CRYPTOLEAGUE

Team 8: Design Document

Varun Gupta, Ritwik Bhardwaj, Utkarsh Jain, Nisarg Kolhe

February 2, 2018

PURPOSE

With the huge amount of attention cryptocurrencies have received in the recent years, a lot of people are gaining interest in cryptocurrencies. But, the sheer complexity of getting started and the volatility of the cryptocurrency market can be overwhelming for many. Thus, we are building a web app which lets the users build their cryptocurrency portfolio using virtual money so that they can practice and make informed investments. The players will be able to participate in different types of leagues and compete with other players for making the highest amount of profit on their portfolio and winning our in-game tokens in the process. This will not only help the players understand the cryptocurrency market but also encourage them to win our in-game tokens to make it to the global leaderboard. This unique competitive nature will encourage players to learn from each other, and get a better understanding of the cryptocurrency market when investing with real money and subsequently contribute to the growth of cryptocurrencies.

Functional Requirements:

Login/Profile:

As a user I would like to:

- Be able to register for a CryptoLeague account using social platforms like Facebook or Google.
- Be able to see my current in-game tokens.
- Be able to see my previous leagues and my current league statistics.

As a developer I would like to:

- Be able to replenish user's in-game tokens if they do not have enough tokens to join the lowest league.

Dashboard:

As a user I would like to:

- Get an overview of my user profile.
- Get an overview of my current league or have an option to join a league.
- Be able to read the current cryptocurrency related news.

Market:

As a user I would like to:

- Be able to see the live cryptocurrency market based on the coinmarketcap's data.

League:

As a user I would like to:

- Be able to see the buy-ins and payouts for different levels of leagues.
- Be able to see the rules for different leagues.
- Be able to join a league or see the status of my current league.

As a developer I would like to:

- Be able to restrict users to join only one league at a time.
- Be able to have multiple leagues running at the same time.

Leaderboard:

As a user I would like to:

- Be able to see the global leaderboard of all players.
- Be able to see the leaderboard of my current league.

Portfolio:

As a user I would like to:

- Be able to make my cryptocurrency portfolio when joining a league.
- Be able to easily modify my portfolio before the league starts.
- Be able to select a captain coin whose gains/losses will be doubled.
- Be able to visualize my current league portfolio in form of a pie chart.
- Be able to see the start and end time of my current league.

- Be able to see my current rank in my current league.

Non-functional Requirements:

Security:

As a developer I would like to:

- Prevent the users from seeing the portfolio of other users in the league until the league is over.

Front-end:

As a developer I would like to:

- Have a front end written in Angular which will use APIs served from the backend to create a fully functional web-app.
- Make the UI seamless, easy to use and responsive such that users can access it from their mobile phones as well.

Back-end:

As a developer I would like to:

- Host my backend on AWS.
- Have a backend developed in Node.JS, which will serve the APIs to drive the frontend.
- Have my backend host concurrent leagues.

- Make sure that a player can join a league within 500 milliseconds.

Database:

As a developer I would like to:

- Store all of the data in a MongoDB database.
- Make sure that all users have unique wallet addresses to secure their tokens.

DESIGN OUTLINE

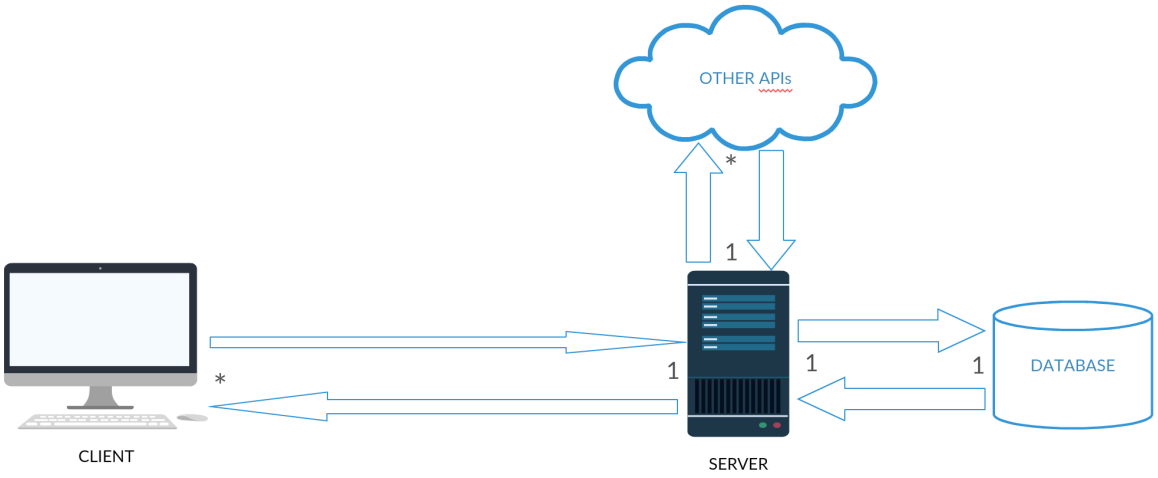
Design Decision

CryptoLeague uses a classic Client-Server model. The user clicks on different activities on the screen and the front end renders the tasks and calls the backend using the APIs. The server then calls upon other APIs and queries the database in order to retrieve more information and ensure the task is completed accurately and quickly.

- Client:
  - Client will make JSON requests to server API.
  - Client will receive JSON responses from server API.
  - Client will use the JSON package and display results accordingly (using Angular).
- Server:
  - Processes all client requests and returns the data as JSON packages.
  - Calls other APIs to receive real-world cryptocurrency data.
  - Queries database for relevant information.
- Database:
  - Database will store information such as the users, leagues, portfolios etc.

High Level Overview of the System and how they Intersect

The front-end client and the server forms many-to-one architecture where multiple clients communicate with the server simultaneously. The server utilizes many external APIs to obtain live data about the cryptocurrency market and then uses it to build the response. The server will query from our MongoDB database to obtain information about users, leagues, portfolios, etc.



The Client-Server model of CryptoLeague

DESIGN ISSUES

Functional Issues

1. How would players create their portfolio?

    - Choice a: By entering the amount in a text field.
    - Choice b: By using a draggable pie chart.
    - **Choice c: Both by entering the amount manually in a text field and using a draggable pie chart.**

    We decided to provide the users two options to change their portfolio, which allows for a seamless user experience. Providing a draggable interface, will allow the users to use up all the money they are provided with to create their portfolio as they will only have to drag the slider instead of knocking their heads over perfect numbers which would add up to the total portfolio value. Also, by having an option of a text field will allow users to enter the perfect numbers for which they are having a hard time dragging the slider into the perfect position. Thus, instead of just having either of the options, having a combination of both will let the users create their portfolio faster.

