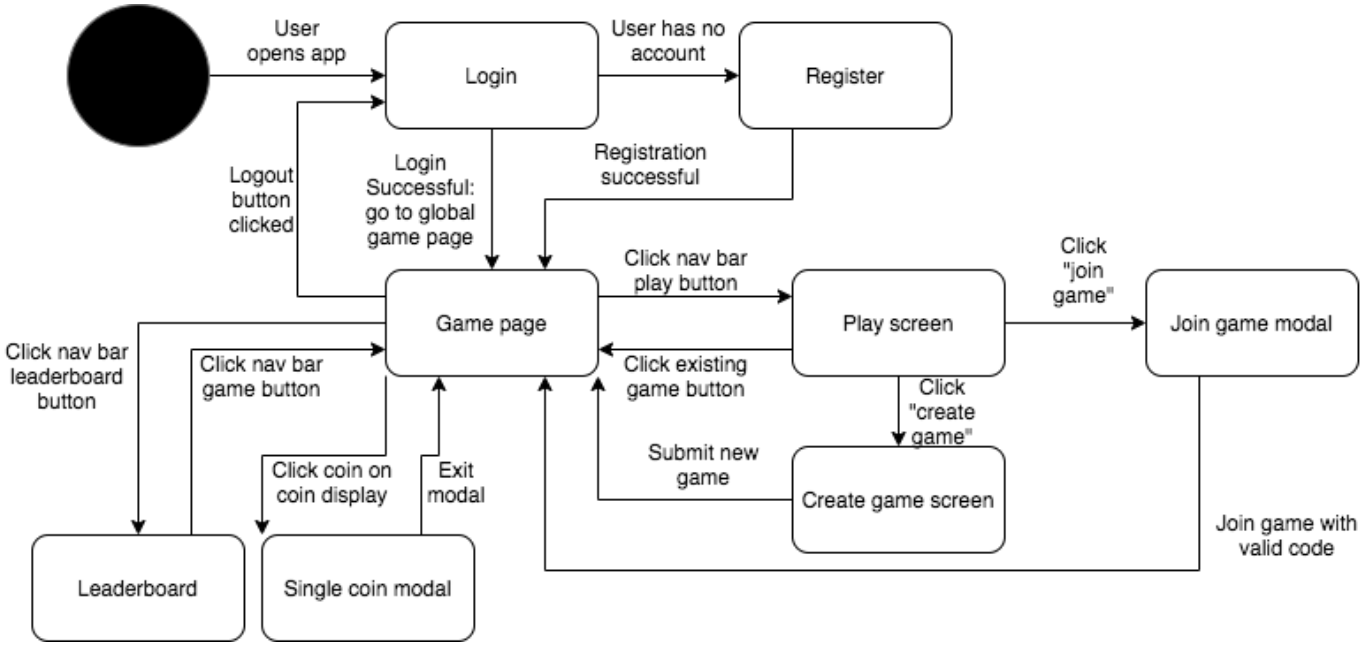
6. Join Game


Actor
Client
Server
Database
User clicks 'join game'
Display join game modal
User submits join code
Add current user to game
Add current user to game
Return joined game
Return joined game
Display game screen
Request prices
Request prices
Return prices
Return prices
Display game screen

7. Buy / Sell


Actor
Client
Server
Database
Buy or sell some of a coin
Send trade data
Request current price data
Return current price data
Verify current price is less than or equal to trade price
If no, return error
If yes, create trade at current price
Return trade data
Return trade data
Update cash and coins

