(

## Brawl Street Specifications

### Action Items

|  |  |  |
| --- | --- | --- |
| **What** | **Who** | **Status** |
| Spec out the service that will create new games automatically.   * Question: how long will new, unstarted games remain visible until they are depreciated? | Don | Need to discuss. Regroup later. |
| Determine which payment gateway we will use and spec out the integration.   * KM - 10/25 - Earl to sign up for PayPal business and report back with any success or blockers. | Sagar | In Progress. |
| Spec how levels work   * Phase 1 - based on current spec on pg. 4 * Phase 2 - determine if a more complex and long term level and skill-based match system is necessary. * KM - consider broader discussion of levels   + I.e. page 4 levels versus Earl’s comment on page 11. | Don | In Progress |
| Spec out how the recruiter affiliate fee will work   * KM (10/25) Kieran to flesh out and run by Don/Earl.   + See pg. 69 | Kieran | In Progress |
| Talk with IEX about which API offering we should use and getting a sandbox account   * KM - 10/25 - Earl needs to sign up for an account to access sandbox. | Earl | In Progress |
| Talk to an attorney about:  1) is this considered a game of skill,   * Legal is writing opinion that game will be game of skill. (done)   2) are there any geographic restrictions we need to spec out for the app   * Need to capture location. * **Confirm**: do we need a geo-location confirmation to align to what the user enters. **YES - geolocation confirmation on location** * **Question:** do we need to geo-locate everytime a user logs in (i.e. if it’s illegal in a state, am I not allowed to play in that state.)   3) what information do we need from players for financial reporting (ie SS#) and 4) are there any restrictions we must adhere to when a player withdraws funds?   * Confirm the above. | Earl | In Progress |
| Need specs on notification settings   * Need to confirm list on pg. 84 is the final list of required notifications. | Don | In Progress |
| Outline the Revenue Model   * Walk through revenue model and just confirm everything. | Don | Done |
| Review of DB Schema   * KM to provide DB schema to Don for review. | Kieran | Not Started |
| Withdraw Funds Workflow & Third Party Integration | Don | Not Started |

### 

### Overview

Brawl Street is an online game where players can compete against one another in trading stocks and growing an investment portfolio within a specific period of time. This is a betting game so players will pay an entry fee to play a game and depending on the type of game, the winner(s) will participate in the pot (sum of entry fees).

### How the Game is Played

Players need to register for a new account on Brawl Street.

If they are going to bet against other players, they will need to deposit funds into their Brawl Street Wallet.

A Player can create a game or join a game that another player has created. Each game has an entry fee, start date and time and duration. There must be at least two players in a game with no maximum number of players.

When a player joins a game, they will pay the entry fee using funds from their Brawl Street Wallet. A player is only allowed to join a game prior to its start date and time.

When a game begins, each player is given $200,000 (play money) to invest in U.S. stocks. Each player may buy and sell stocks throughout the duration of the game and the goal is to grow the overall value of their portfolio.

Brawl Street has a real-time feed of stock prices and related data from the US Stock Exchange and thus all transactions and reporting are based on actual market data.

At the conclusion of the game, each player’s stocks are sold and winners are determined by the final value of their portfolio. The pot (made up of the entry fees) is divided among the winner(s) of the game according to the type of game (explained below)

Players can create new games that other players can join and/or they can join games created by other players. A single player may enter into as many different games as they like and can play multiple games simultaneously.

Payers can withdraw funds in their Brawlsteet Wallet at any time.

#### Types of Games

When a player creates a new game, they must specify what type of game is it will be. The following are the types of games:

**50/50**

In this game, the top 50% of the players split the Pot.

**20/80**

In this game, the top 20% of the players split the Pot.

**Top 10**

In this game, the top 10 players split the Pot. Please note that 11 or more players are required for this type of game.

#### How Levels Work

Each player will be assigned a Level and will initially be assigned Beginner.

As a player wins games, they will progress to higher levels based on the number of games they have won.

When creating a game, the player can set the max level allowed. That way, a beginner would not be excluded from playing up, but an Advanced would be excluded from playing down.

|  |  |  |  |
| --- | --- | --- | --- |
| **Level Name** | **Games Won** | **Level** | **Icon / Badge** |
| TBD | 0 | 1 | TBD |
| TBD | 25 | 10 | TBD |
| TBD | 50 | 20 | TBD |
| TBD | 100 | 30 |  |
| TBD | 250 | 40 |  |
| TBD | 500 | 50 |  |

**Tier Names**

Newbie (1-9)

Analyst (10-19)

Advisor (20-29)

Broker (30-39)

Professional (40-49)

Big Boss (50)

### 

### How Referral Fees Work

Brawl Street would like to offer players a fee for referring other players to Brawl Street. The fee will be a percentage of the winnings of the referred player.

The **Referral Percentage** will be a percentage stored in the **Global** table. This value might change over time. For example, Brawl Street may offer a 5% Referral Fee Percentage for referrals made within the first three months of launching. Any referrals made during this time would carry a 5% referral percentage. Thereafter, Brawl Street may change it to a lower percentage like 3%.

Player A may refer Player B to Brawlstreet. When this occurs, a record is entered into the Players table for Player B. RD will be the ID of Player A and the ReferralPercentage will be a copy of the ReferralPercentage in the Global table (or perhaps the Players table - see discussion).

Brawlstreet may change the ReferralPercentage in the Global table at any time. This does not affect the ReferralPercentage for any Players in the Players table. This would only affect new Players added to the Players table.

The ReferralPercentage is the percentage of winnings that the Referred By Player will receive when a referred player wins.

At the end of a game, Brawl Street will insert records into the Account Transactions table for each winning player. For each winning player, Brawl Street will need to confirm whether or not another player is due a referral fee by the following conditions:

* ReferredByPlayerID exists for the winning player in the players table
* ReferredByPlayerID is in an Active status meaning their account has not been canceled.
* ReferredByPlayerID has played a Game within the past 60 days.

If the above conditions have not been met, then no referral fee is provided. However, if a winning player was referred by another player and that player meets the above conditions, then three records will be entered into the Transactions table. For example purposes, let’s say that Player A won $100 and Player A was referred by Player B and there is a 3% Referral Fee.

* Record inserted into the Transactions table for $90 for the Player A. This is the winnings minus the Brawl Street 10% fee.
* Record inserted into the Transactions table for $3 for the Player B. This is the 3% Referral Fee of Player A’s winnings; value is found in Players.ReferralPercentage.
* Record inserted into the Transactions table for $7 for the Brawl Street. This is the net of the Brawl Street fee minus the Referral Fee.

\*When records are entered into the Transactions table, Brawl Street should be PlayerID = 1. That is how we will track Brawl Street transactions.

If a winning player was not referred by anyone then only two transactions would be entered:

* Record inserted into the Transactions table for $90 for the Player A. This is the winnings minus the Brawl Street 10% fee.
* Record inserted into the Transactions table for $10 for the Brawl Street.

**For discussion:**

We did discuss having a promo time period for each new player. For example, new players will get a referral percentage of 5% on players they refer within their first 30 days and then 3% thereafter. If we want to implement this, then we would need to add ReferralPercentage1 and ReferralPercentage2 to the Global table.

The other issue that arose is if changes to the referral fee should change existing players. Once Player A refers Player B, the percentage of winnings that Player A receives on Player B’s winning **cannot** change. However, can the Referral Percentage change for new referrals? Let’s say that when Player A signed up, the Referral Percentage was 3%. But then Brawl Street changes the Referral Percentage in the Global table to 2%. At that time, will Player A receive 3% of any new referrals, or will Player A receive the 2%? If Player A is locked into the 3% then we need to add a field into the Players table to hold this info. I would suggest naming it something like NewPlayerReferralPercentage.

### 

### Brawl Street Revenue Model

Players must pay an entry fee into the “pot” for each game.

At the end of the game, Brawl Street will fund each winning player’s account with their gross winnings minus a 10% Game Fee.

If any of the winning players were referred by a friend, that friend will receive 3% of the winning player’s gross earnings.

Example Payout for a 50/50 Game

Game Details:

* 10 Players
* $100 Entry Fee
* $1,000 Pot

The Gross Winnings are $200 per player for 5 players. Each player would receive the net of Gross Winnings minus the Game Fee of 10%. In this case, each winning player would receive $180.

If any of the winning players were referred by a friend, those friends would each receive $6.

#### Model with Credit Card Fees

The cost for a player to deposit funds will be 3%.

Those funds may be used in multiple games and each turn will net Brawl Street between 7% and 10%, depending on referral payouts.

There is no cost for a player to withdraw money.

Example

Player deposits $100. Brawl Street pays $3.00 credit card processing fee.

The $100 could be used in an infinite number of games, transitioning from player to player. Each game nets 7% to 10% (depending on referral fees) until either the $100 has been fully spent on Game Fees or a player withdraws the money.

### Technology Stack

The game needs to be available via iOS and Android apps and also via a web browser. We are not set on a specific technology, but have been considering:

**Front End**

React

**Web Services Layer**

Python

**Database**

MySQL

We would be interested in using AWS Lambda or Google Cloud Serverless Computing for services layer.

### 

### Database Structure

#### Global

* All dates and times are GMT

#### Players

|  |  |
| --- | --- |
| PlayerID | Unique ID |
| FirstName | Text |
| LastName | Text |
| Address | Text |
| City | Text |
| Region | Text |
| PostalCode | Text |
| Phone | Text |
| Username | Text |
| PlayerStatusID | Number; 1:Active 2:Inactive 3:Suspended |
| WalletAmount | Currency |
| PointsAmount | Number |
| Pointlevel | Number; Need to create a related list table. |
| EmailAddress | Text |
| PWDSecretKey | GUID; Used in link for forgot password. |
| PWDSecretKeyExpirationDTTM | DateTime |
| EmailSecretKey | Text |
| EmailSecretKeyExpirationDTTM | DateTime |
| ReferredByPlayerID | Number |
| ReferralPercentage | Number |

#### Games

|  |  |
| --- | --- |
| GameID | Unique ID |
| GameNumber | Number; incremented |
| Name | Text |
| TypeID | Number |
| GameStatusID | Number; 1:Open 2:Active 3:Completed 4: Abandoned |
| PrivateInd | Number (1:Public; 2:Private) |
| EntryFee | Currency |
| Pot | Currency |
| Duration | Number (in hours) |
| StartDTTM | DateTime |
| CreatedDTTM | DateTime |

#### Invitations

|  |  |
| --- | --- |
| InvitationID | Unique ID |
| GameID | Number |
| PlayerID | Number |
| FriendID | Number; PlayerID of the friend who invited the player. |
| GamePlayerStatusID | Number; 1:Invited 2:Joined 3:Rejected |
| UpdatedDTTM | DateTime |
| CreatedDTTM | DateTime |

#### Portfolios (GamePlayers)

|  |  |
| --- | --- |
| PortfolioID | Unique ID |
| GameID | Number |
| PlayerID | Number |
| GamePlayerStatusID | Number; 1:Invited 2:Joined |
| WinningInd | Number; 1:Yes 2:No; |
| CashAvailable | Currency |

#### PortfolioStocks

|  |  |
| --- | --- |
| PortfolioStockID | Unique ID |
| PortfolioID | Number |
| StockID | Number |
| Shares | Number |
| LastPrice | Currency |
| Value | Currency (calculated) |
| UpdatedDTTM | DateTime |
| CreatedDTTM | DateTime |

#### StockTransactions

|  |  |
| --- | --- |
| StockTransactionID | UniqueID |
| GamePlayerID | Number |
| StockID | Number |
| TransactionTypeID | Number; 1:Market Buy 2:Market Sold 3: Limit Buy 4: Limit Sold 5: Limit Buy Canceled |
| Shares | Number; Decimal 0 |
| SharePrice | Currency |
| TotalPrice | Currency |
| CreatedDTTM | DateTime |

#### LimitOrders

|  |  |
| --- | --- |
| LimitOrderID | Unique ID |
| PortfolioID | Number |
| StockID | Number |
| TransactionTypeID | Number; 1:Buy 2:Sell |
| Shares | Number |
| Price | Currency |
| CreatedDTTM | DateTime |
| UpdatedDTTM | DateTime |
| ExecutedDTTM | DateTime |
| ExecutedPrice | Currency |
| LimitStatusID | Number; 1:Open 2:Executed 3:Canceled |
| StockTransactionID | Number |

#### Notifications

|  |  |
| --- | --- |
| NotificationID | Unique ID |
| CreatedDTTM | DateTime |
| PlayerID | Number |
| Note | Text |
| ViewedInd | Number (1:No; 2:Yes) |

#### AccountTransactions

Note: The transactions table will hold all transactions related to the wallet: deposits, withdraws, winnings, entry fees, etc. The table will hold both debits and credits (+/-)

|  |  |
| --- | --- |
| AccountTransID | UniqueID |
| CreatedDTTM | DateTime |
| PlayerID |  |
| WalletTransTypeID   * Note: This will hold the type of transaction: deposit, withdraw, entry fee, winnings, etc. | Number |
| Amount | Currency |
| GameID | Number |
|  |  |

#### PointLevels

|  |  |
| --- | --- |
| PointLevelID | UniqueID |
| Points | Numeric |
| Name | Text |

#### PointTransactions

|  |  |
| --- | --- |
| PointTransID | UniqueID |
| CreatedDTTM | DateTime |
| PlayerID |  |
| PointTransTypeID   * Note: So on some of these tables, I wasn't sure if we need a corresponding lookup table because there may be so few types. But, to have a really clean database, we should have a table to hold those values. | Number |
| Amount | Currency |
| GameID | Number |
|  |  |

#### LoginHistory

|  |  |
| --- | --- |
| LoginID | Unique ID |
| CreatedDTTM | DateTime |
| IncorrectEmailInd | Number (1:yes; 2:no) |
| IncorrectPasswordInd | Number (1:yes; 2:no) |
| PlayerID | Number (if successful login) |
| IPAddress | Text |
| CookieID |  |

For tracking purposes, I didn’t know if you drop a cookie or use a unique ID from the phone. I added CookieID as a placeholder.

#### Friends

|  |  |
| --- | --- |
| FriendID | Unique ID |
| Player1ID |  |
| Player2ID |  |
| FriendStatusID | Number (1:Invited; 2:Accepted; 3:Rejected; 4:Blocked) |
| CreatedDTTM | Date, Time. |
| UpdatedDTTM | Date, Time |

#### Stocks

|  |  |
| --- | --- |
| StockID | Unique ID |
| Ticker | Text |
| UpdatedDTTM | DateTime |
| LastPrice | Currency |
| Bid | Currency |
| Ask | Currency |
| Open | Currency |
| High | Currency |
| Low | Currency |
| MarketVolume | Currency |
|  |  |

#### StockHistory

|  |  |
| --- | --- |
| StockHistoryID | UniqueID |
| Ticker | Text |
| LastPrice | Currency |
| CreatedDTTM | DateTime |

#### Watchlist

|  |  |
| --- | --- |
| WatchlistID | Unique ID |
| PlayerID | Number |
| StockID | Number |
| CreatedDTTM | DateTime |

#### Global (Settings)

|  |  |
| --- | --- |
| TotalPayouts | Currency |
| TotalPlayers | Number |
| ReferralPercentageStage1 | Number |
| ReferralPercentageStage2 |  |

### IEX Stock Market Data Integration

We are subscribing to a third party service by Intrinio that will provide real time stock quotes. We have subscribed to the IEX Real-Time Stock Prices offering found here:

<https://product.intrinio.com/financial-data/realtime-stock-prices>

You can find the API documentation here:

<https://about.intrinio.com/we-love-developers>

We may need to get their “Firehose” option that streams stock prices to our server in real-time. I believe this might be the best option instead of constantly calling the API.

* KM - 10/25 - we will go with this option.

The table called Stocks will hold current stock prices. The table StockHistory will store a record of each time we receive an update via the API.

* KM - 10/25 - stocks will be updated in realtime.

If we use the firehose option, we will [require a listener](#_x90tzojj0hxj) to accept the data from Intrinio.

Regardless of whether or not we use the firehose option, we will require a [nightly service to purge old data](#_3l3cmrtlq19j).

Both of these can be found in the Web Services and Functions section

## Web Services and Functions

### API: Login

1. Count the number of failed login attempts within the past 10 minutes from the Device / IP Address. If the count is 5 or more, then return a failure: too many login attempts.
2. Authenticate the user using login credentials.
   1. If incorrect credentials, then return failure - incorrect credentials.
   2. If success, then authenticate and return success.
3. Insert a record into LoginHistory to record the attempt and result.

### API: Total Payout Data

This API will return:

* TotalPayouts: Lookup from Global Table
* TotalPlayers: Lookup from Global table.

### API: NavigationCount

This API will return:

* Total Open Games
  + This is calculated by adding the count of public games in an open status that the player has not joined plus the count of private games that the player has been invited to and not yet joined.
* Total Active Games
  + This is calculated by taking the count of records in GamePlayers (joined to Games) where the Game Status is active and the GamePlayerStatus is Joined.
* Total Notifications

### API: Players

#### Post (create new player)

1. Verify that the email address does not already exist.
2. Create record in the Players table.
3. Send confirmation email to verify the email address.
4. Return success or failure

### API: Games

#### Post (create new game)

1. Check to make sure the player has sufficient funds in their wallet for the Entry Fee. If not, return error.
2. Insert record into the Games table.
3. Insert record in the GamePlayers table.
4. Insert a record into the WalletTransactions table for the entry fee payment.
5. Update the Wallet Amount in the Players table.
6. Return success.

#### Get (to retrieve list)

1. Parameters
   1. PlayerID
   2. Game Status ID
   3. GameType
   4. Entry Fee Low
   5. Entry Fee High
   6. Min Duration (hours)
   7. Max Duration (hours)
   8. Invitations Only Ind
   9. Open Games Only Ind
   10. Friends Only Ind - Limits list to games that at least 1 friend has joined
2. Results
   1. GameID
   2. FriendUsername - Username of the Player who invited the Player to game.
   3. FriendCount - count of friends joined the game
   4. Min Entry Fee
   5. Max Entry Fee
   6. Min Duration
   7. Max Duration
   8. All Game fields

### API: Portfolios

#### Post (to join a game)

1. Check to make sure the game is still in an Open status. If it is not, return error.
2. Check the GamePlayers to see if a record already exists for this Game and Player combination.
   1. If exists, then update GamePlayerStatusID to Joined and return success.
   2. If does not exist, then create a record in GamePlayers for the combination and return success.

### API: GameFriends

#### Post (to invite a player to a game)

This API is called to invite a friend to a specific game

#### Get (List of Friends for a game)

This API is called to retrieve a list of your friends and whether or not each one has been invited to a specific game.

### Listener: Routine to accept and process the Data from the API.

1. Receive an updated stock price.
2. Insert record into StockHistory table.
3. Update record in Stocks table with the latest data.
4. Check for any limit orders in the LimitOrders table that are open and would be triggered.
   1. If any Limit Orders are triggered then call the LimitOrderExecution API
5. Update the LastPrice and Value fields in the PortfolioStocks table for records for that StockID

### API: Watchlist

#### Post (add to watch list)

1. Add a record to the Watchlist table

#### Get (retrieve watchlist list)

1. Parameters
   1. PlayerID
2. Results

### API: Portfolio Stock

#### Post (Buy shares)

#### 

1. Check to make sure the player has enough cash available for the purchase. If not, then result error.
2. Insert a record in StockTransactions
3. Update the PortfolioStocks table
   1. If a record already exists for the StockID and the PortfolioID, then update the Shares, LastPrice and Value.
   2. If a record does not exist for the StockID and the PortfolioID, then insert a new record.
4. Update the Value and Cash Available fields in the Portfolio table

#### Post (Sell Shares)

1. Insert a record in StockTransactions
2. Update the PortfolioStocks table
   1. If a record already exists for the StockID and the PortfolioID, then update the Shares, LastPrice and Value.
   2. If a record does not exist for the StockID and the PortfolioID, then insert a new record.
3. Update the Value and Cash Available fields in the Portfolio table

### API: Limit Orders

#### Post (Create Limit Order)

1. If a Buy Order, then make sure the Player has the available cash.
2. Insert a record in StockTransactions
3. Insert record into the LimitOrders table.
4. Update the Cash Available fields in the Portfolio table.