2. Where would the timebar be placed within the league page?

    - Choice a: Have the time bar placed at the top of the page.
    - **Choice b: Have the time bar placed at the bottom of the page.**
    - Choice c: Display the time left in the leaderboard.

    Earlier we decided on placing the time bar at the top of the page but after analyzing the screen space on the league view of our game, we noticed that the bottom of the screen had visibly non-appealing whitespace left. So, we decided that having the time bar placed on the bottom of the screen would be a much better design

option. We were also thinking of displaying the time left on the leaderboard but decided against it as it made the leaderboard look cluttered.

3. How should the players view and change their profile information?

    - **Choice a: Have a separate views for profile and settings.**
    - Choice b: Have a single view for viewing and changing the settings.

    We decided to have a two separate views for viewing the profile and setting. The profile page will show the user's profile, past leagues' stats, etc and the setting page will be where the user edits things like their username, email, password, etc. We thought that having separate views for viewing brief information about the user and his leagues' statistics and for changing the profile settings of the user like their avatar, username, etc would be more feasible and the views will not be cluttered.

4. How should the users be notified about the updates in the game?

    - Choice a: Use desktop notifications.
    - **Choice b: Use email notifications.**

    We decided that email notifications would be the best way to notify users about the updates in the game because people are not on their registered devices all the time and would not get the desktop notifications if they are not on their registered device, however, they usually have access to their emails, no matter what device they are on. Also, email notifications are saved on your email and can be viewed for future references whereas the desktop notifications are not saved anywhere.

5. When would the players be able to see each other's portfolio in a league?

    - Choice a: Players can see each other's portfolio anytime they wish.
    - Choice b: Players can see each other's portfolio only after the league starts.
    - **Choice c: Players can only see each other's portfolio after the league ends.**

In order to maintain the competitive nature of the league and add some suspense until the end of the league, we decided to show the other players' portfolio only after the league ends.

Non-functional Issues

1. What will be the different login and signup options for the users?

    - Choice a: Sign Up using email.
    - **Choice b: Sign Up using 3rd party OAuth (Google and Facebook accounts).**
    - Choice c: Give the user option to signup with both.

    We decided to give user the option to sign up using Google and Facebook accounts. After a lot of discussion and research, we came to the conclusion that nobody likes to have a new password to remember and people usually sign up using their Facebook or Google accounts. According to loginradius.com, around 73% of people prefer 3rd party OAuth logins. We thought having traditional sign up with the 3rd party signup would be redundant and thus, we decided to go with only Facebook and Google logins.

2. What language/framework are we going to use for front-end?

    - **Choice a: Angular**
    - Choice b: React
    - Choice c: Ember

    We decided to use Angular because most of our team members have expertise in the language and thus, it will be the best choice for fast development given the short timeframe we have for this project.

3. What language are we going to use for back-end?

    - Choice a: PHP
    - Choice b: Golang

- **Choice c: Node.js**

Since we are using Angular for front-end, we decided to use Node.js for back-end as both work seamlessly together. Moreover, most of our team has experience with Node.js and thus, it suits our project best, given the short timeframe of the project. Also, since we will be using JSON objects in our database, Node.js has the easiest support for handling JSON objects and its related queries.

4. What database software are we going to use?

    - Choice a: MySQL
    - Choice b: Firebase
    - **Choice c: MongoDB (NoSQL)**

    We chose MongoDB to host our database because according to our analysis it fits our criteria the best. We eliminated MySQL because we can not store JSON objects in MySQL, whereas it can be done using MongoDB. Also, comparing to Firebase, MongoDB is more compatible with our backend.

5. How are we going to host our backend services?

    - **Choice a: AWS**
    - Choice b: Azure
    - Choice c: Virtual Private Server providers such as Digital Ocean

    We decided to host our app in AWS because we already have a EC2 instance setup on AWS for our web app. Moreover, we have AWS student credits, which would let us host our app without incurring any extra costs.

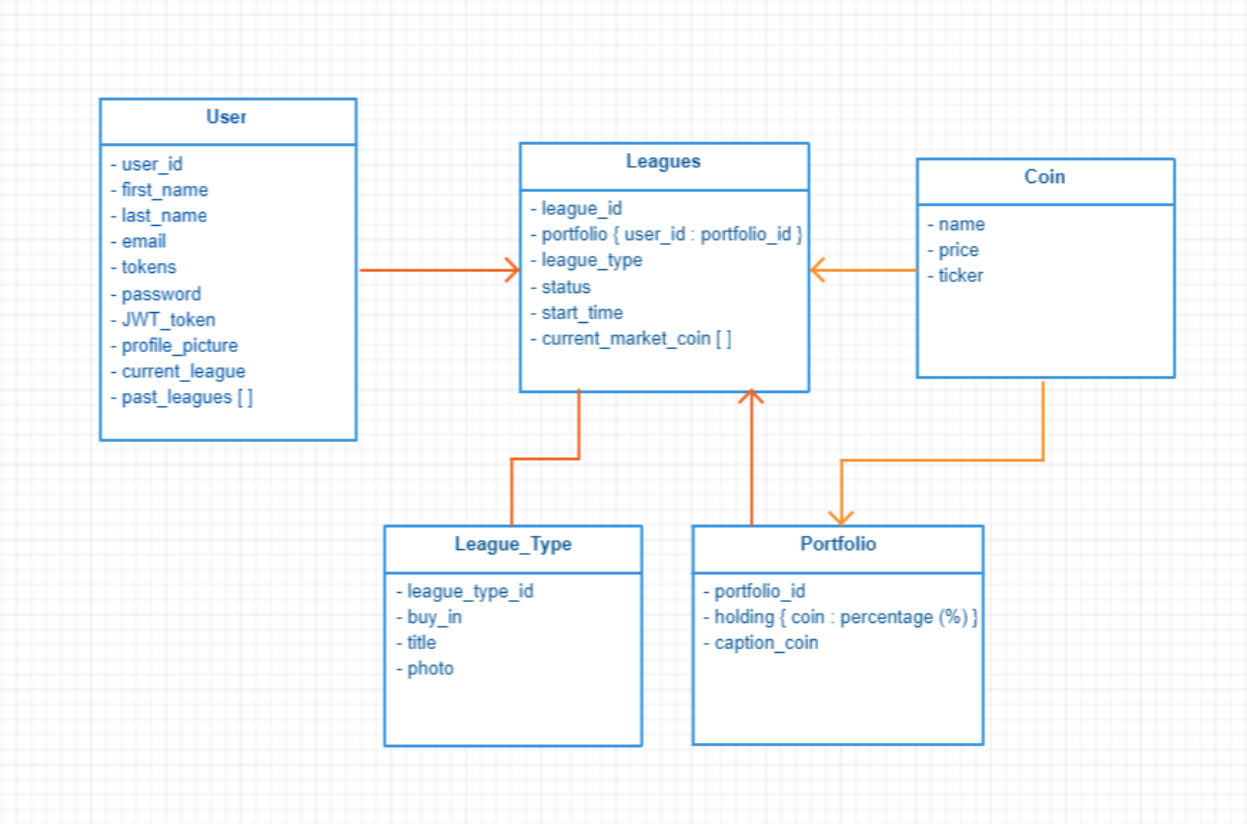
6. What source would the users get their cryptocurrency news from?