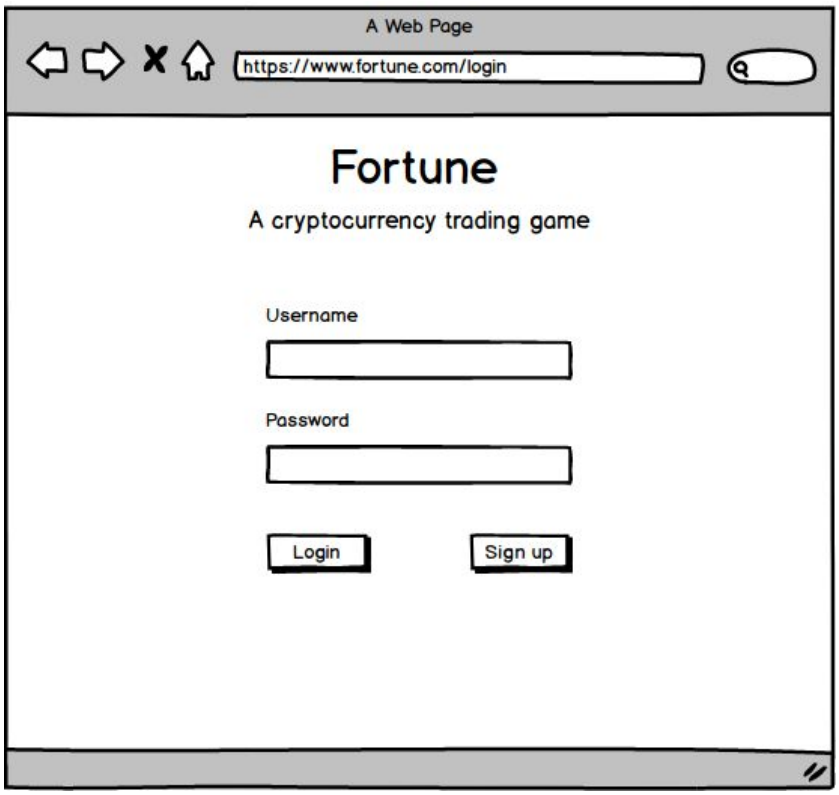
Navigation Flow Map



This diagram shows the flow of how a user navigates the website to play the game. We intend for the game to be simple and intuitive, meaning many screens are similar in features and are easy to understand at first glance. When a user first visits the website, they are brought to the login page, where they can either login or create an account. After successfully passing the login/registration pages, they are brought to the landing page / page for the current global game. From there, a player can view their standing in the global game or visit the play screen where they can create a game or navigate to an existing game, which uses the same model as the global game. Once in a game, the player has the option to view information of various cryptocurrencies (“coins”) or view the leaderboard for that game.