### API: Limit Order Execution

1. Insert a record in StockTransactions
2. Update the PortfolioStocks table
   1. If a record already exists for the StockID and the PortfolioID, then update the Shares, LastPrice and Value.
   2. If a record does not exist for the StockID and the PortfolioID and it’s a Buy Order, then insert a new record.
3. Update the LimitOrder table to close the : UpdatedDTTM, ExecutedPrice, LimitStatusID and StockTransactionID
4. If it’s a Sell Order, then update CashAvailable in the Portfolio table.

### API: Cancel Limit Order

1. Where can we store audit activity?
2. Update LimitStatusID to canceled and UpdatedDTTM to now in the LimitOrders table.
3. Update the CashAvailable in the related Portfolio table record if the TransactionTypeID for the limit order is a Buy.

### API: Forgot Password

1. Check if the email matches EmailAddress in the Players table.
2. If a match is found, check if the SecretKeyExpirationDTTM exists and has not passed. If true, then resend the email with the secret key. Quit.
3. If the SecretKeyExpirationDTTM has passed or is null, then update the SecretKey and SecretKeyExpirationDTTM fields in the Players table. Expiration should be set to 30 minutes.
4. Send an email with a link back to the [Password Reset Screen](#_vokf75sc06mf) with the SecretKey as a parameter variable in the URL.

### API: Reset Password

1. This API will need to be designed by the development firm.

### Service: Purge Old Records Routine

Frequency: Nightly

Each night we will want to purge historic records over X days old from the StockHistory table. We will need to find out how much data is delivered each day from the Firehose to determine how much data we can feasibly maintain in the database.

### Service: Update Total Players and Payouts

Frequency: Hourly

Update the TotalPlayers and TotalPayouts fields in the Global table.

* Total Payouts calculation: the sum of Pot from Games where game status is completed.
* Total Players calculation: the count of records in Players where PlayerStatusID is Active.

### Service: Purge Email and Password SecretKeys

Every 10 minutes a service should run to delete the PWDSecretKey, PWDSecretKeyExpirationDTTM, EmailSecretKey and EmailSecretKeyExpirationDTTM fields in the Players table where the PWDSecretKeyExpirationDTTM or EmailSecretKeyExpirationDTTM has passed.

### Common APIs

1. Get Navigation Data
2. Get Player Profile

|  |  |
| --- | --- |
| **API** | **Purpose** |
| GetNavigationData | Returns the Open Game count, Active Game count and the Notifications count. |
| GetTotalPayoutData | Returns the total payouts and players in the system. |
| GetOpenGames | Returns count and list of Open Games. |
|  |  |

### Verify Email Address

When a new player signs up, they will be sent a confirmation email with a link. When they click the link, it needs to update the Players table.

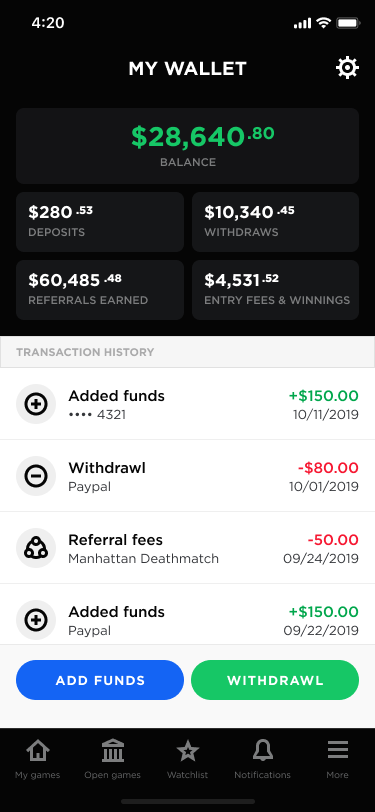
## 

## Screens

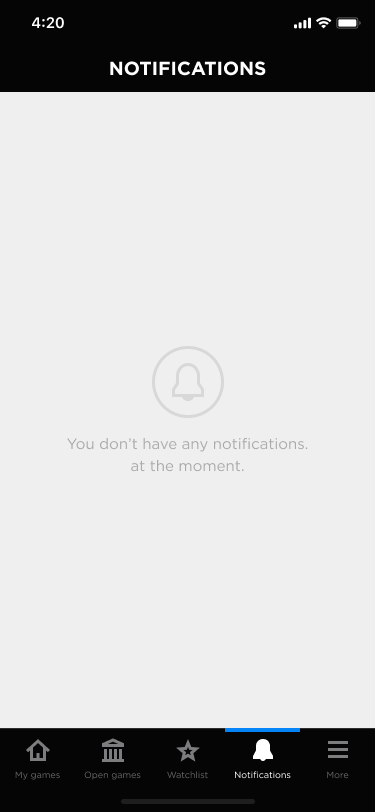
### Navigation

#### Bottom Navigation

This shows with no selection:



This shows with a selection:

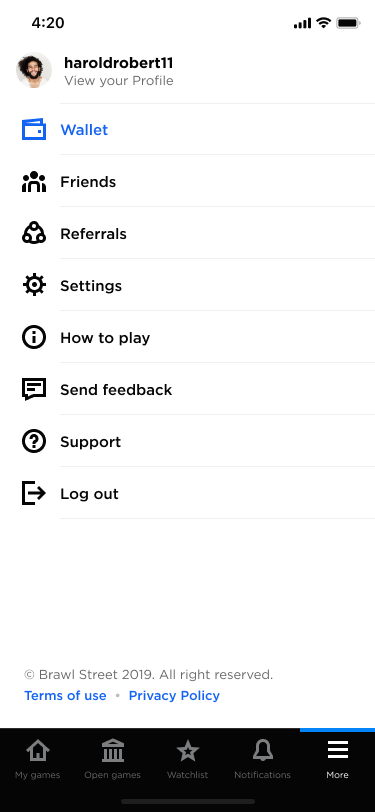


The bottom navigation will show the following:

* **My Games** - This opens the [Active Games List Screen](#_2q5kkql7kxz2).
* **Open Games** - This opens the [Open Games Screen](#_3h8mzm5vz65t).
* **Watchlist** - This opens the Watchlist Screen.
* **Notifications** - This opens the Notifications. The Notifications icon shows the number of open notifications for the player from the Notifications Table that have not been marked as viewed. This count must be updated at the same frequency as the footer navigation counts. This data can be retrieved using the NavigationCounts API and will be updated at the same time as the bottom navigation counts.

#### 

#### More Navigation (Hamburger)



|  |  |
| --- | --- |
| Menu Item | Links To / Action |
| Avatar + Player Username | No link |
| View Your Profile | Player Profile |
| ---------------------------------- |  |
| Wallet | Wallet |
| Friends | Friends |
| Referrals | Referrals |
| Settings | Settings |
| How to Play | XXX |
| Send Feedback | XXX |
| Support | XXX |
| Logout | Logout & return to Login Page |
| Terms of Use |  |
| Privacy Policy |  |

### 

### Preloader Screen

This screen will show when first loading the application or any particular screen.

### 

### Intro Screen

Login Button opens the Login screen.

Sign Up with Email Button opens the Sign Up Wizard.

[DISCUSS]: Should we combine the Intro Screen and the Login Screen?

[DEVELOPMENT]: We need to discuss how the auto-login typically works. Is it via a cookie and are there any settings we should have in the Players table?

### 

### Login Screen

Design

Please add a checkbox “Keep me logged in” to this screen.

Functionality

Forgot Password will link to [Forgot Password Screen](#_nlrgovueh6ut).

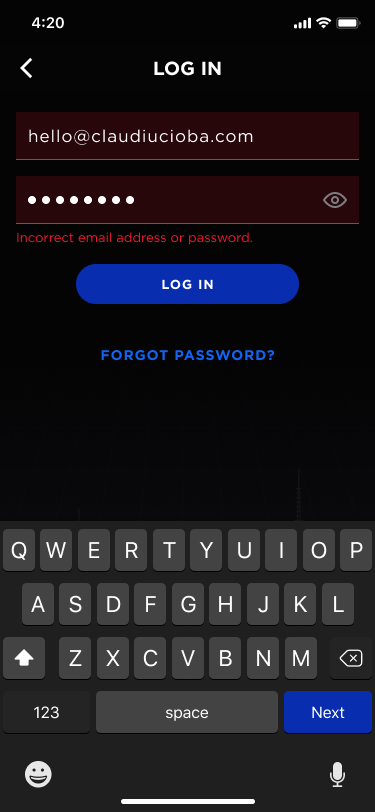
Keep me Logged in toggle will [TBD by Dev].

Players will login in using their email address and password. Clicking the Login button will call the [Login API](#_2mcmruiggu6l) and return one of the following:

* Login Success
* Failure: Too Many Attempts
* Failure: Incorrect Credentials

*Login Success:*

Go to the Dashboard XXX

*Failure: Incorrect Credentials*

*Failure: Too Many Login Attempts*

We need to have safeguards against someone attempting to guess login credentials. We will allow five [TBD] login attempts before locking the account and forcing them to reset their password.

If the login API returns a Failure: Too Many Attempts, then we need to:

1. Show the following text on the screen: “Sorry, but you have exceeded the allowed number of login attempts. Please click forgot password to reset your password.”
2. Disable the login button.

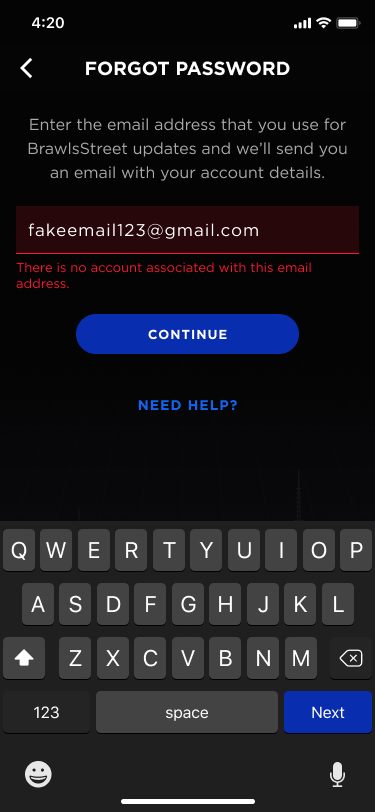
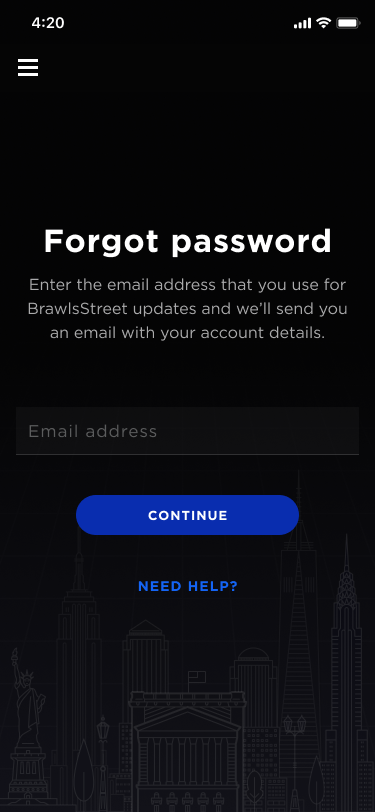
DEVELOPMENT: Is there a better way to handle preventing too many login attempts?

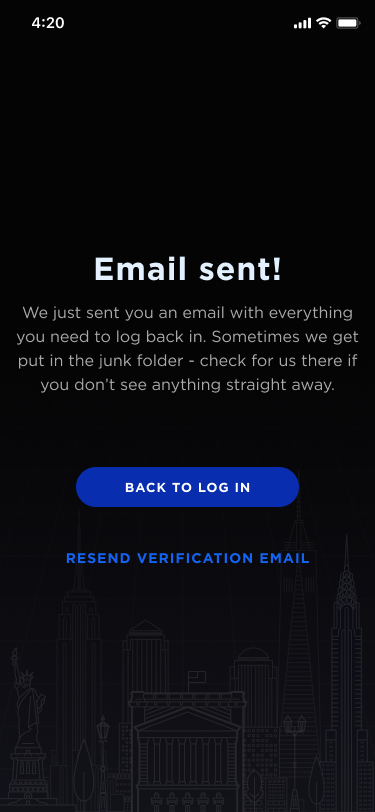
DEVELOPMENT: Need to discuss how autologin will work. Do we drop a cookie?

### 

### Forgot Password Screen

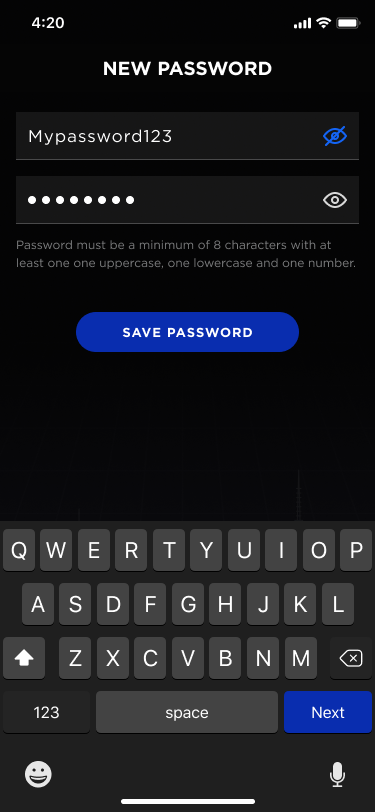
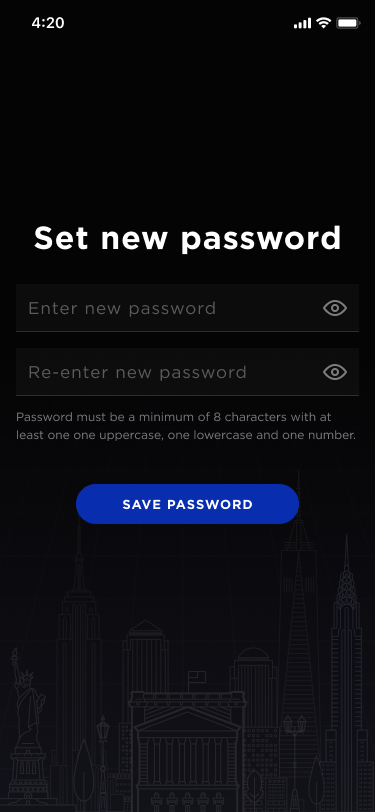
If the user clicks Forgot Password on the login page, they will be taken to this screen.

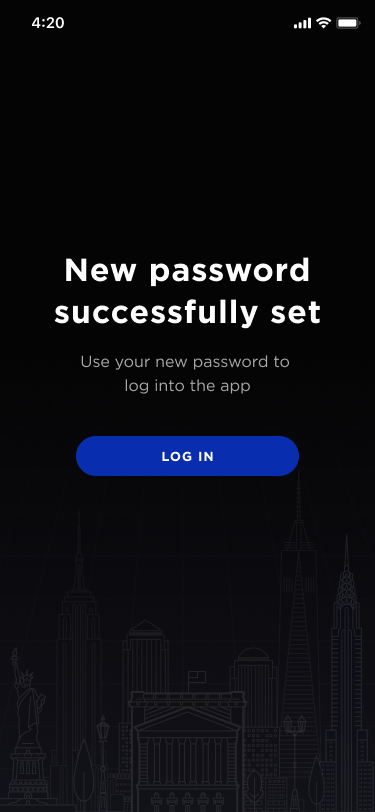




Please note that the Resend Email link will only be visible after the user has clicked the Send Reset Link button.

An email needs to be sent to the email address with a link that will then show the following:

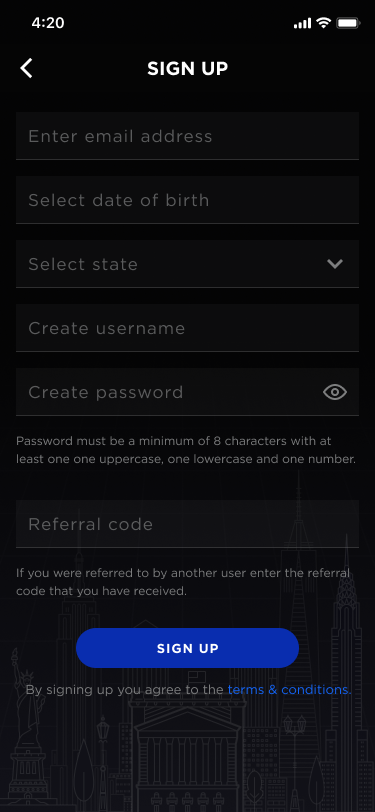




Make sure that the new password adheres to the same validation as when creating a new account.

### 

### Sign Up Screen

This screen will be used to sign up a new player.

Validation

1. Email
   1. Make sure it’s a valid email format on field exit event. If validation fails, show “Please enter a valid email address”.
   2. Make sure the email address does not already exist. If fails, then show the text “That email already exists. Send forgot password link?”
2. Birthday
   1. User must add a valid birthday.
   2. Birthday must be before today and after 1/1/1915.
   3. If failure because too young, then show “We’re really sorry, but you must be 18 to register.”
3. A state must be selected.
   1. If fails, then show text “You must choose a state.”
   2. If the user selects a state that is prohibited, then show the text “We are sorry, but state regulations prohibit users from playing Brawl Street in [State Name]. Click here for more information.”
4. Username
   1. Must not match any existing username. If fails, show “Username already taken”
   2. Must be 4 to 20 characters long with no spaces. If fails, highlight “Username must be 4 - 20 characters long with no spaces”
5. Password
   1. Password must contain
      1. Min 8 characters in length
      2. Min 1 lowercase
      3. Min 1 uppercase
      4. Min 1 number
6. Referral Code
   1. Make sure the referral code matches an existing referral code on field exit event. If validation fails, please show “That referral code does not exist”.

Functionality

The password fields will be masked and the show password on/off control will be set to off by default. If the user clicks on, then it should show the password.

Back arrow returns the user to the Intro Screen.

The SIgnup Button will:

1. Create the Player Account
2. Take the Player to the Onboarding Wizard

### 

### 

### 

### 

### Signup: Screen: Location

Design

* Need this screen

Validation

Functionality

This will enable the application to use the phone’s location services.

To Discuss with Development

What if the user does not allow this?

What kind of errors might we get?

What table do we use to show which locations are allowed or not allowed?

What happens if someone signs in from an unallowed location?

### 

### Onboarding Wizard

Design

We will have three screens that show a new player how to play the game. The following are the screen titles:

1. How to Play
2. Types of Games
3. Trading Stocks
4. Trading Stocks Final

These need to be graphical in nature and need to show a progress bar at the top of the screen.

We probably need a skip link on the screen.

Below is the text for each screen

**How to Play**

Welcome to the Brawl Street! You are joining a growing group of players who are winning money even when the stock market is doing poorly.

To play, you can join an existing game or create your own game.

You are given $200,000 of Brawl Street virtual dollars to buy stocks with which you build your portfolio.

Your goal is to make your best stock picks and if you outperform other players in your game you win cold hard cash. So even if the market is down or your stocks tank, you just need to finish with more virtual dollars than other players based on your games rules and you win!