    - Choice a: Google feed api
    - **Choice b: Use the newsapi.org API**
    - Choice c: Scrape news websites for the news

We looked into different news sources for getting the up to date cryptocurrency news for our users. We decided to use the newsapi.org api because they get news from a plethora of sources and thus, would be the best way to get a variety and good quality content to our users. This would help them make informed decisions while making their portfolios. We decided against the Google feed api as it has been deprecated. We also looked into scraping different news websites, but found that we would first need permissions from them to do that. Looking at all these different factors, we decided to use the newsapi.org API.

DESIGN DETAILS

Description of Data Classes and their Interactions



Class Diagram

This is a mockup of our class diagram for our project. This diagram will satisfy our basic functional requirements.

Users:

1. This represents the users in the game.
2. This is the part where all the individual tokens, statistics such as photo, fname, lname are stored about the user and updated as users change their profiles.
3. The portfolios are mapped according to the user_id.

4. New instance will be created when a new user signs up.

Leagues:

1. Represents a league which users can participate in.
2. This is where users in the league and all other stats are stored.
3. A new league is created whenever there are no leagues or there is no space in the leagues which haven't started yet.

League Type:

1. Represents the type of a league.
2. Stores other information such as minimum buy-in for league, league_id.
3. Only four league types: Bronze, Silver, Gold, and Platinum.
4. When a new league is created, it is mapped to one of the four leagues.

Coin:

1. Represents the crypto coins that are available for the user to choose from.
2. It will store the name, price and ticker of the coin.
3. Name: name of cryptocoin (Eg. Bitcoin, Ethereum etc.).
4. Ticker: the identifier of cryptocoin (Eg. BTC for Bitcoin).
5. Value: real time value of coin which will be pulled from coinmarketcap.
6. Only the top 100 coins from coinmarketcap will be available to the users to choose from for their portfolios.

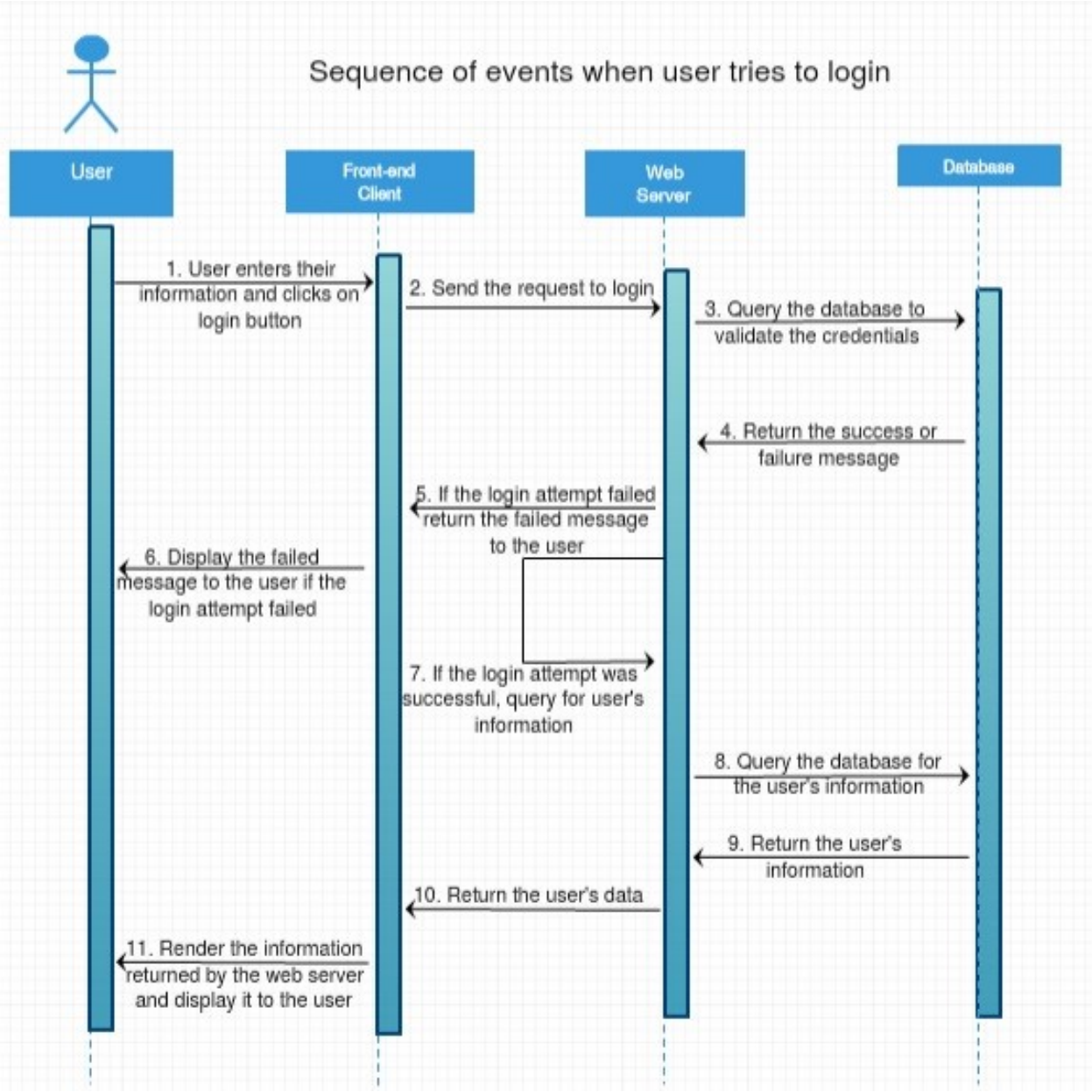
Portfolio:

1. This represents the portfolio with which the user can play a league.
2. It will consist of the coins the user has selected to invest in.