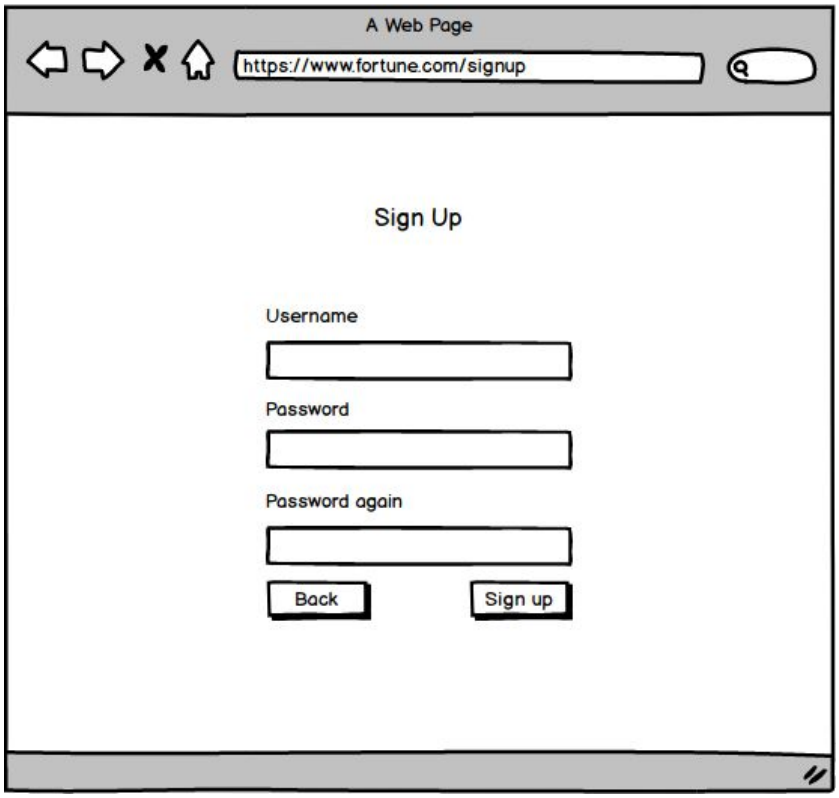
UI Mockup

Figure 1 - Log in



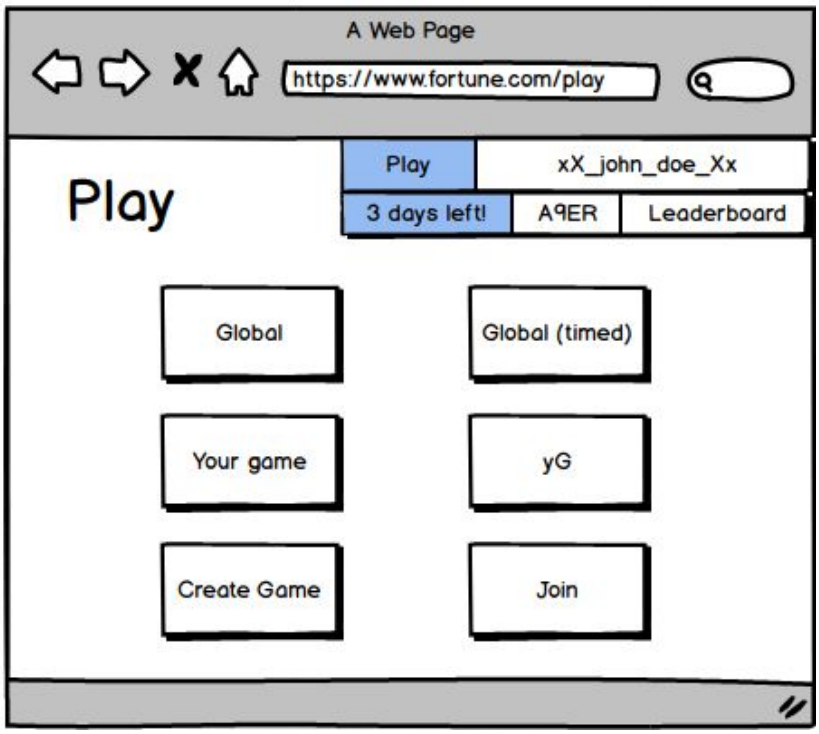
This is the page displayed when a player would like to log into their account. A “Forgot your password?” option can be implemented as time allows. This is the first page the player is brought to when first visiting the website, since an account is needed to display information about the global game. A button is shown if player’s need to create an account.

Figure 2 - Sign up



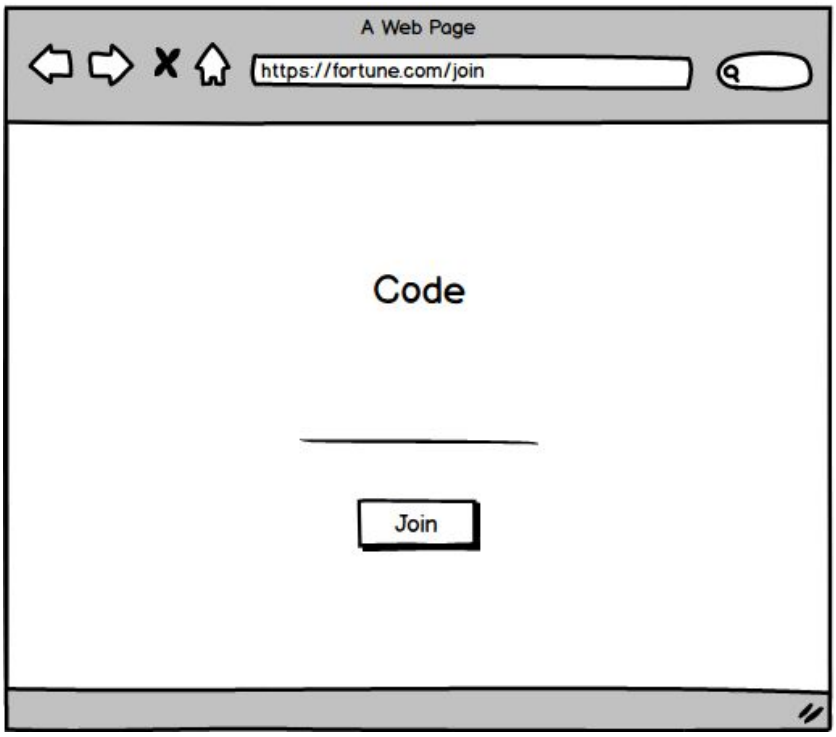
This is the page displayed when a user creates their account. They supply a username and password and an account is created. If time allows, we also thought about adding functionality to connect to a third party account such as Facebook or Google and also to sign up via email.