**Types of Games**

Each game requires you to pay an entry fee with most of that added to the Pot. Winners of the game split the Pot based on the type of game.

**50/50**

The top 50% of the players split the Pot evenly. This is a massive 80%

**20/80**

The top 20% of the players split the Pot evenly

**Top 10**

The top 10 players split the Pot evenly.

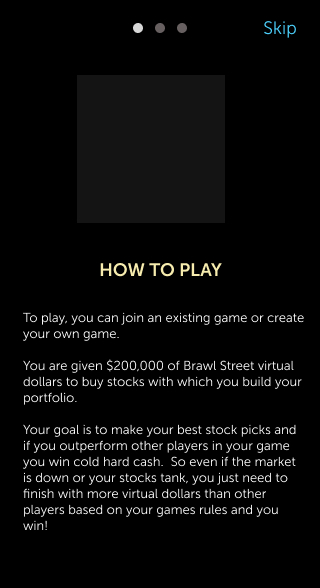
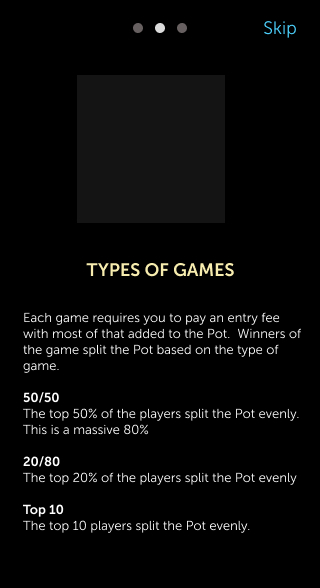
**Trading**

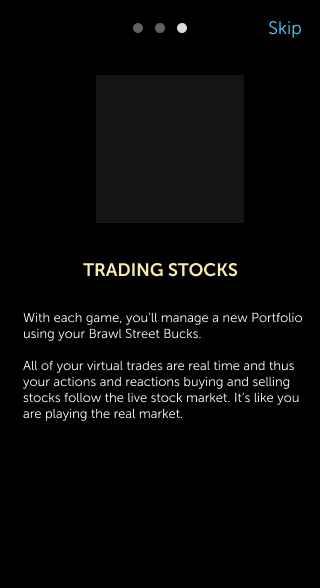
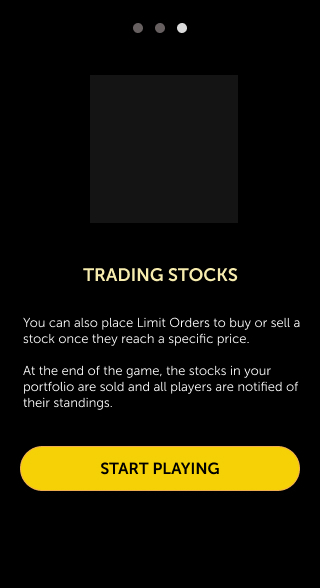
With each game, you’ll manage a new Portfolio using your Brawl Street Bucks.

All of your virtual trades are real time and thus your actions and reactions buying and selling stocks follow the live stock market. It’s like you are playing the real market.

You can also place Limit Orders to buy or sell a stock once they reach a specific price.

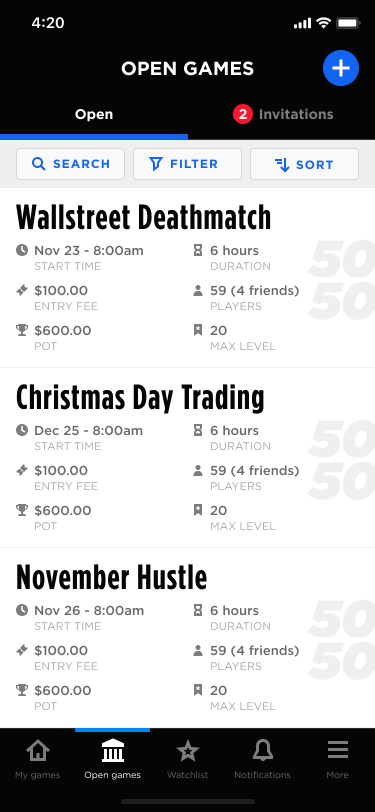
At the end of the game, the stocks in your portfolio are sold and all players are notified of their standings.





Start Playing button will take the player to the Open Games screen.

### Open Games Screen



This screen will show open games and invitations to games to which the player has been invited.

The tab control at the top toggles Open Games and Invitations. The number next to each in parenthesis is the count of games in either Open Games or Invited Games.

List Data

This should only show games that the logged in player is able to join based on their level. A player may join games of their level and higher. A player may not join a game where the max level is less than the player’s level.

Data for this screen can come from the Games API.

Filter

The filter icon will drop down a filter for the list of open games. See [Open Games: Filter](#_mf9hq9qu1cc6).

Sorting

To the right of the filter is the list sort control. See Open Games: Sorting.

The default will be Ascending Order of Start Date and Time.

Functionality

Each block represents an open game.

The list needs to be an infinite scroll.

Clicking on a game in the list will take the player to the [Join Game Detail Screen](#_r58yvm2k2yni) where they can view game details and confirm that they would like to join.

The player can also create a new game by clicking on the Create New Game (plus) button. The button links to the [Create Game Screen](#_vmcdh7c3aa1b).

### Open Games: Filter

The following shows how the filter for the Open Games tab works.

The player is able to update or clear out the filter.

The filters available are:

1. Game Type
2. Entry Fee (Range)
3. Duration (Range)
4. Maximum Player Level
5. Only Show Games with Friends (checkbox; not shown in image)

For the ranges, determine the max based on open the max found in the open games. Min and Max should be returned by the Games API.

Functionality

Upon clearing or adding a filter, call the Games API to requery and refresh the Open Games list.

### 

### 

### Open Games: Sorting

Functionality

Clicking on an item will toggle the up and down arrow to show sort order.

There will always be some sort order; so, there is not a state where there will not be a sort order.

### 

### 

### Open Games: Invitation Tab

This screen will show game to which the player has been invited by other players.

Functionality

The data from this screen will be provided by the Games API.

Notice that each row shows the avatar and username of who invited the player. Since the player may have been invited to the game by one or more players, this will be the user who last (most recently) invited the player to the game.

The user can click the X to decline the invitation at which time the record just gets removed from the screen.

[DISCUSS]: If the player clicks the X to hide the invitation, but is then invited to the same game by a different player, should the new invitation show?

### 

### Join Game Screen: Main Screen

The player navigate to this screen from the Open Games list (open or invited tabs).

Design

Start Date format should be:

Nov 23, 2019 8:00 AM EST

Functionality

The Pot is the total of the entry fees paid by players who have joined the game. Thus, it keeps increasing with each new player who joins the game.

The Join game button will:

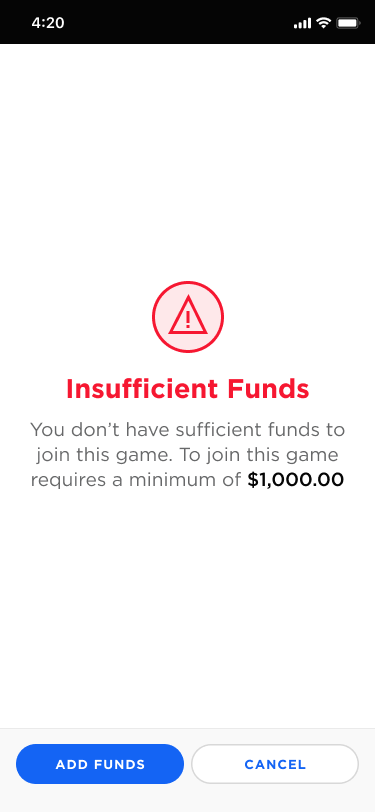
1. Verify that the game has not already started.
2. Verify that the player has enough funds to join the game.
3. Joint he player to the game.

Clicking the Join Game button will:

* Call the Portfolio API
* If success, then show a popup “Success! ” with a Start Trading button that goes to the Game Detail screen.
* If unsuccessful (due to start date passing) show “Sorry, but the game has already started.” with an OK button that leads to the Open Games list view.

Clicking back will take the user to the previous screen (open games).

The following image shows the error screen for when the player doesn’t have enough money to join the game.



### 

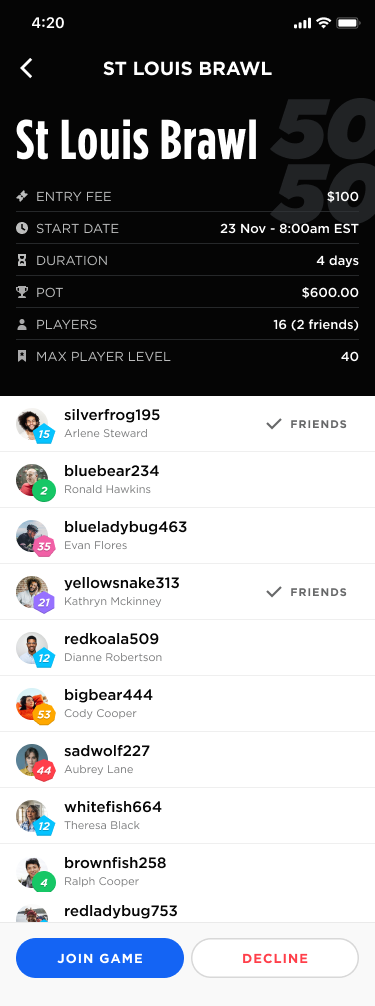
We need a similar screen to show if the game has already started. It should state:

The game has already started.

We’re sorry, but the game has already started. Players are not allowed to join a game that has already started.

Button: View Open Games

### Join Game Screen: Invited to Game



The following shows the detail screen for when the player has been invited to the game.

Notice that there are two buttons at the bottom of the screen: Join or Decline.

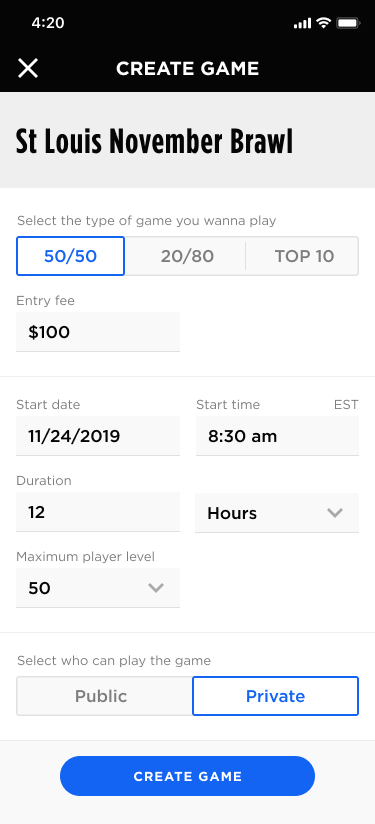
Join will run though the same validations and process as described above.

Decline will remove the game from the invitations section and will return the player to the Open Games: Invitations screen.

### 

### Create Game Screen

This is the screen that enables the player to create a new game.



Validation

* All fields are required.
* Game Name
  + Must not be used for an open / active game and not be the same as a game that has ended within the past 30 days.
* Start Date / Time
  + Should be at least 30 minutes into the future. This enables time for other players to join.
* Entry Fee
  + Must be greater or equal to $10; no decimal places
  + Must be equal to or less than the amount in the player’s account.
    - If they enter an amount that is greater than the account balance, then show a warning icon to the right of the entry fee field. We should also place some text right below the field with a link to add funds to their account.

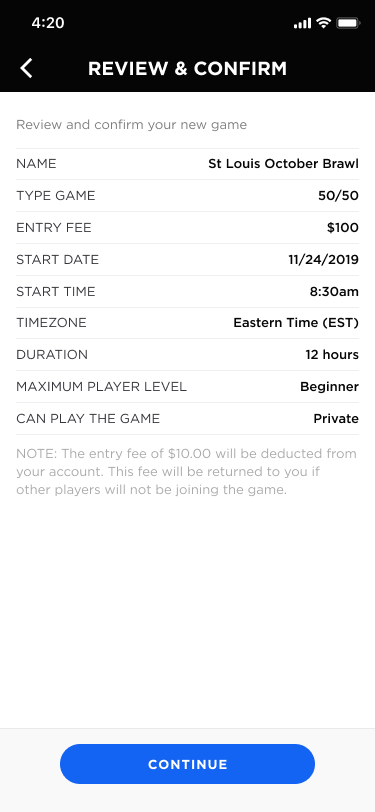
Functionality

The create game button should be disabled until all validation is cleared.

Create Game button will open the Confirmation screen.

### 

### Create Game Screen: Review & Confirmation



This screen is meant to confirm the game being created.

Functionality

The Continue button will run the API to create the game and then take the player to the Invite Friends.

### 

### Create Game Screen: Invite Friends to Game

This enables the player to invite friends to the game that the player has created or joined.

Design

If the player reached this page after creating a new game, then the back arrow should not show. Only the Skip link will show.

If the player reached this page from the Pending Game Detail Screen, then the screen will show the back arrow and not the Skip link.

Player List

If a player has already been invited to the game, then suppress that player from the list.

Do not show friends who would not be able to join the game due to their level.

Functionality

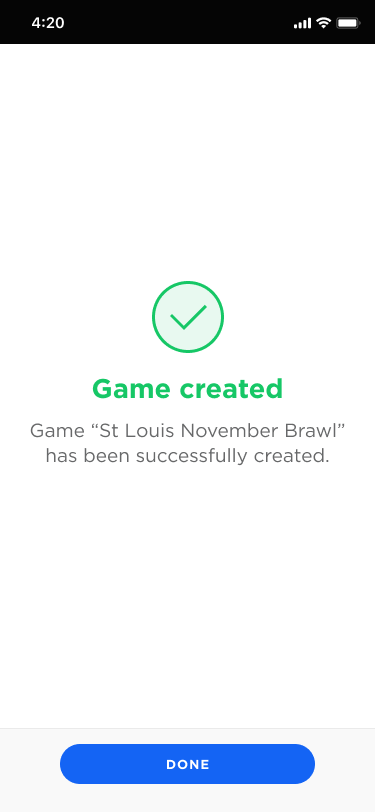
The Skip link will take the player to the Game Created screen (below).

The Invite Friends button should be disabled until the player has selected at least one friend.

Upon clicking the Invite X Friends button, the system will:

* Invite the friends
* If this screen is part of creating a new game, then it will show the Game Created screen.

### Create Game Screen: Game Created



This is the last screen in the sequence of creating a game.

If the player invited any friends, then it should show the text “Game “St. LouisNovember Brawl” has been successfully created and you have invited 8 friends.”

The Done button will take the player to the Detail view of the game in which they just created.

### 

### 

### My Games: Active List

The purpose of this screen is to show games that the player has joined and have either started or are going to start in the future.

The three tabs will show active games that I had joined and have started, upcoming games will show games I have joined but have not started and history will show completed games.

Functionality

If a game has less than 7 hours to end, then show the time left with a red highlight.

The player can click into the game to view the details. If they are looking at Active Games, then take the player to the [Active Game Detail Screen](#_auu3zvs8lph8) for the selected game.

If they are looking at Upcoming Games, then take the player to the Upcoming Game Detail Screen for the selected game.

If they are looking at History, then take them to the completed game screen.

### 

### 

### 

### 

### 

### 

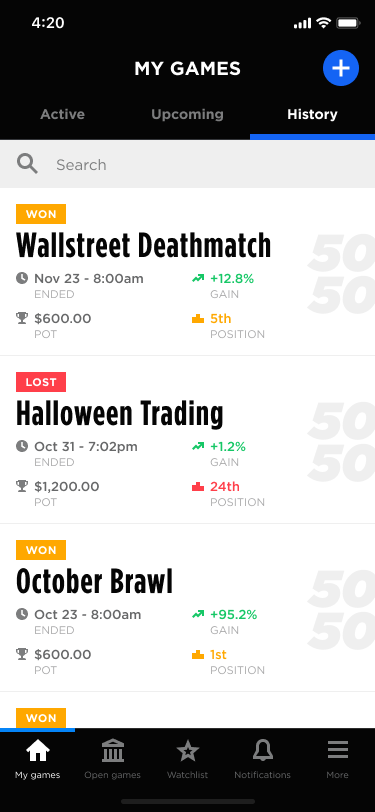
### My Games: Upcoming List

This screen will show games that the player has joined but have not started.

Clicking on a row will take the user to the Upcoming Game Detail View.

### 

### Game History List

The purpose of this screen is to show a list of games that have been completed.

Functionality

Data for this screen will come from the Games API.

Clicking into a game will take the player to the Completed Game Detail Screen.

### Upcoming Game Detail Screen

This screen will show the details of a game that the player has joined but which has not started.

Design

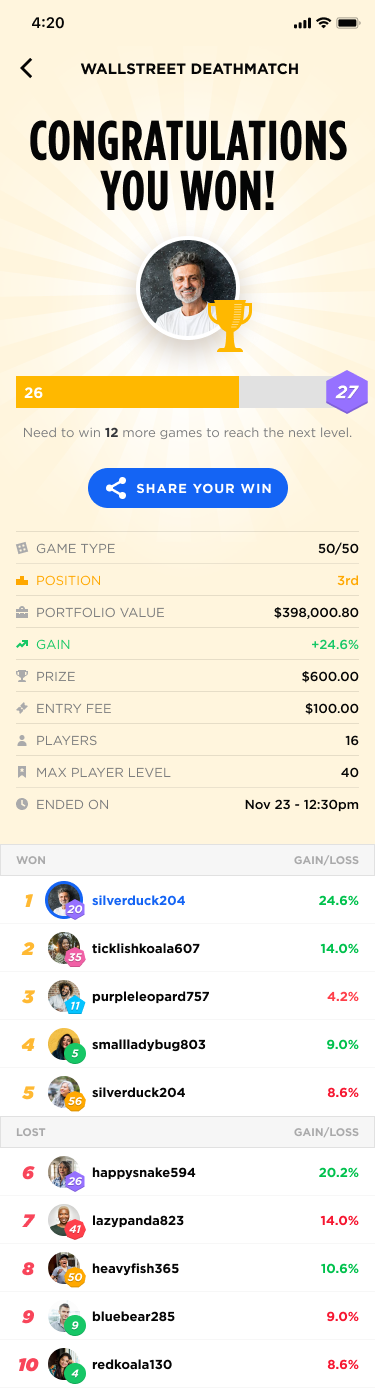
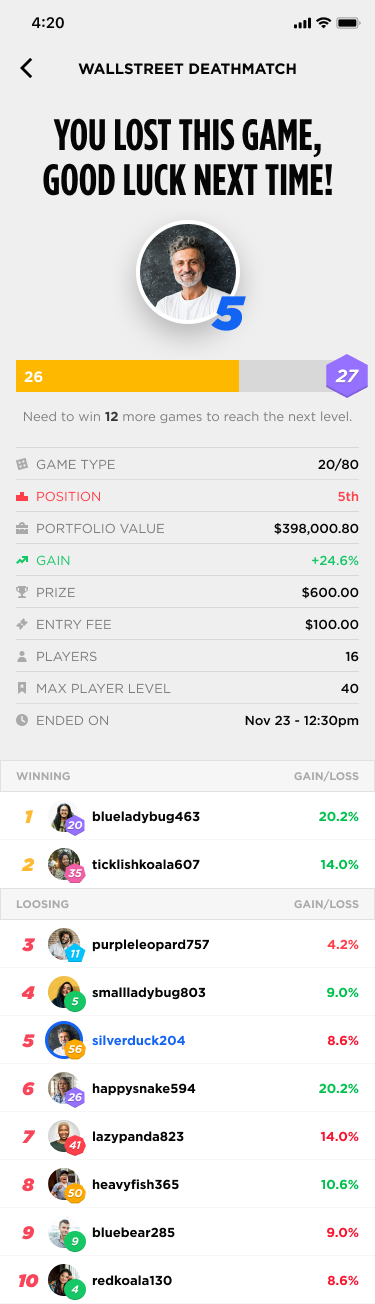
Ideally, we should show something on this screen like “Game starts in X hours”.