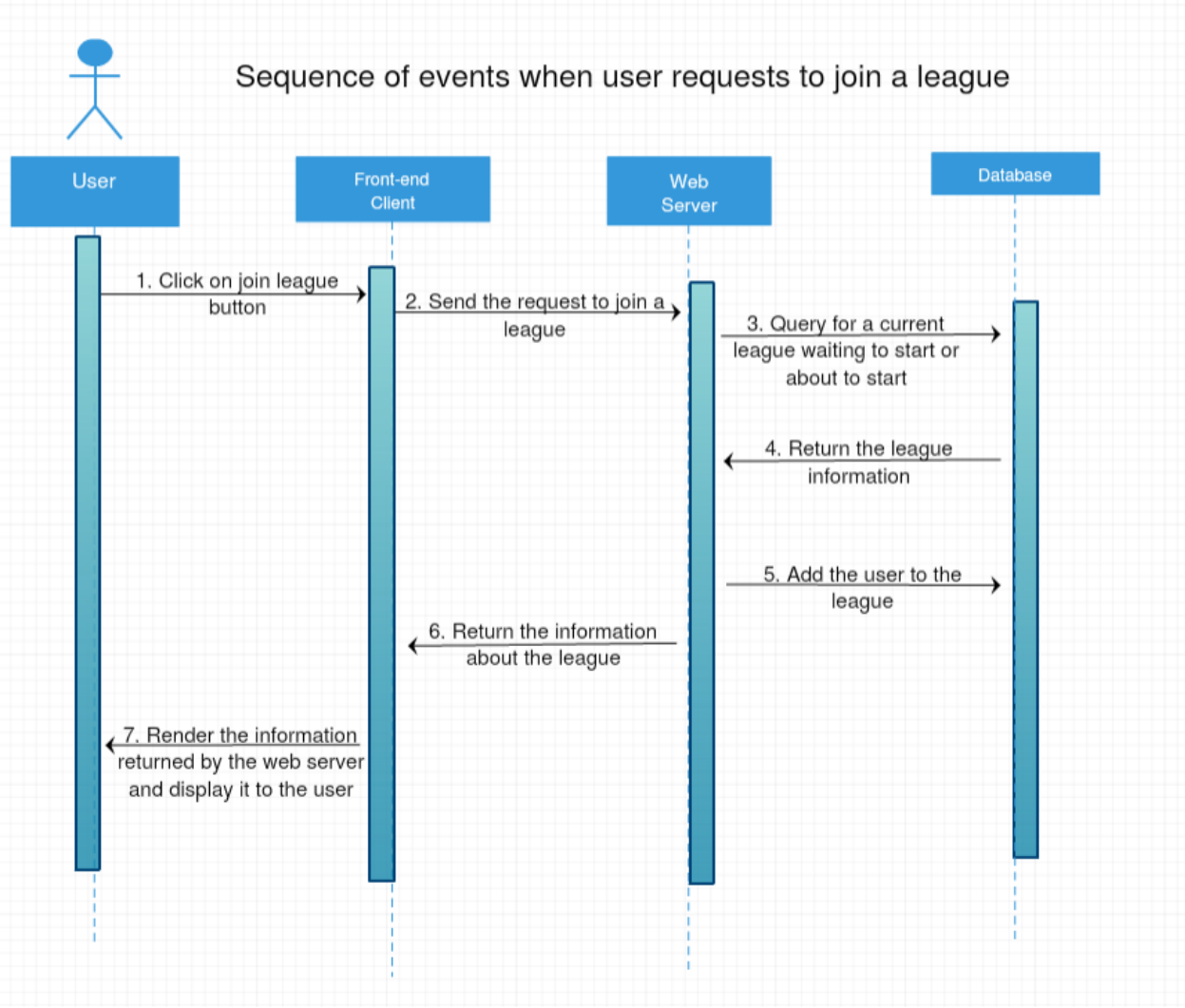
3. It will consist of a captain coin.

    - Captain coin: a coin whose gains or losses will be doubled.