Figure 3 - Play



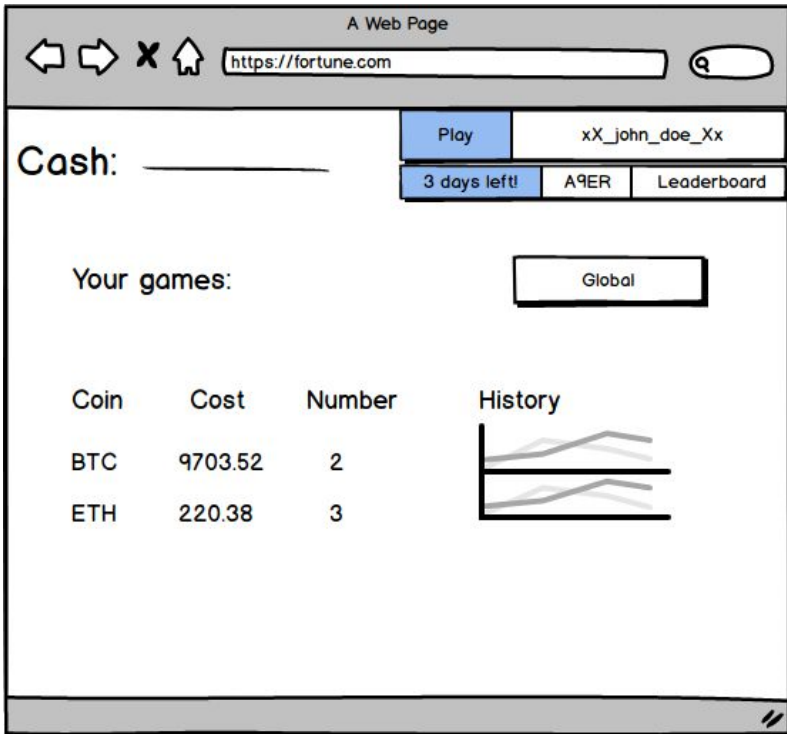
This is the page that displays all the various games a player can choose from. It will be a page of varying length given the different games a player is a part of. There will also be a list of pending games a player can join. Also displayed will be the options to create a game and join a current game.

Figure 4 - Join



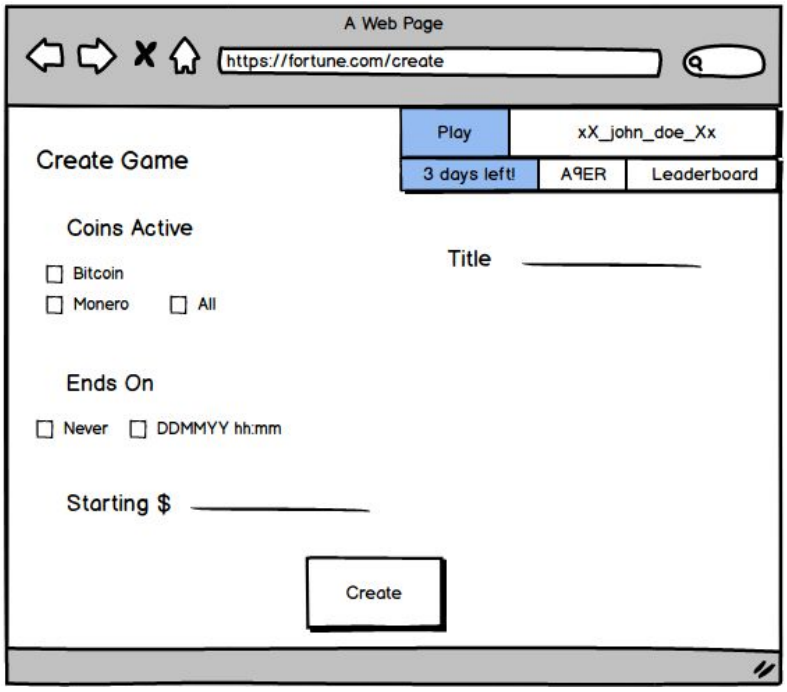
This is the page a player is brought to when they click on “Join” from the “Play” page. The player enters the 4 digit code that is provided on the “Create a game” page and is brought to that game page.