Functionality

The screen image we are showing to the right is of the open game detail view. The upcoming game screen will be the same, but the button needs to be “Invite Friends”. The invite friends button will open the invite friends screen, filtered for only friends who have not yet been invited to the game.

### Completed Game Detail Screen

There are two versions of the completed game detail screen: won and lost.



### 

### Active Game Detail: Overview Tab

This will show an active game to which the player has already joined and has started.

Functionality

Get detail data from the Game API.

### 

### Active Game Detail: Portfolio Tab

The Portfolio screen enables the player to see all the stocks they own in the portfolio and their limit orders for a portfolio.

Functionality

Available to invest is the amount of virtual funds that have not been invested or committed to invest via a limit purchase order.

Pending Limit Orders is the amount of Limit Orders to buy a stock.

Total Invested is the current value of the stocks purchased for this game.

Portfolio value is the sum of the virtual funcs available to invest plus the pending limit orders plus the current value of the investments.

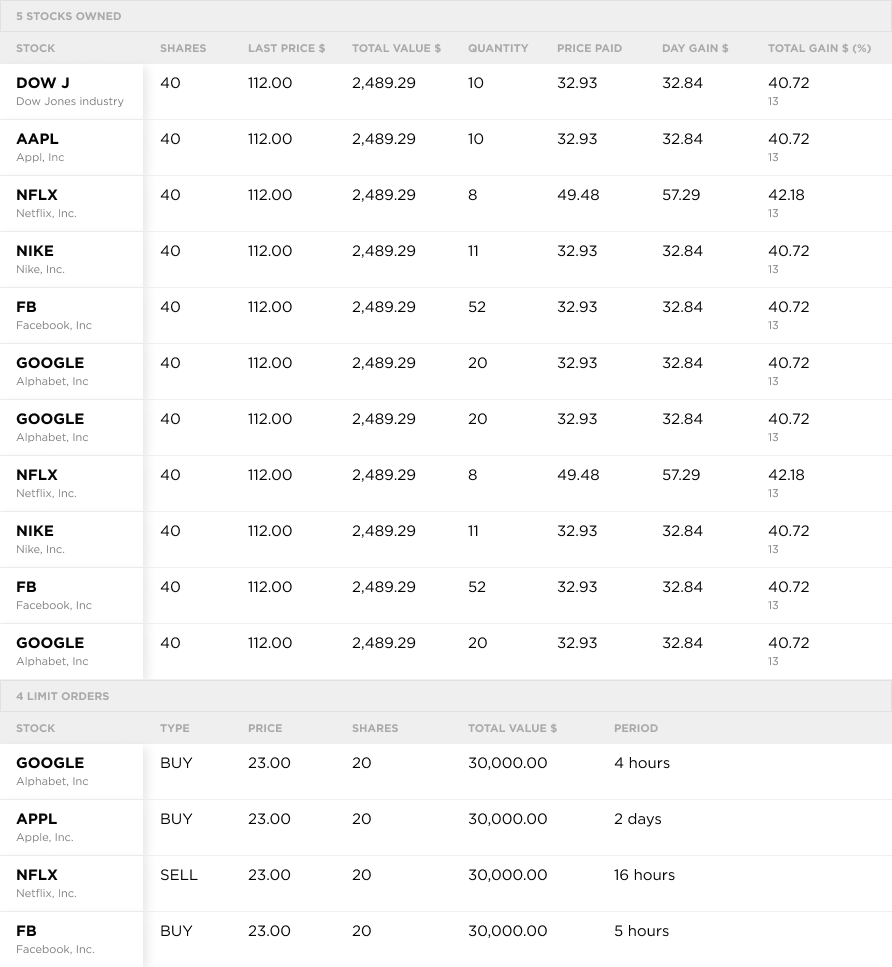
If the player clicks on a row, it should go to the Stock Detail screen.

If the player clicks on a limit order, then it should open the corresponding Edit Limit Order screen.

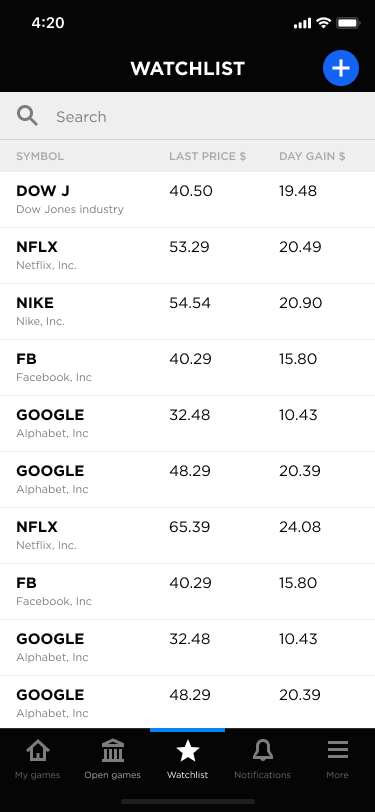
The Buy New Stock button should take the player to the Find New Stock to Buy screen.

It is important that is the player clicks on a stock or the Buy new stock button, that the system remembers which portfolio the player was on when clicking on the item.

The following is the layout of the fields in the portfolio tab:



### Select Stock to Buy



The purpose of this screen is to enable the user to either select a stock from their watchlist or find a different stock to buy.

Design

The screen to the right is the Watchlist screen.

The Select Stock to Buy screen is basically the same as the Watchlist screen, but:

* Change Watchlist to Select a Stock to Buy
* Remove the plus button.
* Do not show the Watchlist as selected in the navigation.

List

The list will show the watchlist stocks.

### Select Stock to Buy: Stock Search

If the user begins to search for a new stock, then it will show the following.

Design

The screen on the right is for adding a new stock to the watchlist.

For the Stock Search screen, remove the Add / Added buttons and replace them with a Buy button.

Change the title from Add to Watchlist to Find a Stock.

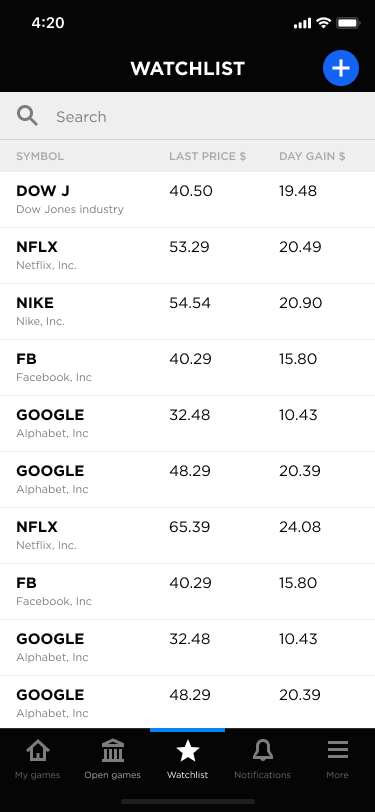
Clicking on the stock name will take the player to the Stock Detail Screen.

Clicking on the Buy button will take the player to the Buy Stock screen.

.

### 

### Watchlist



The purpose of this tab is to show a list of stocks the player has marked to watch. It also enables the player to quickly lookup a new stock.

The watch list is not portfolio specific. It’s a global watch list.

Functionality

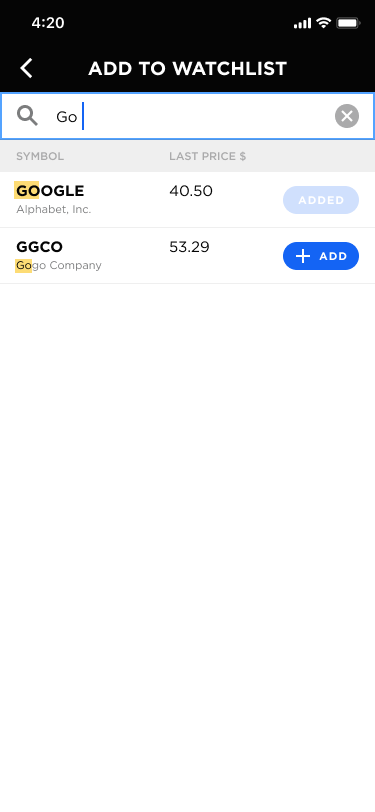
The Watchlist is based on an API call to get data from the Watchlist table.

### 

### Watchlist: Add Stock to List

Clicking on a stock will take the player to the stock detail page.

Clicking on the Add button will add the stock to the player’s watchlist and then will change the button from Add to Added.



### 

### Stock Detail Screen



The purpose of this screen is to show the player details about the stock and enable the player to buy, sell or add the stock to their watchlist.

Functionality

Data from this screen will come from Stock API.

The Buy button will link the player to the [Buy Shares screen](#_940re475o3p).

The Sell button will link the player to the [Sell Shares screen](#_vyneoxydgwq4).

The Add or Remove toggle (star) for the watch list will call the Watchlist API.

Items to Figure Out

We need to determine what data we will actually be getting from the IEX Feed and then modify the fields shown on this screen to match.

We need to determine how we are going to handle the historical graph. We will hold a limited amount of data on each stock based on what we choose for purging rules. So, we may not be able to show price history without making a special call to the IEX API.

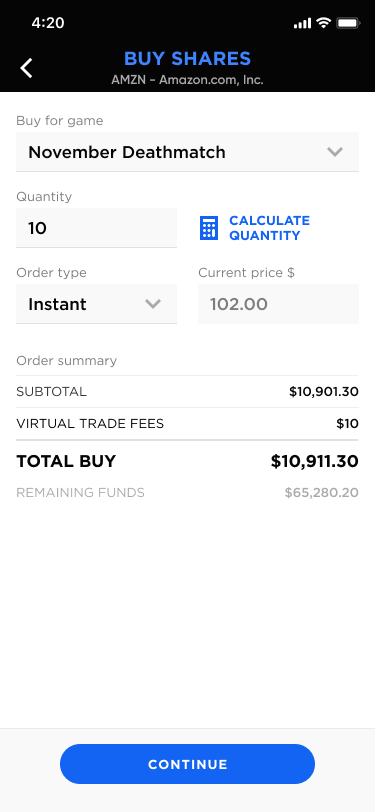


When a stock is added or removed from the watchlist, show the following at the top:

### 

### Buy Shares Screen: Instant

This screen enables the player to buy shares of stock for a game. They can purchase at Market Price or create a Limit Order.

Functionality

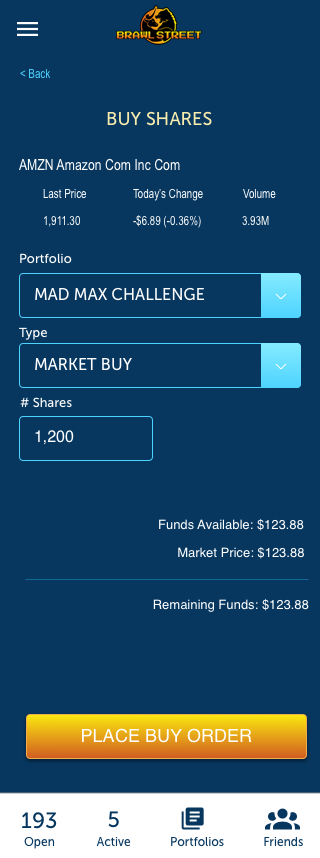
If the user has come from a specific game, then the Buy for Game field will be filled in.

If the user changes the quantity or the portfolio, then change the order summary values.

The Order Type field options are Market Buy and Limit Buy. If the player chooses Limit Buy then it needs to show fields for Limit Price and Duration. Also the text on the button will change to Place Limit Buy Order. See the Limit Order Screen.

The Continue button should be disabled until the player has selected a portfolio, entered the quantity and Remaining Funds is >=$0.

When the player clicks the Continue button, then it needs to show the confirmation screen.



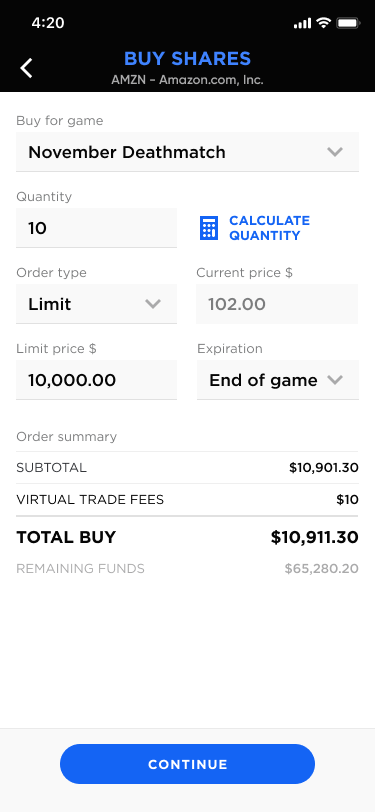
### Buy Shares Screen: Calculate Quantity

This shows the calculator when the player clicks on the calculate quantity button on the Buy Shares screen.

The enter quantity button needs to populate the quantity on the Buy Shares page.

### 

### Buy Shares Screen: Limit Order



The following shows the Buy Shares page when the Order Type is Limit.

The limit price should be less than the current price. Otherwise, the player would simply buy the stock at the current price.

The Continue button should be disabled until the player has selected a portfolio, entered the quantity, entered the limit price and Remaining Funds is >=$0.

When the player clicks the Continue button, then it needs to show the confirmation screen. It will not consummate the transaction until the next screen.

### 

### 

### 

### 

### 

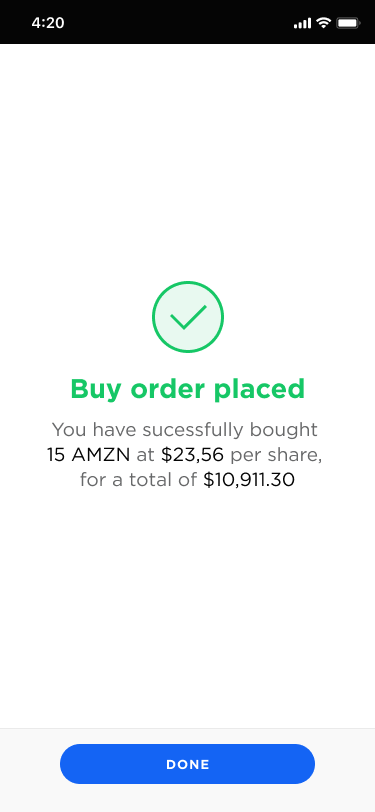
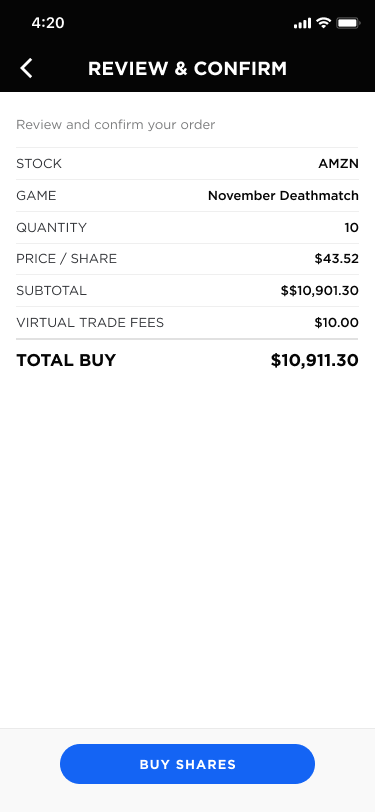
### Buy Shares Screen: Review and Confirmation Screens

After entering the information for a buy order, the following following confirmation screen will show. It needs to have the fresh stock price data. Remember that the stock price may have changed from the last screen. So, the data needs to be current.

The Buy Shares button is for an Instant Order. If the Order Type is a Limit Order then it needs to say “Place Limit Order”.

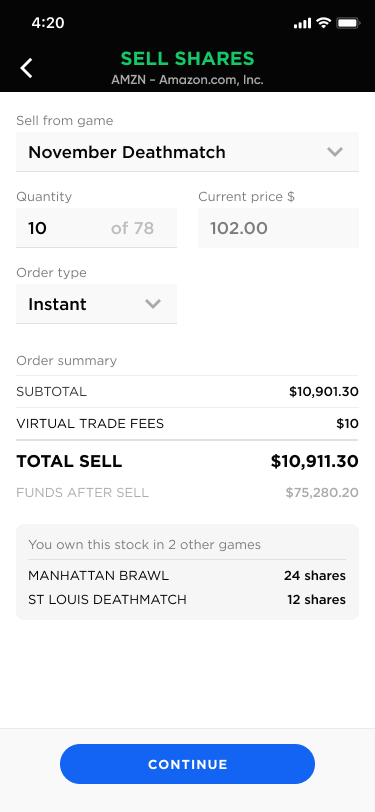
When the player clicks on the Buy Shares / Place Limit Order button, then the transaction must be consummated and the confirmation screen will show.

Done will take the player back to the game detail screen for which they entered an order.



### Sell Shares Screen: Instant Order

This screen enables the player to sell shares of stock for a game. They can sell at Market Price or create a Limit Order.

Functionality

If the user has come from a specific game, then the Buy for Game field will be filled in

On change of the Portfolio dropdown, update the Shares Owned. This data will come from Portfolio Stocks API.

The Type field options are Instant and Limit Order; default is Instant.

The screen to the right shows an Instant Order.

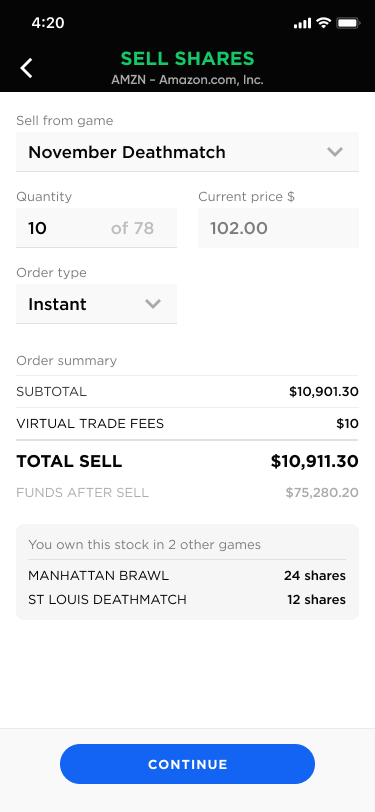
Button should be disabled until all fields are filled out and Quantity is <= Shares Owned.

When the player clicks the Continue button, then it needs to show the confirmation screen. It will not consummate the transaction until the next screen.

\*\* If the player sells shares of a stock for a game, and also has an open limit order to sell those shares, then limit order must be canceled. We will need to alert the player about this in the confirmation screen.

### 

### Sell Shares Screen: Limit Order



We do not have a design for this screen; however, it is the same as the Sell Shares Instant Order screen with the following modifications:

* Limit Price Field
* Expiration Field

The limit price must be greater than the current stock price. Otherwise, the player would just sell the stock in an Instant order.

Button should be disabled until all fields are filled out and Quantity is <= Shares Owned.

When the player clicks the Continue button, then it needs to show the confirmation screen. It will not consummate the transaction until the next screen.