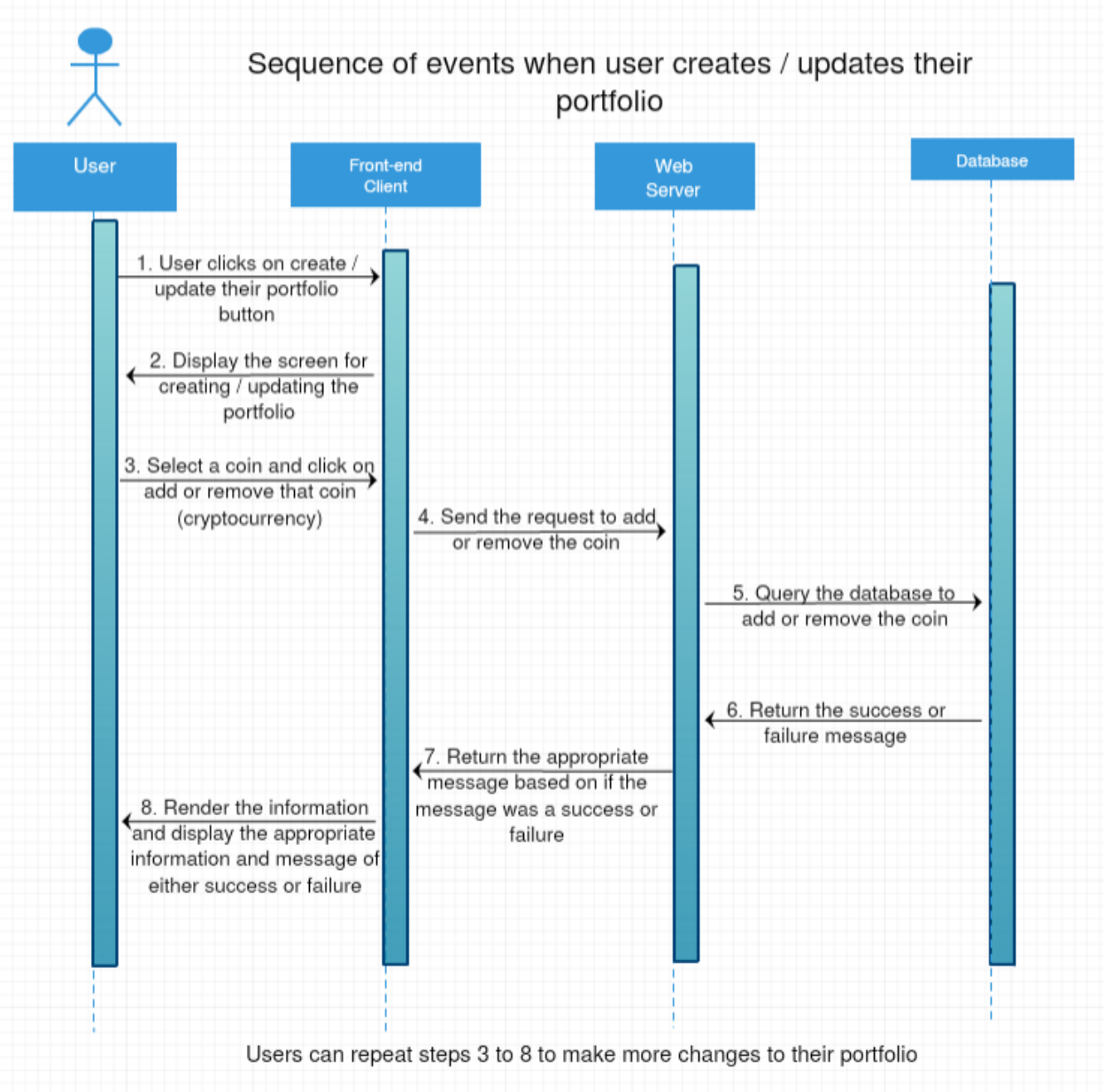
Sequence Diagrams



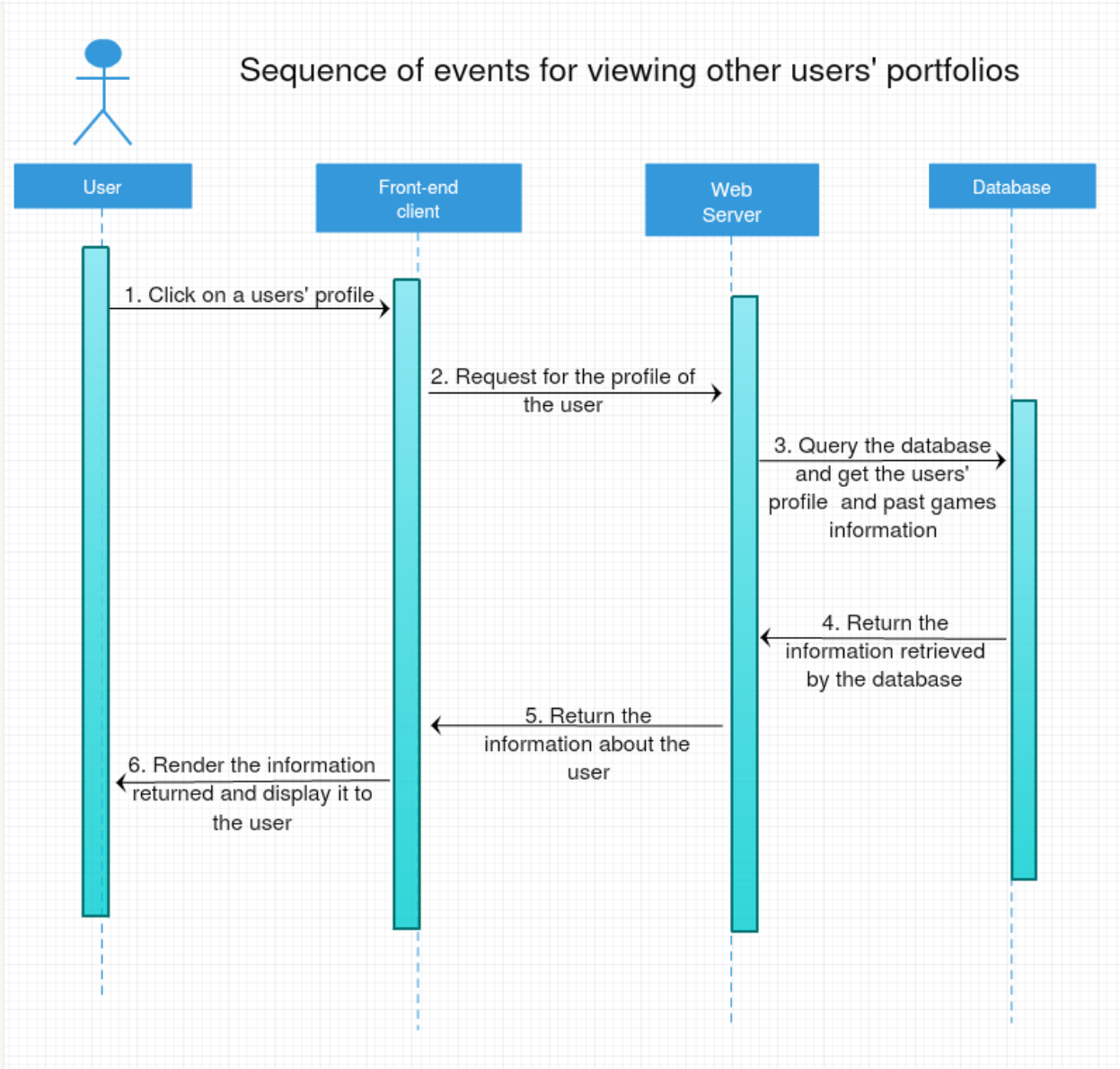
This is the sequence diagram for the event in which the user is trying to login to their CryptoLeague account.