Figure 5 - Landing page / Global Game Page



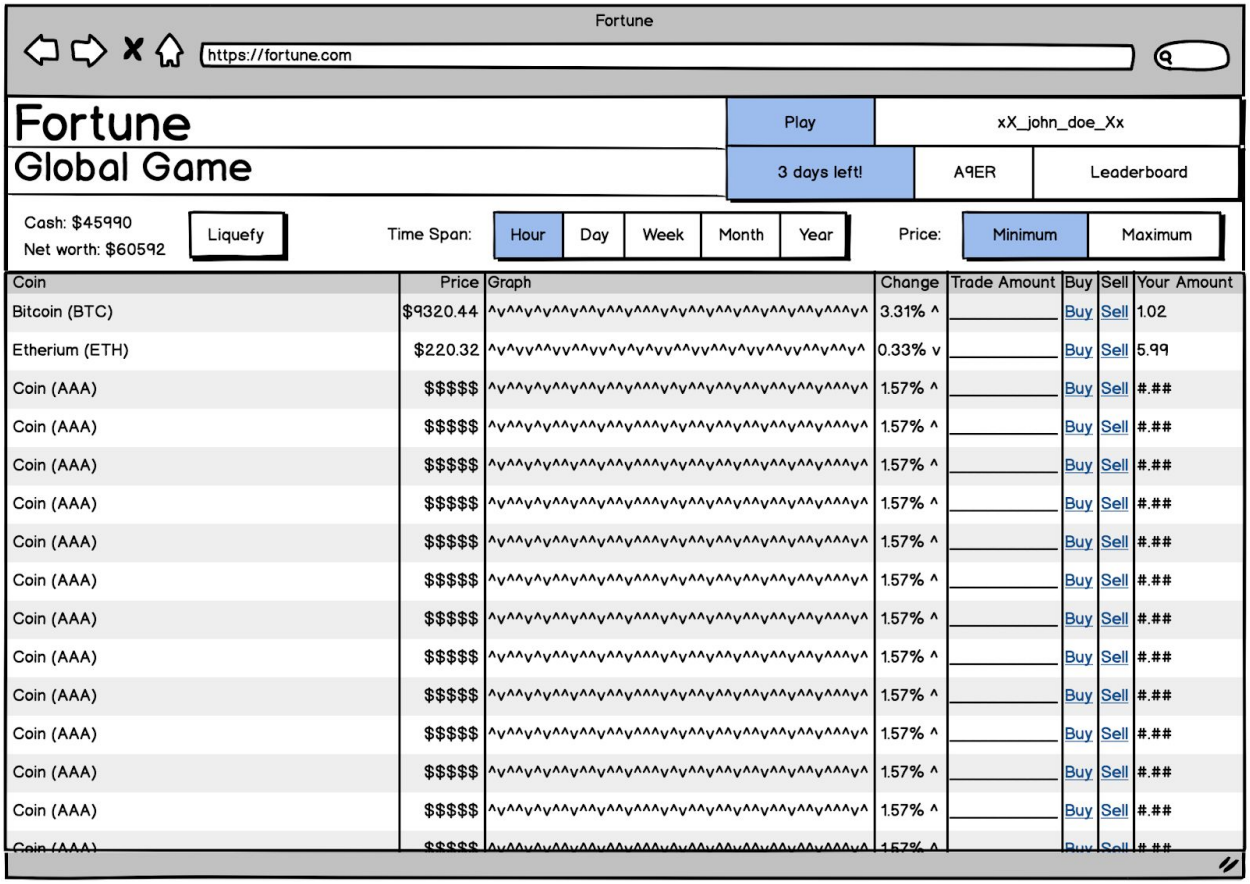
This is the landing page as well as the global game page. This page will display information for various cryptocurrencies and show the leaderboard for the global game. It will show their information such as how much virtual money they currently have and how much of each cryptocurrency they currently own. This page also provides a way to quickly access current games a player is a part of.

Figure 6 - Create game



This is the “Create a game” page where players are shown the various different options (not all shown) they can choose for their game, such as the different cryptocurrencies they can use or the duration of the game. Once the player presses create they are taken to the game page for the game they just created.

Figure 7 - Game



This is the page that is shown for a given game. It shows pricing information for different cryptocurrencies as well as a graph showing historical price change for each. A player can click on the name of each coin and view more pricing and historical information about it. On the far right of each line is the amount of that specific coin has. Near the player's amount is where a player can buy and sell that particular coin. Since prices can vary between different cryptocurrency exchanges, an option to display the highest price and lowest price of all the exchanges is shown above the buy/sell option. Above all the graphs is a bar showing the amount of cash a player has and the player's net worth (the total amount of cash + cryptocurrency value). There is also a liquefy button that lets a player sell all their currently owned coins and converts it to cash. Also displayed on the bar is a panel that allows players to change the time span shown in the graphs and percent change.