### 

### Sell Shares Screen: Review and Confirmation Screens

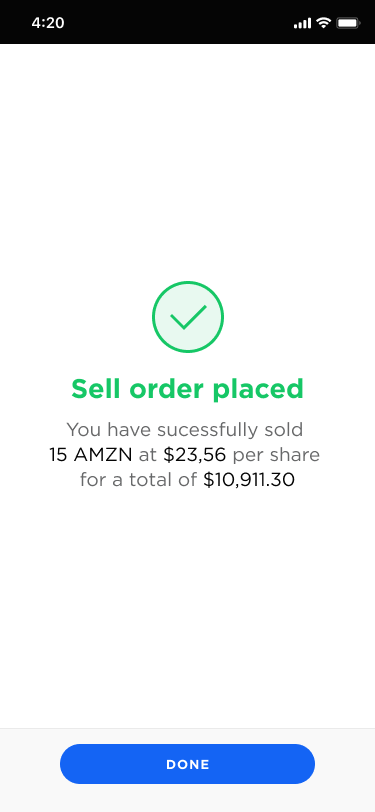
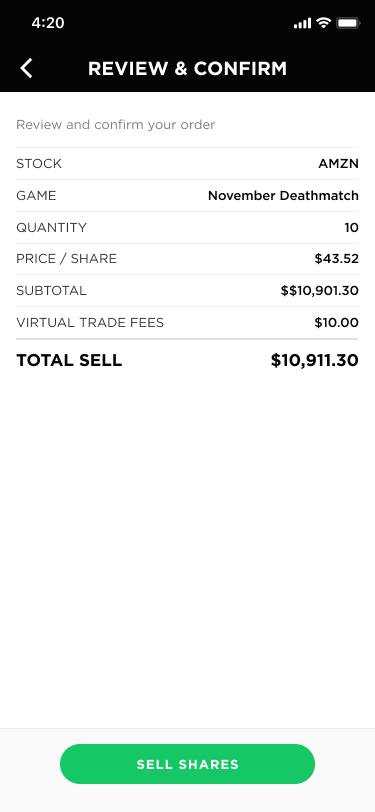
After entering the information for a sell order, the following following confirmation screen will show. It needs to have the fresh stock price data. Remember that the stock price may have changed from the last screen. So, the data needs to be current.

The Sell Shares button is for an Instant Order. If the Order Type is a Limit Order then it needs to say “Place Limit Order”.

When the player clicks on the Sell Shares / Place Limit Order button, then the transaction must be consummated and the confirmation screen will show.

The text on the confirmation screen will be different based on the order type (instant vs. limit).

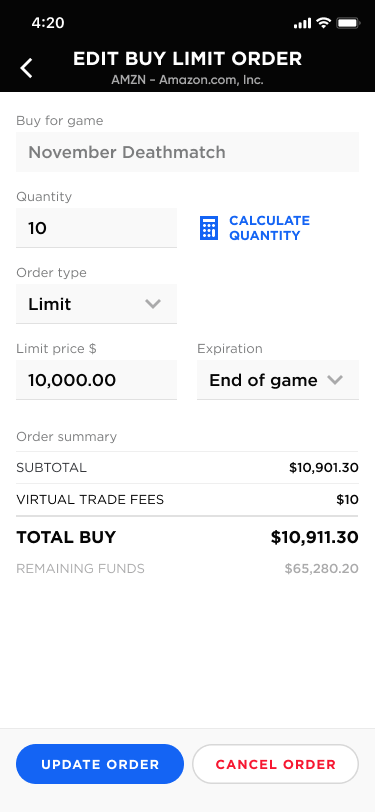
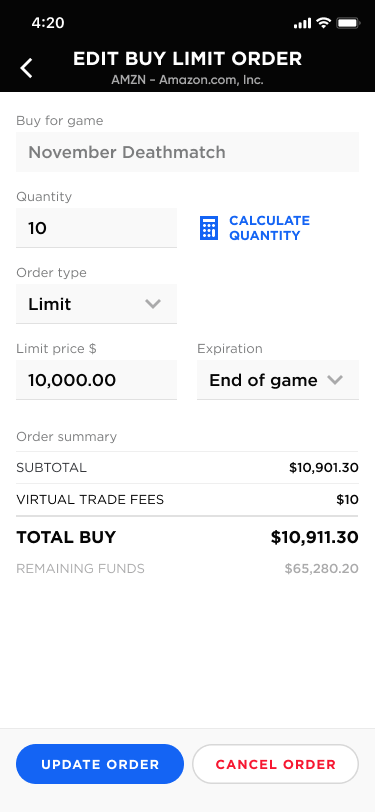
Done will take the player back to the game detail screen for which they entered an order.



### 

### Edit Limit Order

This screen enables the player to edit a limit order to buy shares of stock for a game. The following shows the screens for editing a buy order and a sell order:



Functionality

The Update Order button and the cancel buttons will call the Limit Order API.

If we are editing an existing buy order, the Remaining Funds needs to be adjusted for the order in which the player is editing. Validation on available funds needs to be made if it’s a buy order.

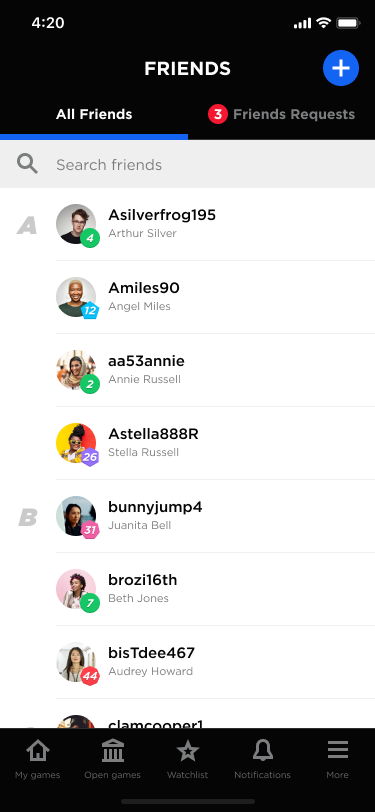
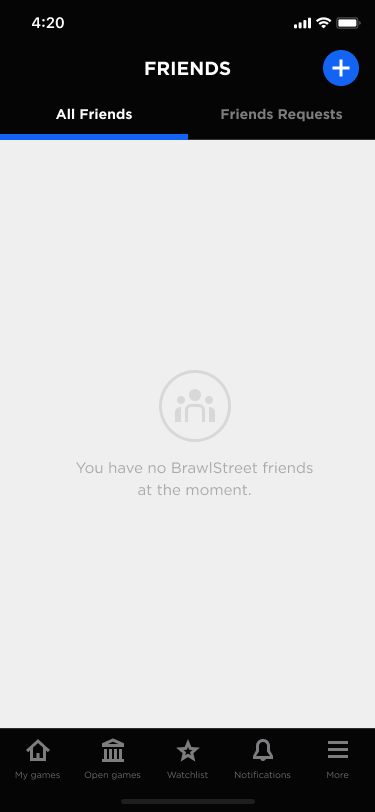
This screen needs to be followed by a Transaction Confirmation Screen (same as if the order was being placed for the first time.

### 

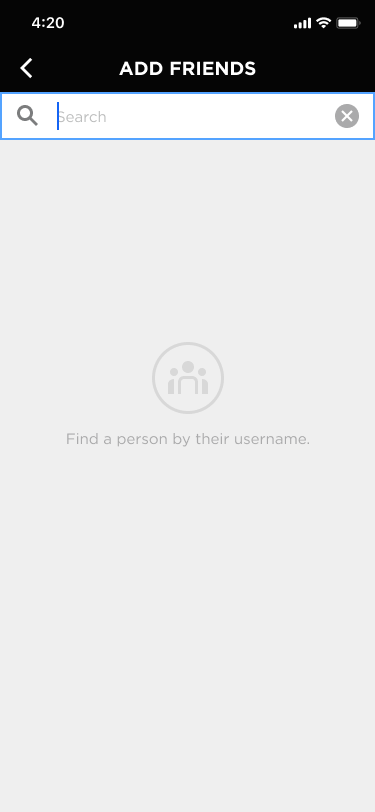
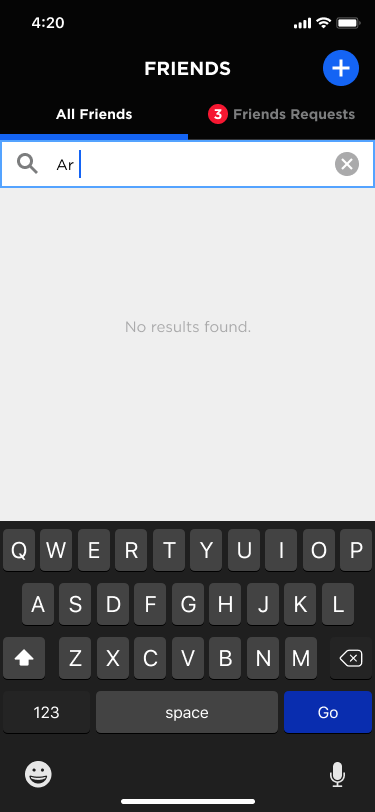
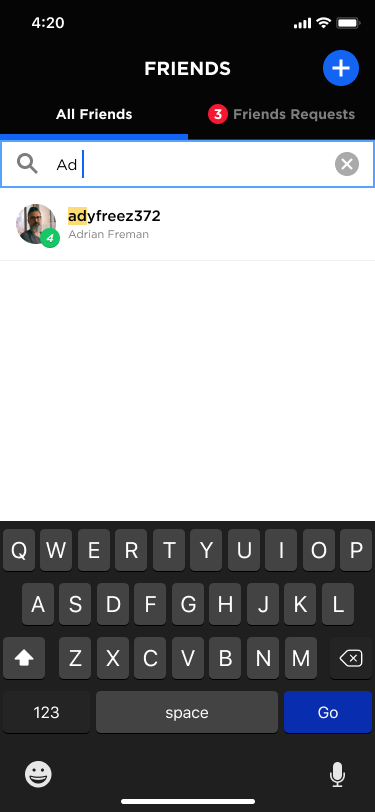
### Friend List Screen: All Friends Tab

The purpose of this screen is to show a list of the player’s friends and also a list of invitations the player has received from other players.

The following shows the friend list populated and empty:

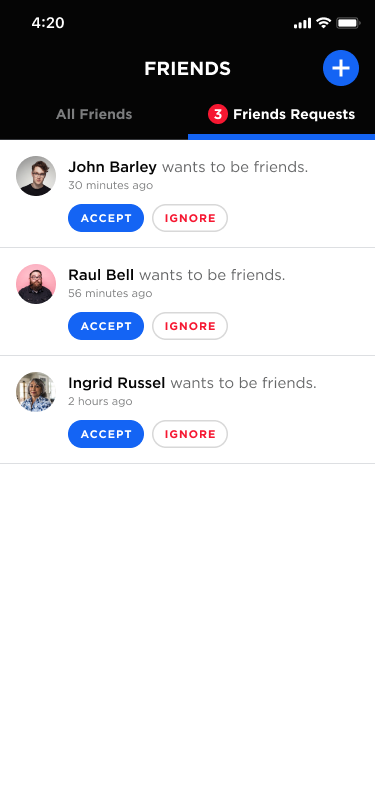
Clicking on the record in the list will show

The following shows how to add friends:



### 

### Friend List Screen: Friends Invitations Tab



The following shows a list of friend requests the player has received.

Clicking on the player name will open their profile.

Accept will add them as a friend and refresh the list.

Ignore will decline/remove the friend request.

### 

### Referrals: Main Screen

The purpose of this screen is to enable the player to send an affiliate link via email, social, sms, etc. This is different from adding a friend because the player might want to refer others to Brawl Street without becoming their friend.

Functionality

The copy button will copy the link to clipboard.

The Share Link button will call the phone’s share method with the following text:

“Check out Brawl Street: [URL]. I love this game!”

Refer a friend and gain an affiliate fee for every game they win (2%).

### 

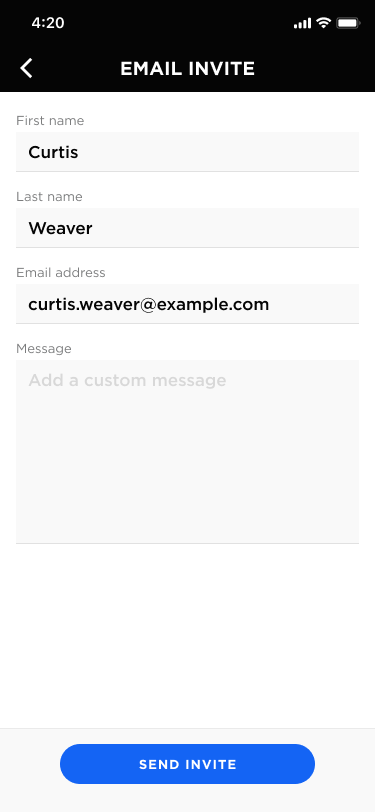
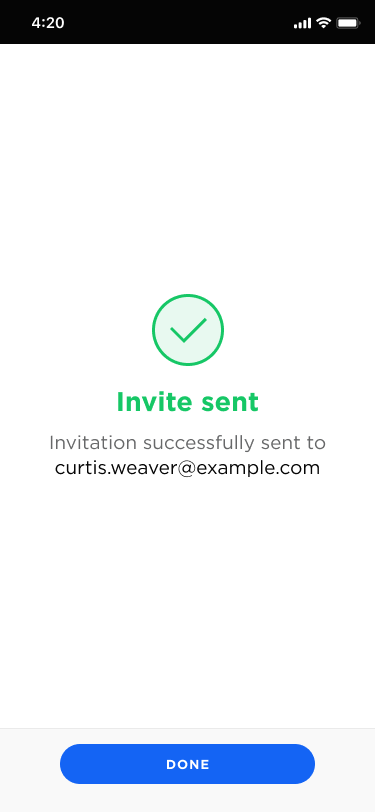
### 

### 

### Referrals: Invite via Email

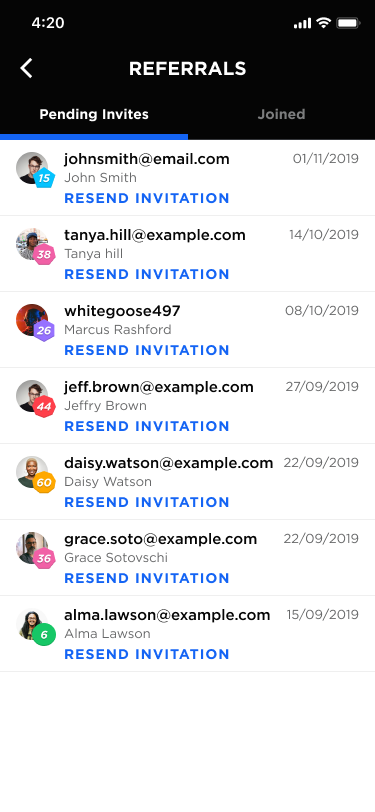
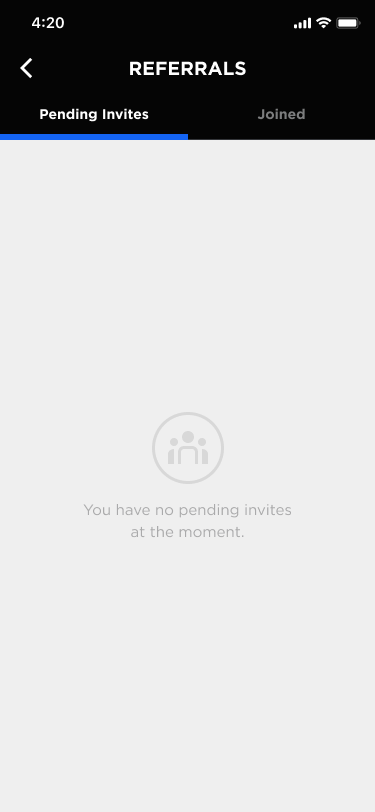
The player can choose to send an email invite. The email will need to be sent through the Brawl Street app.

XXX we need the text and html for the invite.



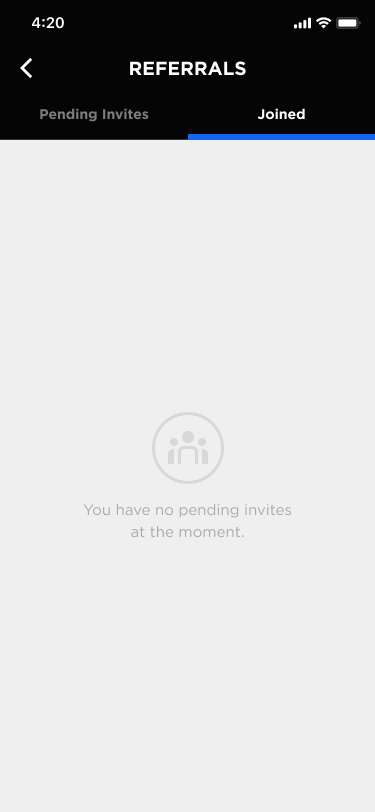
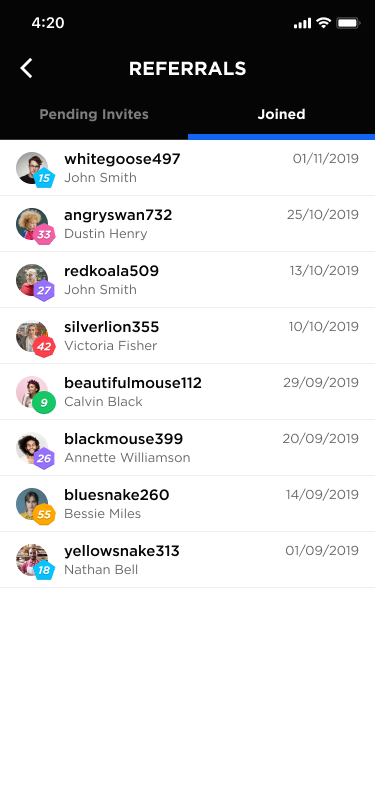
Referrals: Pending Invites

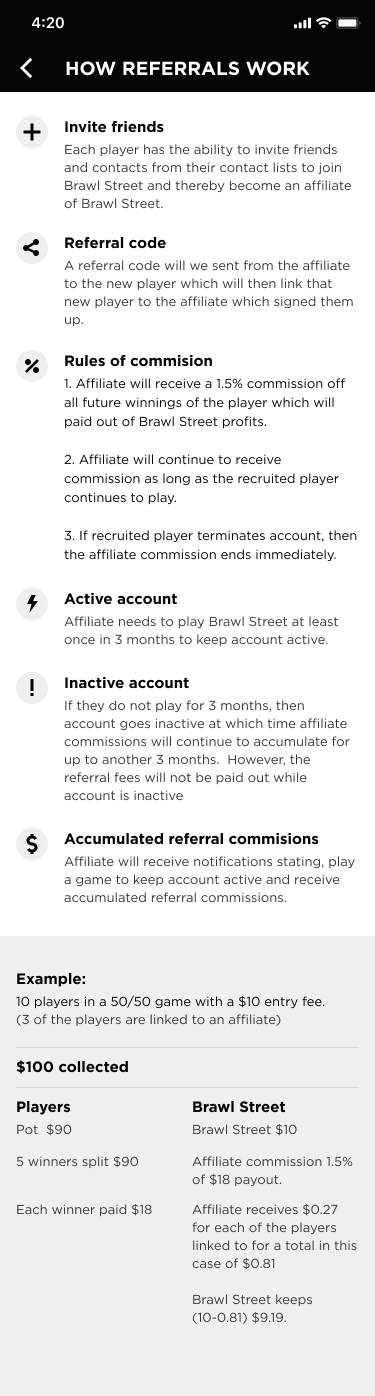
The Pending Invites will show a list of contacts the player has invited via email and who have not accepted the invitation.