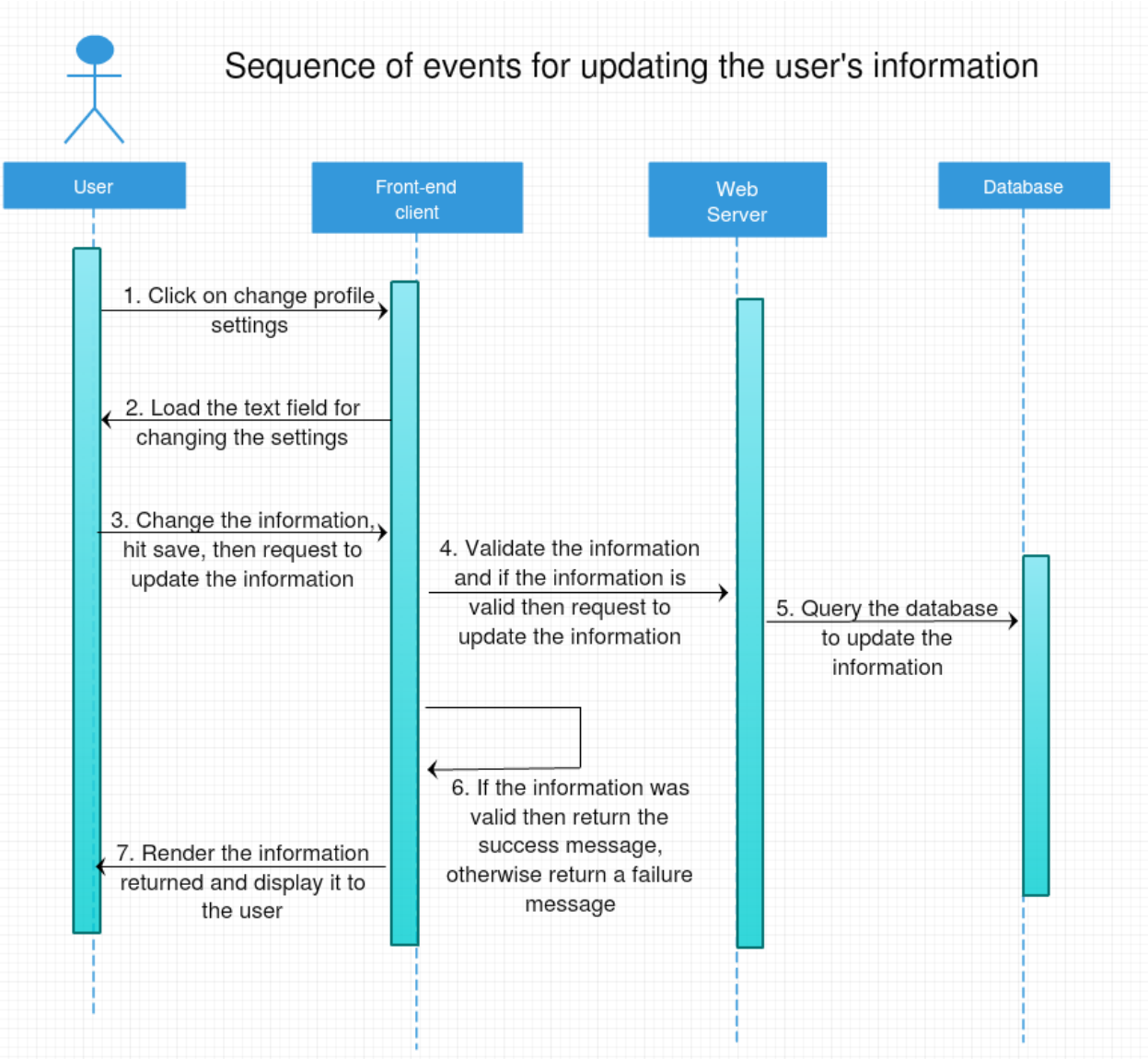
This is the sequence diagram for the event when a user requests to join a league.



This is the sequence diagram for the event when a user is creating or updating their portfolio in a league.



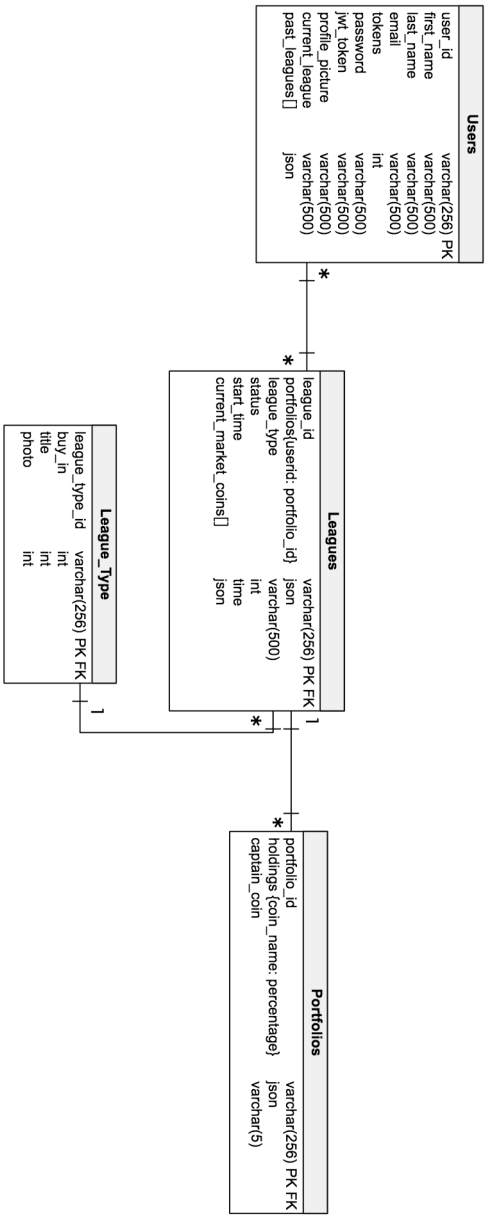
This is the sequence diagram for the event when a user is trying to view another user's portfolio.



This is the sequence diagram for the event when a user is trying update/change their profile information.