Referrals: Joined

The following screen shows all players that have joined Brawl Street from the player’s referrals, no matter what the method of referral was.



Referrals: Information

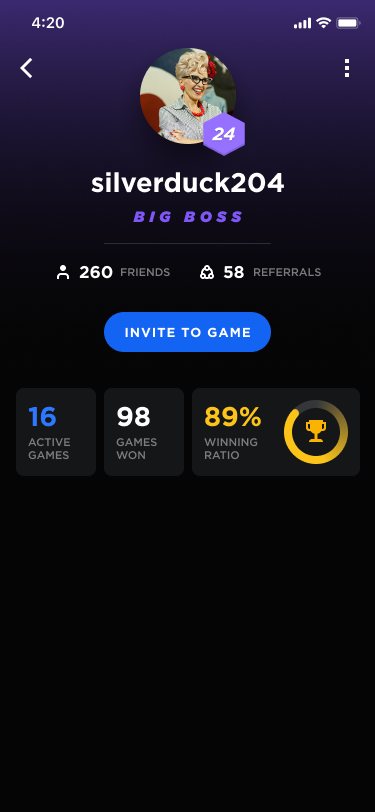
This screen will show when the player clicks on the info icon.

### My Profile

XXX Need to figure out how they get to this screen

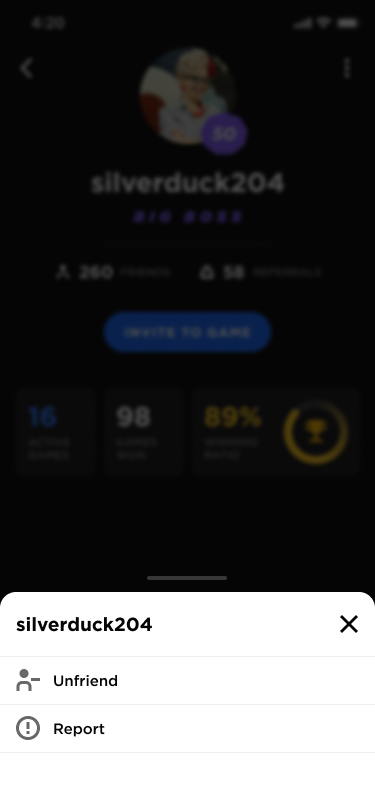


### Profile: Friend



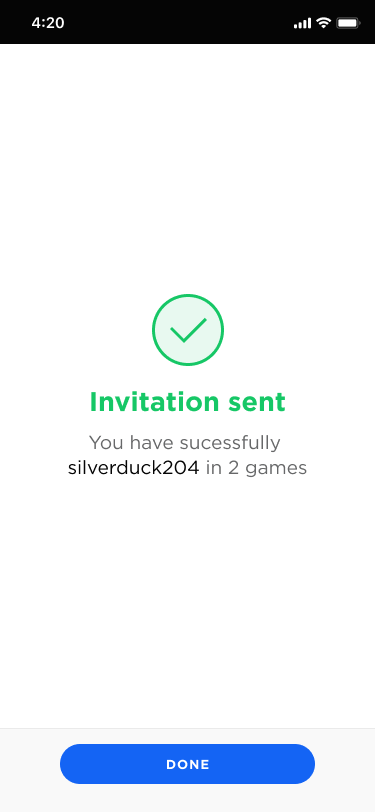
This screen will show when a player clicks on a user within the application that is a friend.

The upper right menu will show the Profile: Friend: Menu (see below).



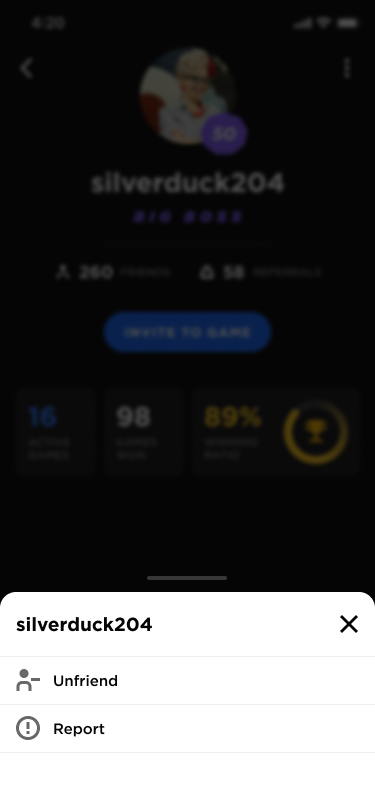
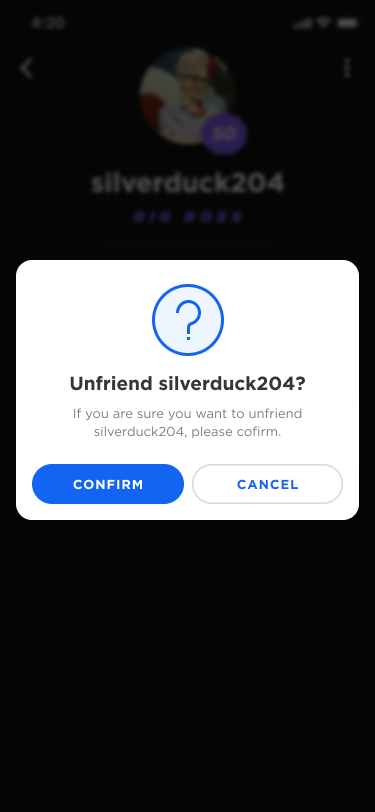
### Profile: Friend: Invite to Game

From the Profile: Friend screen, the player can invite the player to join an upcoming game. If the player clicks on the Invite to game button, then it will show upcoming games that have not started but which the player has joined.



### 

### Profile: Friend: Unfriending Process



### 

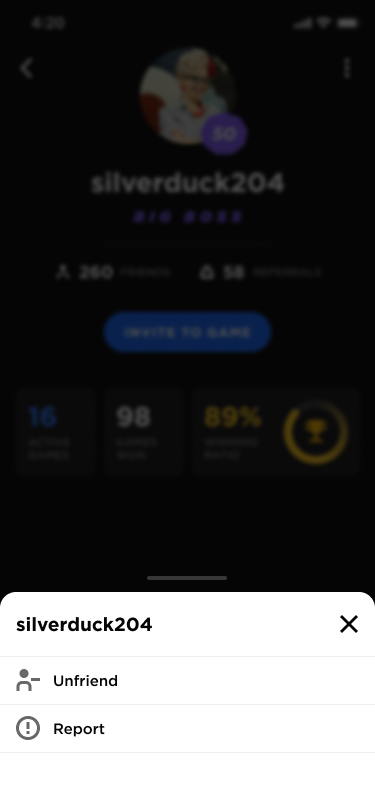
### Profile: Stranger

This screen will show when a player clicks on a user within the application that is NOT a friend.

The Add as Friend button will only show if the Player has not invited the stranger to be a friend.

If they have invited the stranger, then the button should show as Invited and should be disabled.

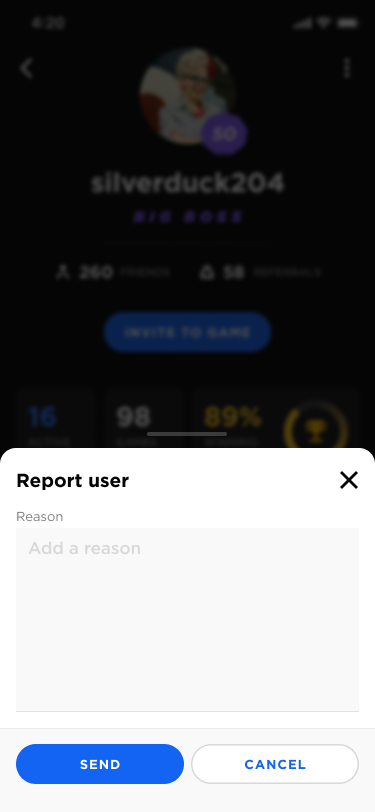
The upper right menu will show the menu below, but will only show the Report item. The Unfriend men item will not show since the person is a stranger.



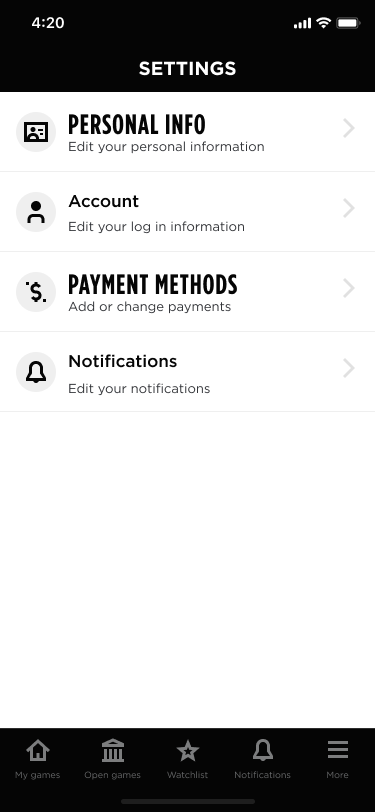
### 

### Profile: Report

A player can report any other player for abuse. If the player is reporting another player, then Brawl Street support needs to receive an alert via email of the report. The data needs to be stored in the database.



### Settings: Main Screen



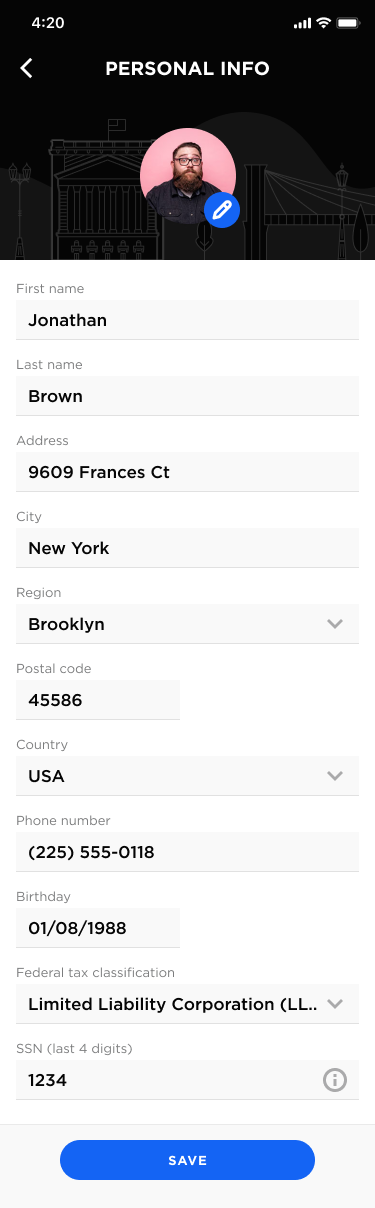
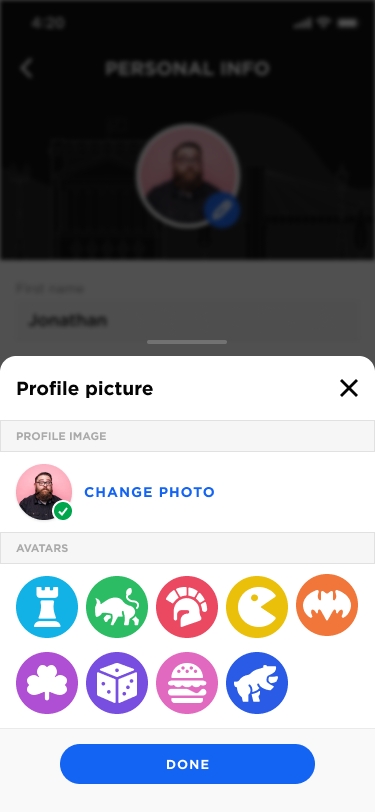
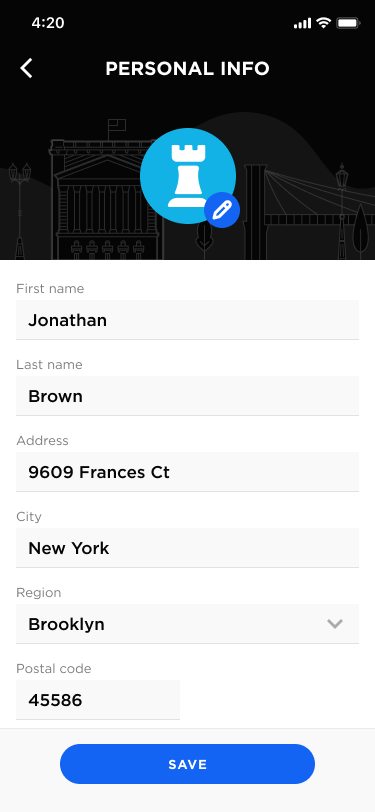
The following is the main screen for Settings. I believe the fonts are messed up in the image to the right so refer to Figma for the correct image.

We need a fifth section for Cancel Account. If they have funds available, then they need to be prompted that they must withdraw the funds first before cancelling the account. Otherwise, they need to go to a confirmation screen where they confirm that they want to cancel their account. Once they confirm it should go to a final screen confirming that their account was cancelled.

### Settings: Personal Info

The following shows the main screens for editing personal infomation. Some comments:

* We will not need Federal Tax Classification.
* If the user has already entered information from the signup screens, we may need to disable those fields and require that they call us to change that info. [FOR DISCUSSION].



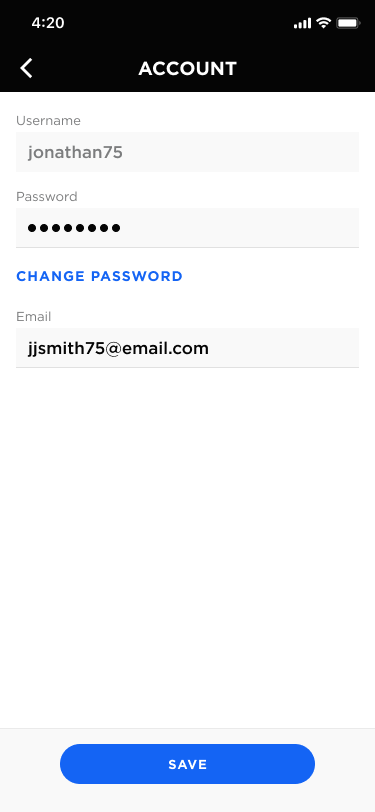
### 

### Settings: Account Info

The account info section enables the player to change their email and password.

The username and password fields will be disabled on this screen

If they change their email, then the new email address must be confirmed via email.

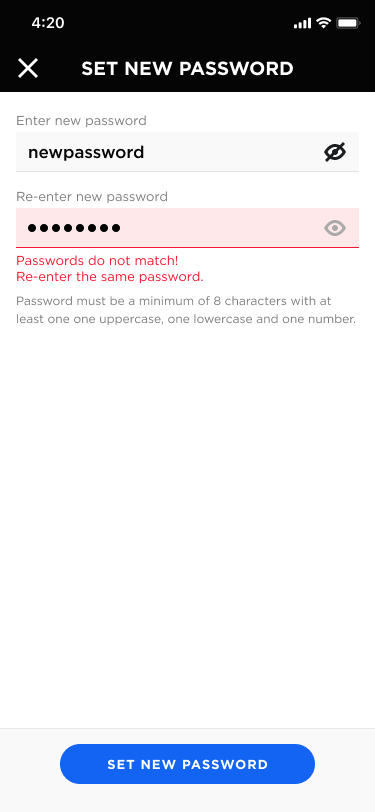
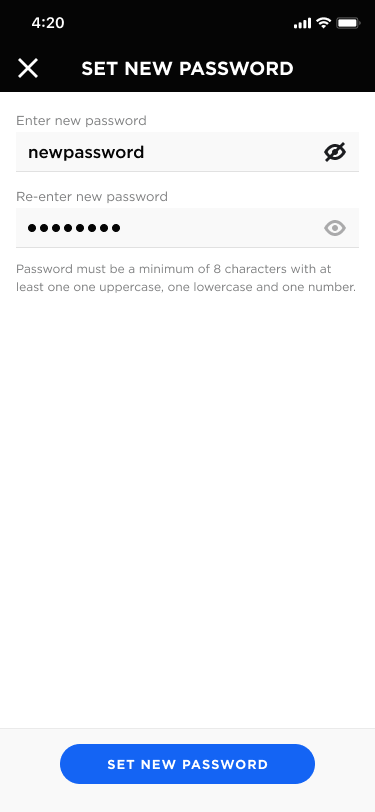
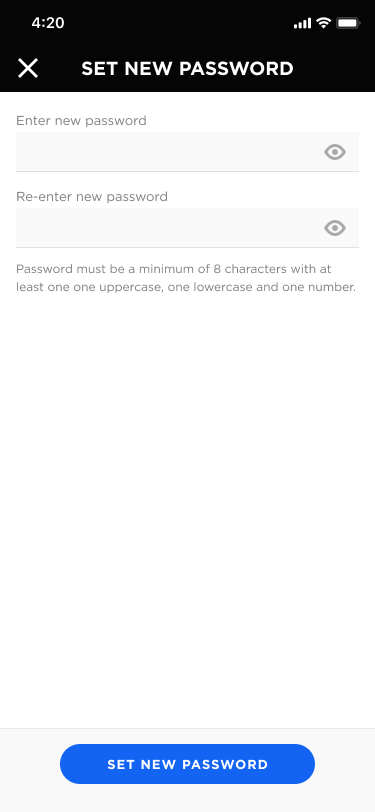


#### 

### Settings: Changing Password

The following screens show how the password is changed from within the app.

Please add a field above Enter New Password for Enter Existing Password. We need them to enter their existing password so we can confirm that it’s the player who is changing their password.



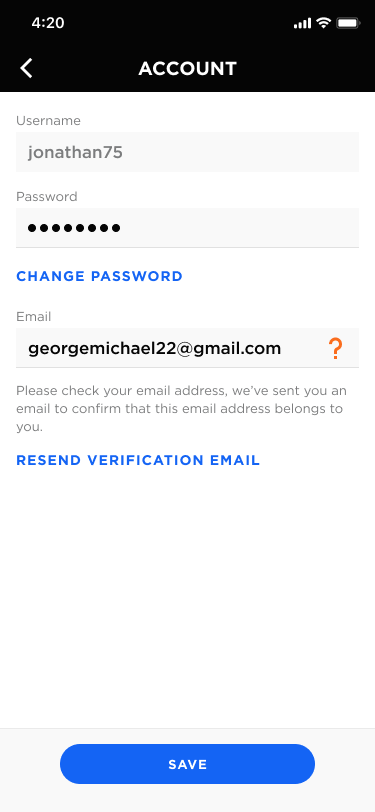
We need a confirmation screen after the password has been changed. Below is a screen. However, the user will not be logged out after they have changed their password from within the app. It just needs to say Password Updated and button should be “Return to Settings”.

#### 

#### 

### Settings: Changing Email Address

If the player changes their email address clicks save, then it will show the following message.



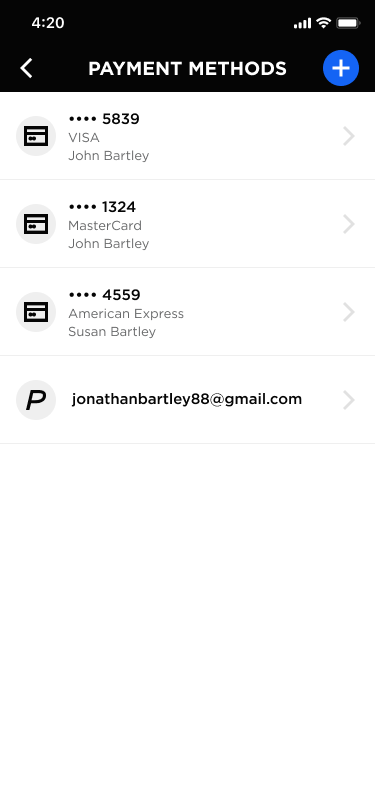
The player should receive an email with a link to verify the email address. Clicking on the link should show a confirmation screen that says “Email Updated”. I realize that this may be a regular webpage instead of the app - they may click the link on their desktop computer. Thus, there is no need for a button at the bottom.

#### 

#### 

### Settings: Payment Methods

This screen enables the player to view, addd, edit and delete existing payment methods. We need to have a screen that shows how it looks without any payment methods.



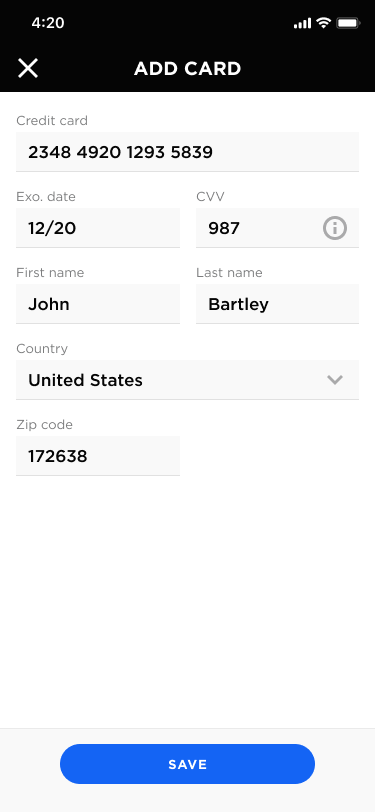
#### Adding a Payment Method

Choose type. If they already have a PayPal account connected, then disable the PayPal option. We only allow one PayPal account to be connected for a Player.



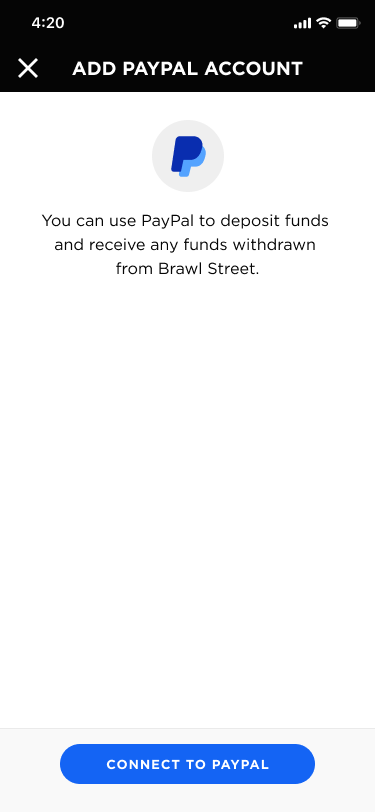
Adding a Credit Card

The fields on this screen may change based on the requirements of the credit card processor. For instance, we may need credit card type and instead of first and last name we may need full name.



Adding PayPal

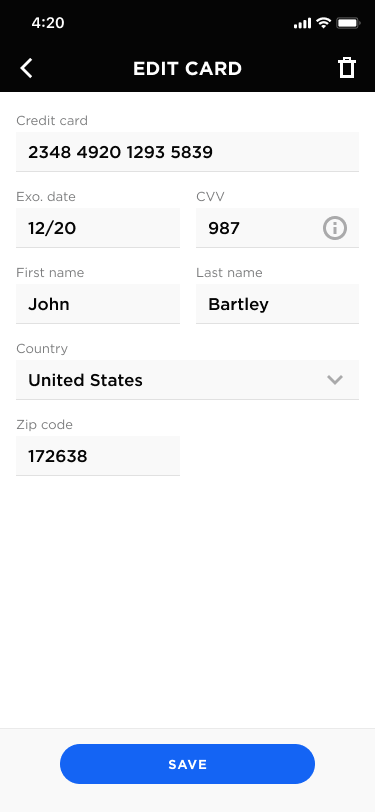
This button would trigger the PayPal connection



After connecting to PayPal, route the use back to the Payment Methods screen.

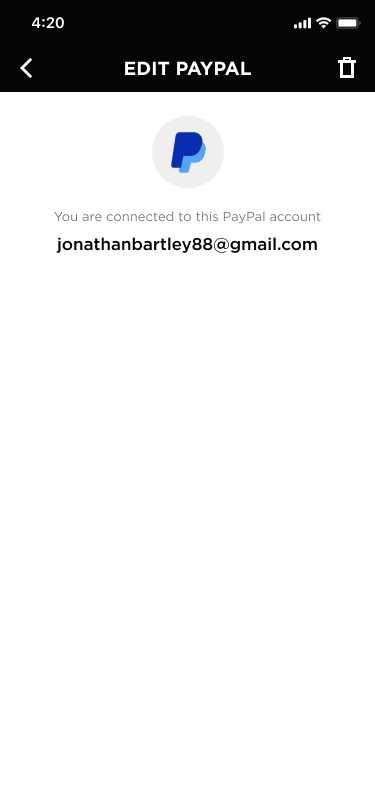
Editing a Card

If they choose to edit a card, show the following. As stated above, we may need to change the fields based on what credit card processor we use. Notice that the player can delete the card from here.



Editing PayPal

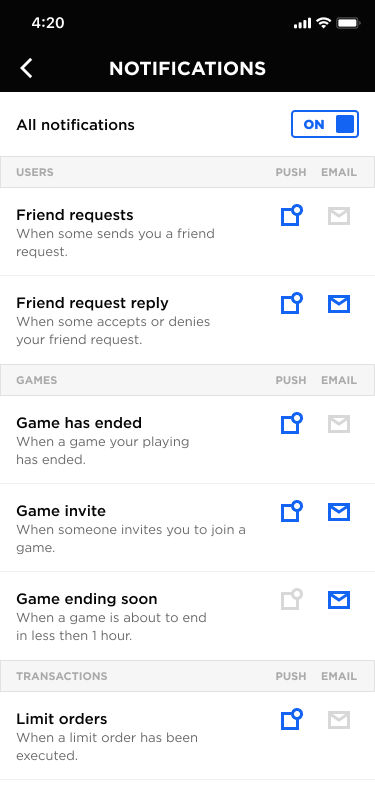
The player does not really have the ability to edit the PayPal connection. But they can delete it.



### 

### Settings: Notifications

This enables the player to set how they want to be notified. Each of the icons are toggles. For instance, a player could choose to be notified via both email and push for Friend Requests.



This screen will enable a player to set what types of notifications they want to receive.

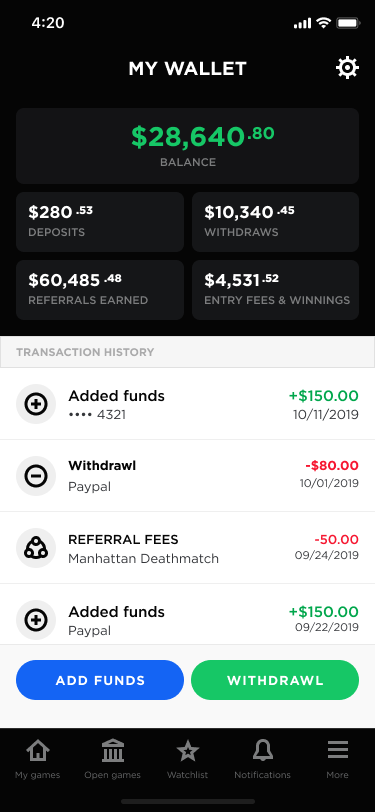
Design

This screen needs to be designed. For each notification type, the user can mark what method they will receive the notification.

|  |  |  |
| --- | --- | --- |
| **Notification** | **Mobile Push** | **Email** |
| [Full Name] invited you to be a friend. | X | X |
| [Full Name] invited you to join [Game Name]. | X | X |
| [Game Name] is ending within an hour | X | X |
| [Game Name] ended and you [Won or Lost]. You won [Amount] and it had been added to your account. ← only include second sentence if they won. | X | X |
| Your limit order for [Buying/Selling] [# Shares] of [Stock Name] at [Share Price] executed on [Date/Time]. | X | X |
| [Friend Full Name] won a game and you received a referral fee of [Amount]. | X | X |
| [Friend Full Name] accepted your invitation to be a friend. | X | X |
| [Friend Full Name] accepted your invitation and joined [Game Name] |  |  |

### 

### Wallet

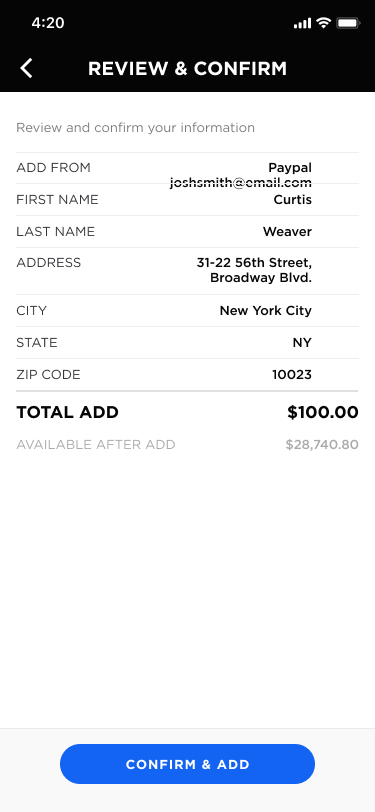
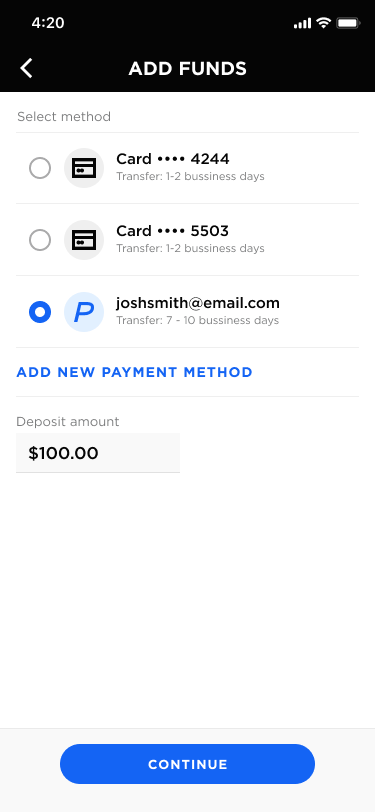
This screen will enable a player to:

1. View a snapshot of funds in the player’s wallet
2. Add funds
3. Withdraw funds
4. Referral Fees
5. Entry Fees & Winnings
6. View past transactions

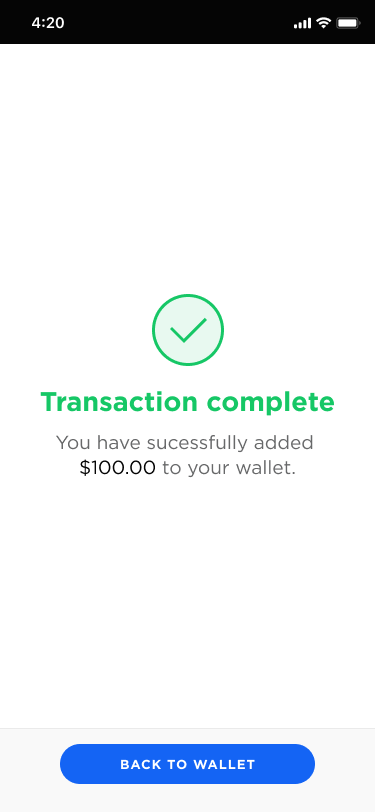
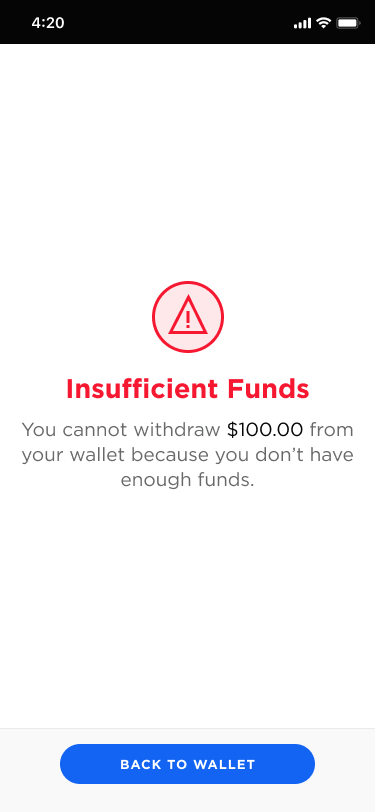
### 

### Wallet: Add Funds

The player can add funds via PayPal or a credit card. They can store multiple credit card accounts but only one PayPal account.



Once they click the confirm and add, we will process the transaction which will either end in a success or a failure. The text on the screens below is not correct, but the design is.



We would like the ability to add a new payment method on the fly. If the player clicks to add a new payment method, then show the following:

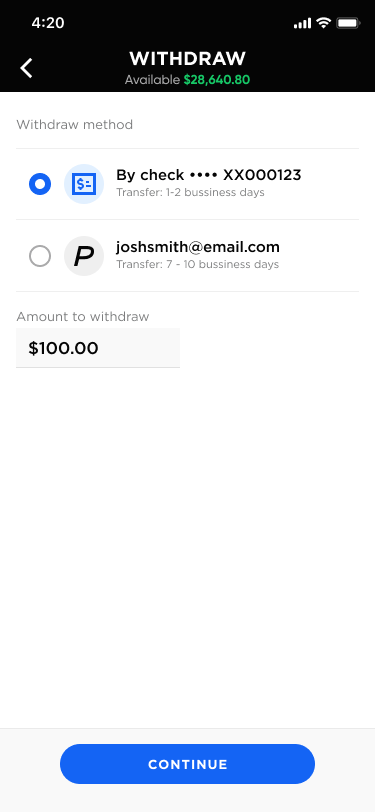


The player can add a new credit card or PayPal account (new PayPal only if they don’t already have one). After they have entered the new card, then it needs to return the player back to the Add Funds screen.

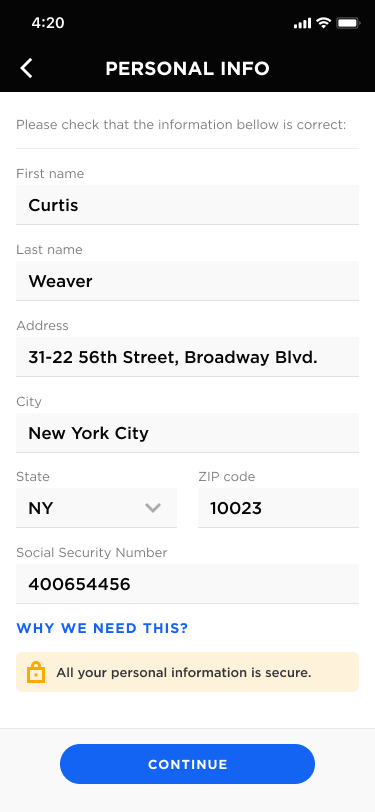
### Wallet: Withdraw Funds Screen

The first step in withdrawing funds is to select the payment method. There are only two methods we can use for paying money: by check or via PayPal.

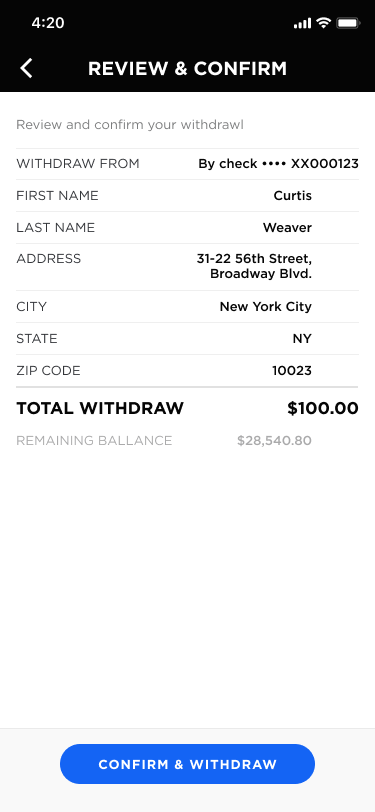
[DISCUSS] If we are sending funds via PayPal, do we do it via their connected PayPal account or do we use PayPal to send to thier registered email address?



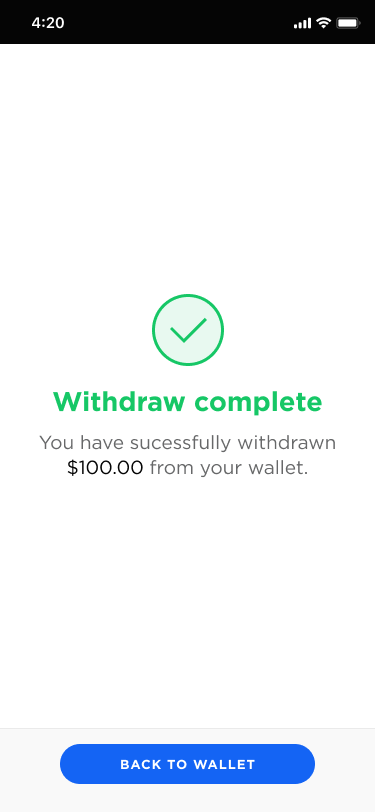
If the player would like to withdraw funds, we need to ensure that we have accurate info on them. Therefore, the first screen will show the data that we currently have for the player. The player will need to fill out any missing information. This information will be saved in their profile. For any fields that the player has already filled in, we should not allow them to change those fields. This is really to enable them to add missing information.



Next, we need to confirm the withdraw:

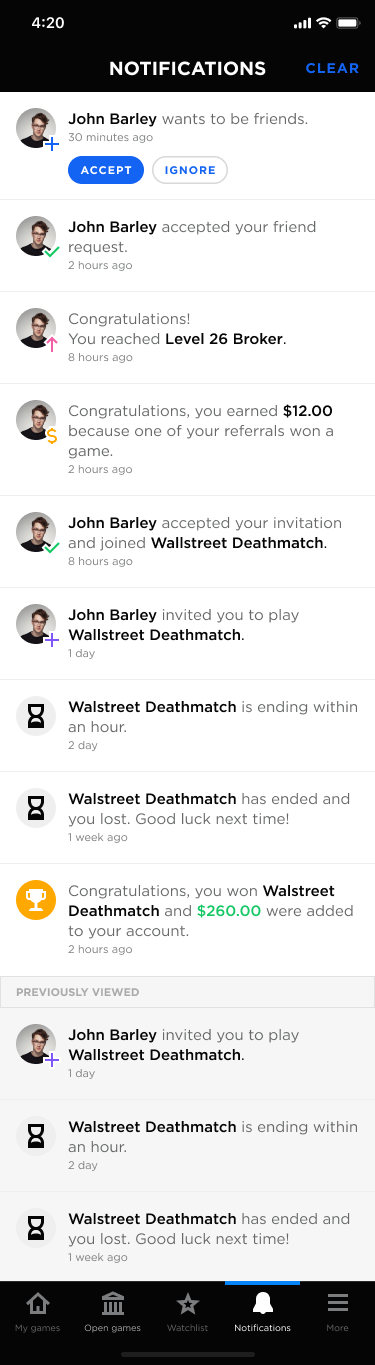
Please note that the text WITHDRAW FROM is not accurate. It should be WITHDRAW VIA.

Once they click the Confirm and Withdraw button, it needs to take the player to a final screen. We will need to change the text on this a bit.



### 

### Notifications Screen

In the footer is a bell icon with a number. The number will show how many open notifications exist. The player can click on this to reach the Notifications screen.