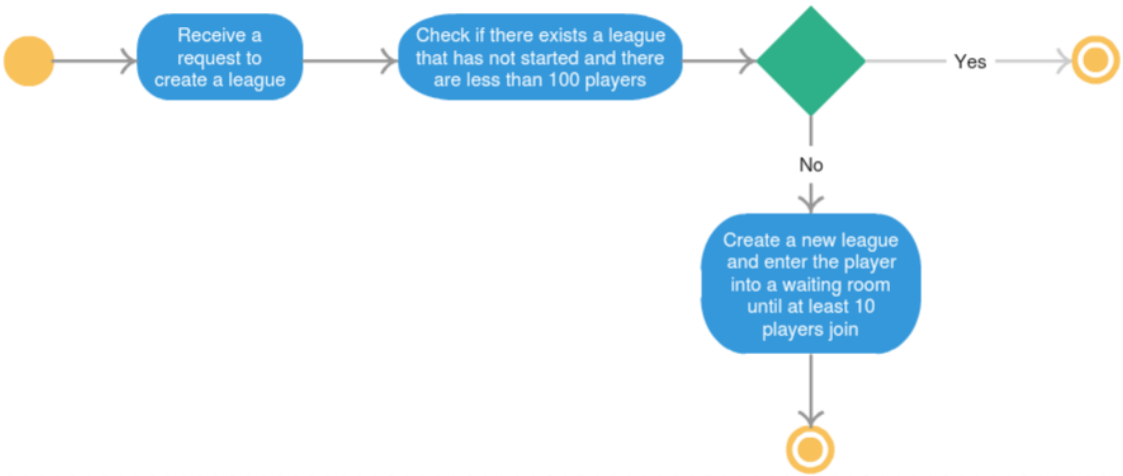
Database Schema Mockup

We are using a NoSQL database which will be hosted on MongoDB. The one to many and many to many relationships have been explained in the database schema mockup using 1…* and *...* respectively.



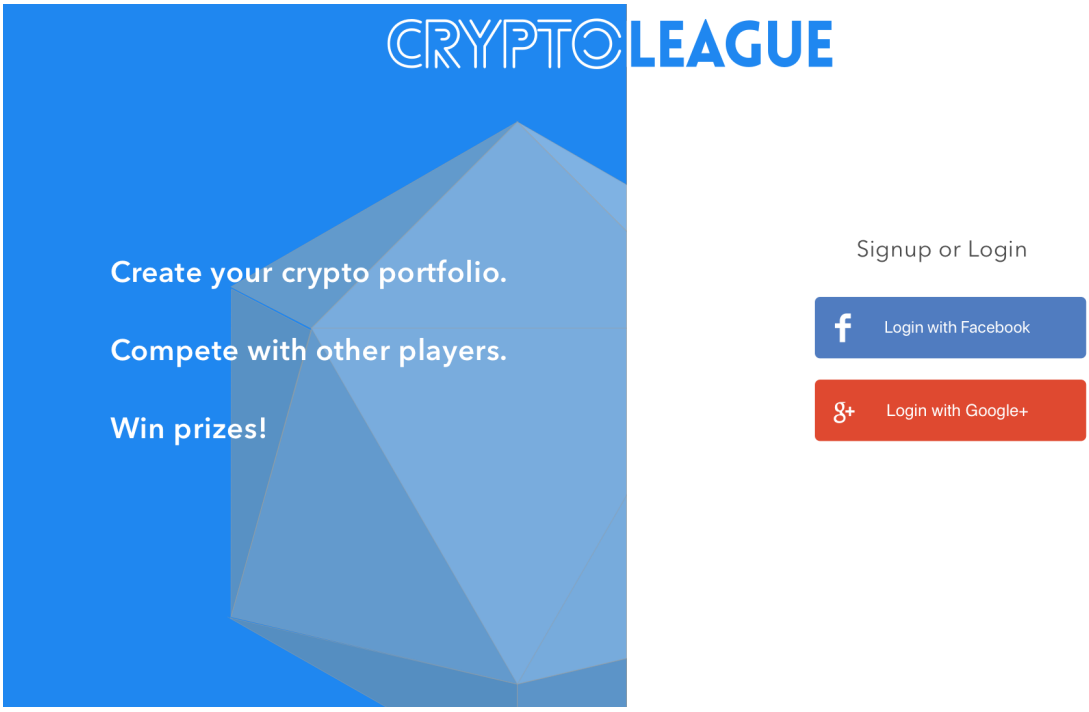
Database Schema Mockup.

Activity Diagram

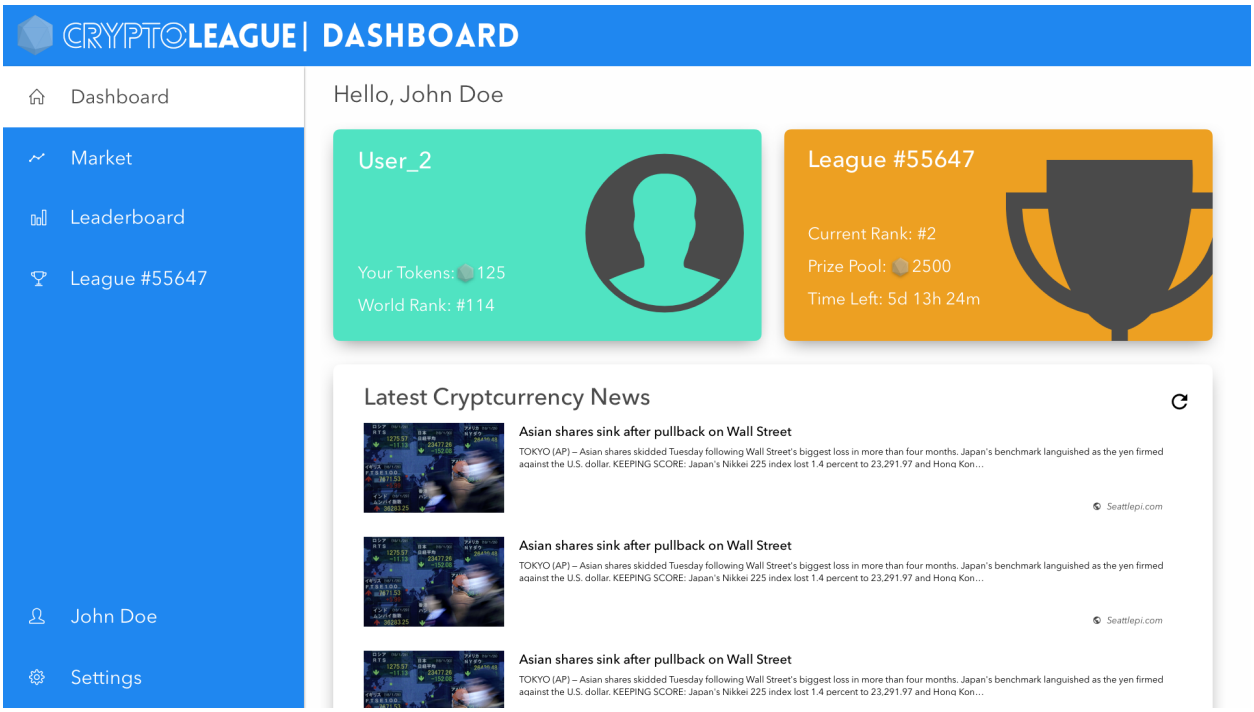


This is the activity diagram for creating a league.

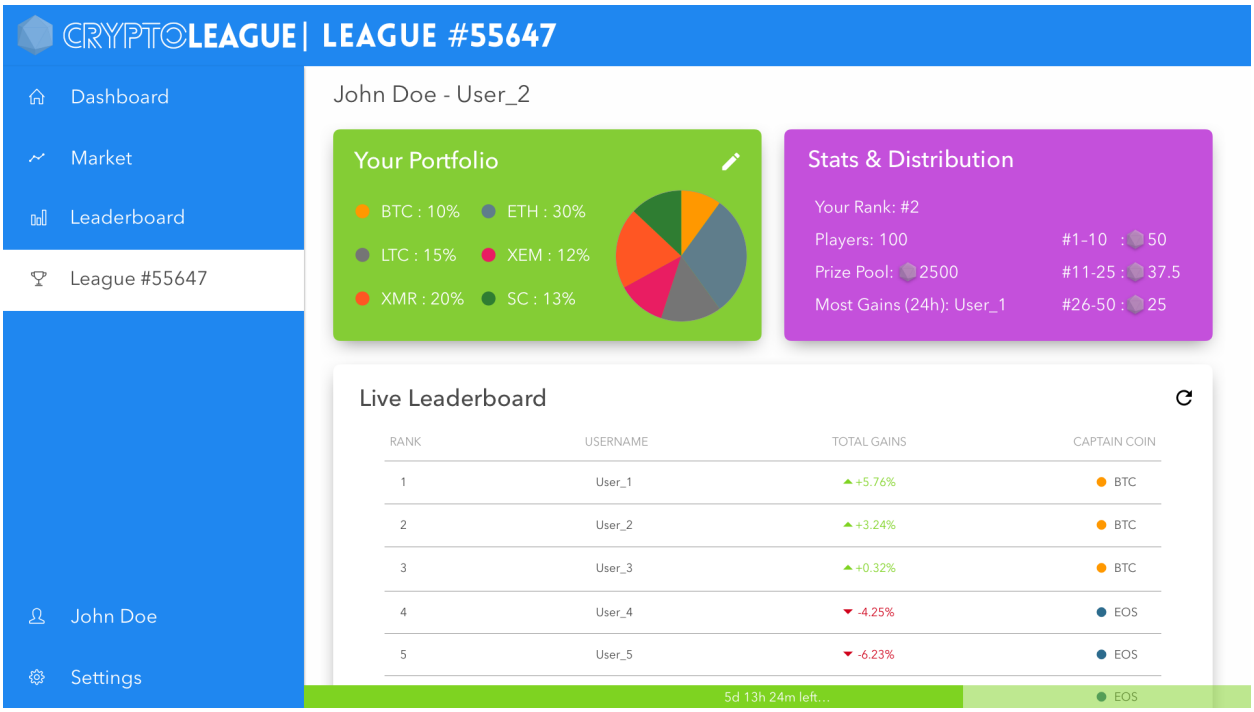
UI Mockups



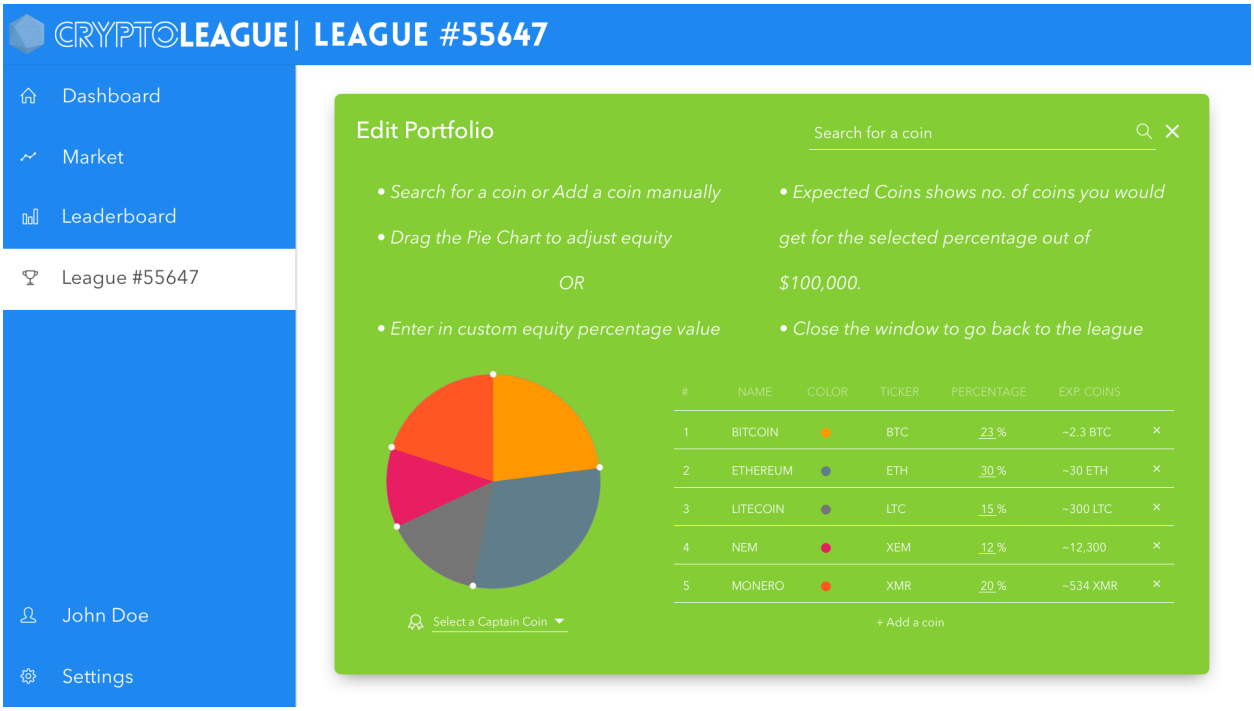
UI mockup of the login screen.



UI mockup of the dashboard.



UI mockup of the current league.



UI mockup of the creating/editing the portfolio view.

Notes:

1. We will have more non-trivial views for which we haven't shown the mockups.
2. We used the online tool Creately (https://creately.com) to make our sequence diagrams, activity diagram, Database schema mockup, and the class diagram.