Functionality

Clear will clear all notifications.

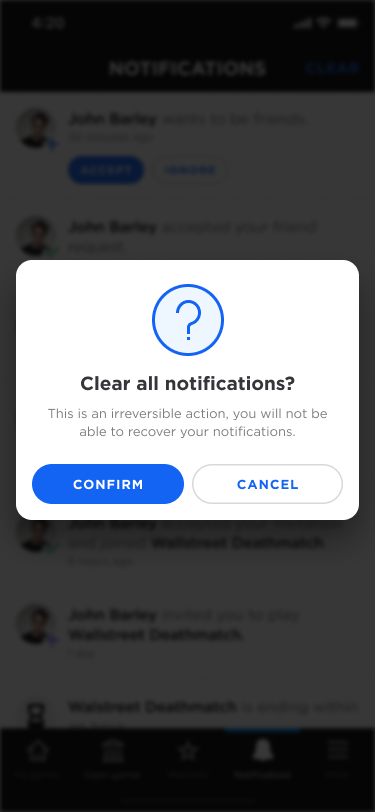
The system will track if notifications have been viewed. If a player visits this screen, then the notifications in the list will be marked as viewed.

The payer should be able to accept or ignore friend requests here. We could also route them to the Friends section if that would be easier.

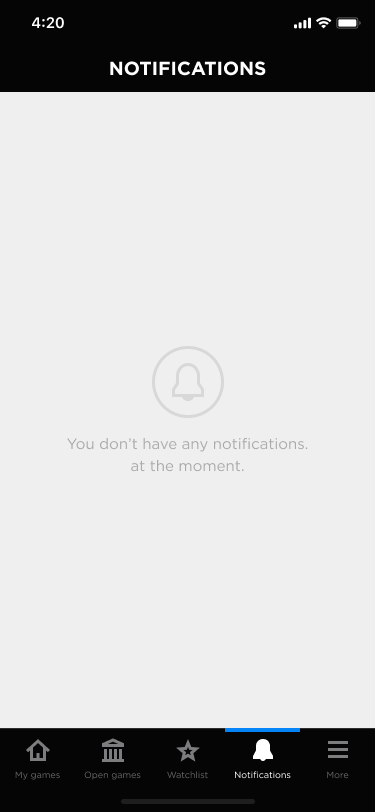
Clicking on any notification will jump the player to the approporate area in the application.

XXX We need to show each type of notification and where it will link to.

#### Clearing Notifications



#### Cleared Notifications



## 

## Questions / Issues

**How are checks going to get issued to a player for withdraw.**

The player can choose PayPal or eCheck. When a player requests for funds to be withdrawn, it needs to go into a queue and Brawl Street personnel will need to process the payments. Brawl Street may need the ability to export the transactions so they can upload them to PayPal and/or an eCheck service.

**Is there any period of time we should hold funds from withdraw for a player?**

When a player withdraws funds, we need to have text that states that their payment will be processed within 5 business days.

**How does a player cancel their account?**

In the Settings section we need a Cancel Account link.

**How does a player reactivate their account?**

This will need to be done by contacting support. Brawl Street admins with high security need the ability to reactivate a canceled account.

**We don’t have a way for a player to see the stock transactions for an active game. Should we provide this?**

Need to address this.

**How are we going to hold sensitive data (SS#, DOB, etc.) This needs to be super secure.**

This data should be held on a separate server.

**Players can only purchase and sell stocks when the market is open. Otherwise, they have to do limit orders. Need to see if IEX has an API call to know if the market is open or not.**

Limit orders happen during the course of the game. There are no trades when the market is closed. You can make your picks prior to the game starting with either market orders or limit orders which will execute once the game they have joined starts.

**For the levels, we need to have the names of the levels.**

TBD

**The system will need to constantly create games and only allow people at or below a level to join. Automatically filter these games for the player based on their level.**

This will be a stored procedure that runs every 15 minutes. If will look at what games are currently open and create any games that are missing. This enables the app to always offer a variety of open games.

**Can we combine the Intro screen and the login screens?**

**We need to discuss how a player stays logged in**

**Discuss what happens if someone tries to login to the app too many times.**

**We do not have any screen design for the using the phone’s location. I assume we don’t need this. If correct, then we should delete that section from the spec document.**

**If the game ends in the weekend, should we auto time it to end with the stock market closure i.e. friday evening?**

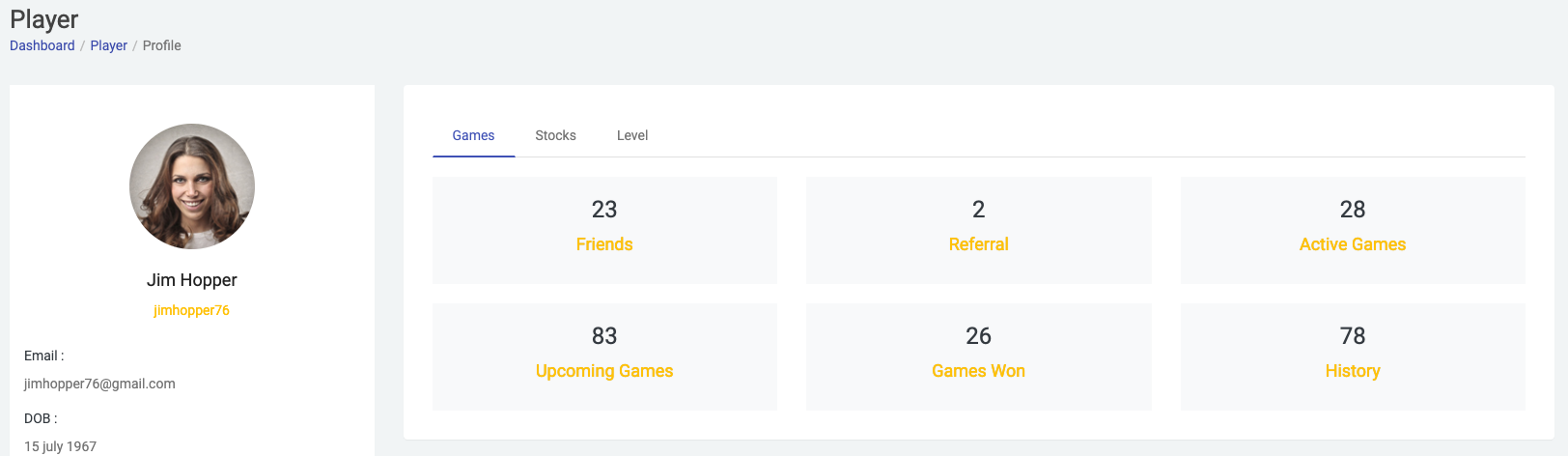
**We probably need a detail screen if the player clicks on a transaction in Wallet transactions list to show detail of the transaction.**

## 

## Admin Panel

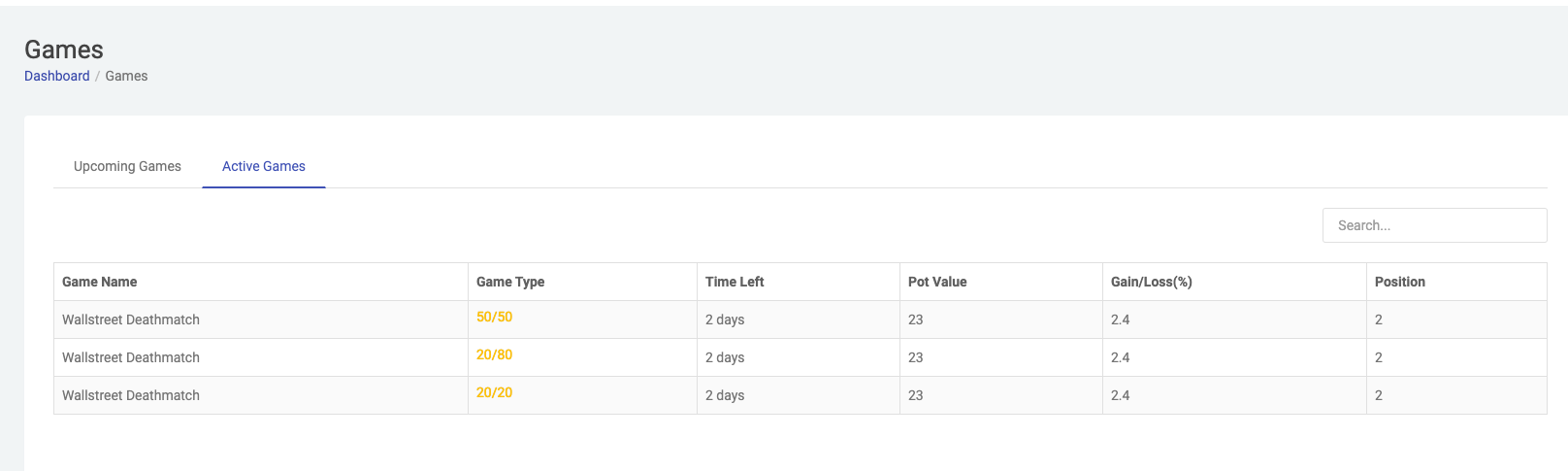
### Dashboard Review

--



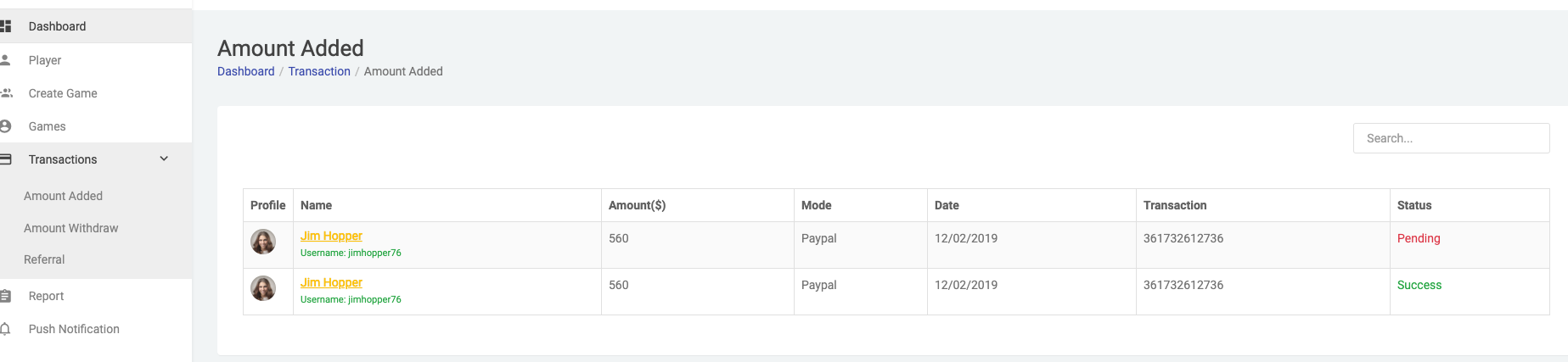
* This screen should have the following Tabs
  + Overview (shows blocks)
    - Wallet Balance
    - Level
    - Upcoming Games
    - Active Games
    - Completed Games
    - Winning Ratio
  + Games (list)
    - Game Name (hyperlink to view game detail)
    - Date Started
    - Game Type
    - Duration
    - Status (upcoming, active or completed)
    - Entry Fee
    - Amount Won
  + Referrals (list)
    - Player Name
    - Amount Won
    - Total Referral Fees
  + Transactions (list)
    - Date
    - Type
    - Game (if applicable)
    - Description
    - Amount (positive or negative)
* In the Player info on the left, I should be able to see which player referred this player.
* I assume that each of the blocks in the overview will click through to the appropriate screens.
* I should be able to suspend or reactivate a player from the player detail view. I should also be able to see their current status.

--



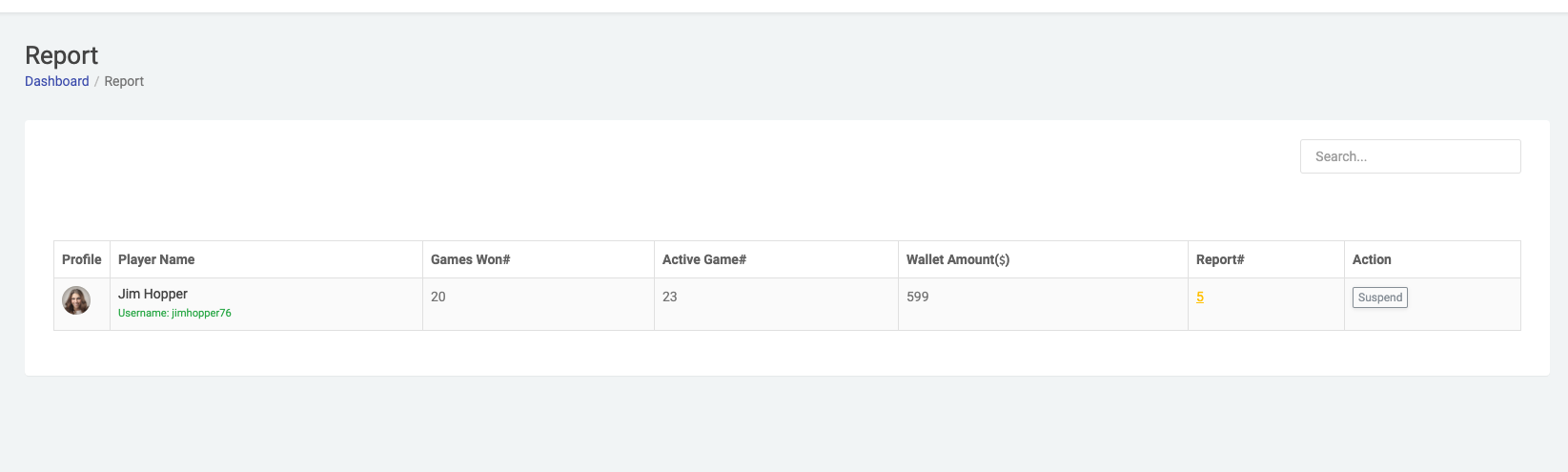
* I should be able to see completed games in addition to upcoming and active games.
* I should be able to click into the detail view of a game to see:
  + Core info about the game
  + Players in the game along with their virtual balance and rank.
  + Stock transactions by player
  + Current stocks owned for a player in the game.

--



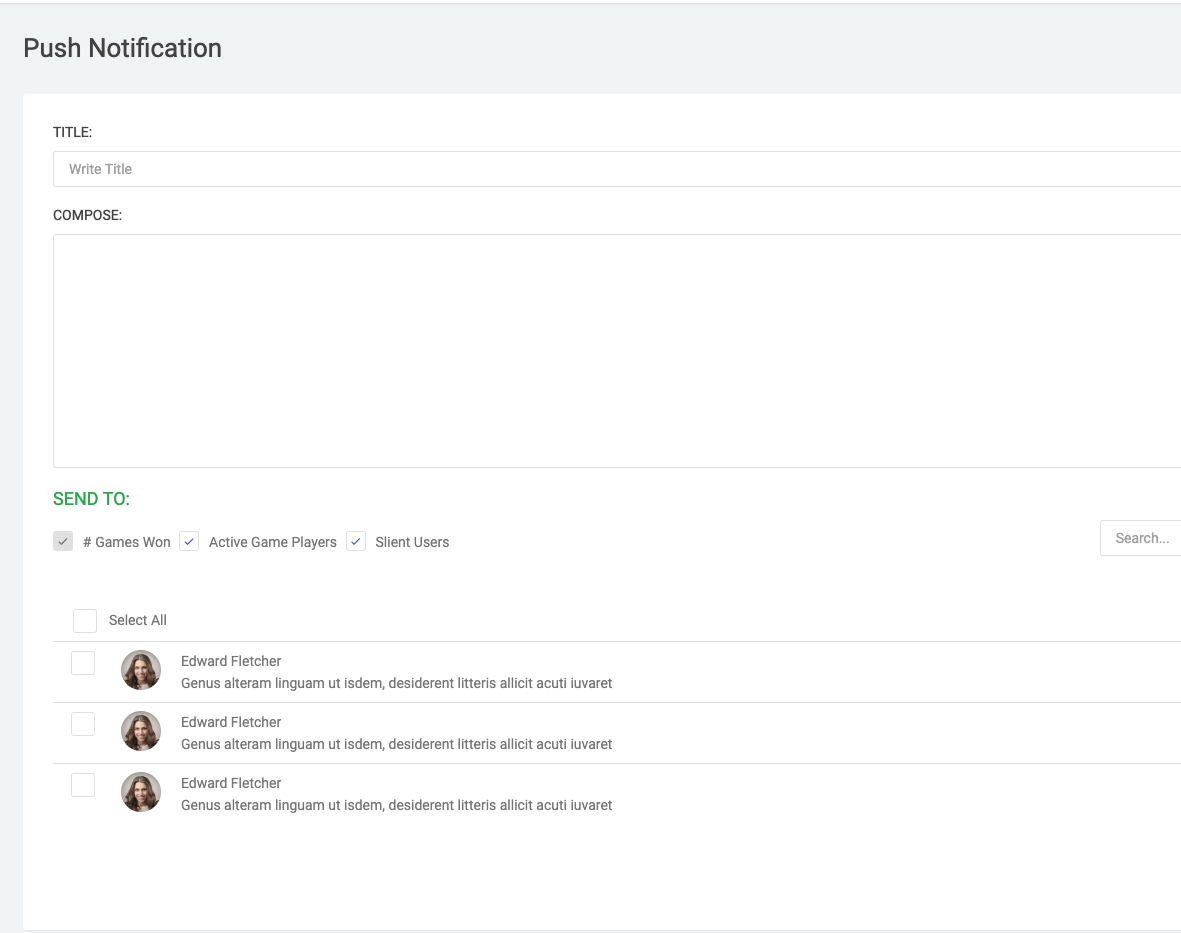
* This screen is showing deposits and withdraws across all players which is fine.
* Referrals should be named referral fees and show the following columns:
  + Date
  + Player (player who receives the fee)
  + Referral (player who played the game)
  + Game
  + Amount
* We should also have a section in transactions for Brawlstreet Revenue for completed games. It should contain the following:
  + Date
  + Game
  + Total Pot
  + Brawl Street Amt
  + Referral Fees
  + Net Amt (Brawl Street Amt - Referral Fees)

--



Not sure what this screen is for.

--



Not sure how this will be used.

We will need an admin panel for customer service. The following are the basic requirements:

### Use Cases

1. Security (Standard and Admin)
   1. Admins can:
2. Need the ability view
   1. Suspend Player
   2. Terminate Player
   3. Reactivate Player
   4. Update contact information
   5. Terminate a Game

### Suggested Screens

1. Players
   1. List View
      1. Filters
         1. Name (Contains)
         2. Status
      2. Fields
         1. Player Name
         2. Status
         3. Date Created
   2. Detail View
      1. Data:
         1. Contact Info
         2. Referred By
         3. Wallet Amount
         4. List of Games
         5. List of Transaction
         6. List of Referrals
      2. Actions
         1. Suspend Player
         2. Terminate Player
         3. Reactivate Player
         4. Update contact information
2. Games
   1. List View
      1. Filters
         1. Name (Contains)
         2. Status
      2. Fields
         1. Name
         2. Start Date
         3. Type
         4. Players
         5. Entry Fee
         6. Pot
         7. Status
   2. Detail View
      1. Data
         1. Core fields
            1. Name
            2. Start Date
            3. Type
            4. Players
            5. Pot
            6. Entry Fee
            7. Max Player
            8. Status
         2. List of Players
         3. Link to View Stock Trades
3. Stock Trades
   1. List View
      1. Filters
         1. Game
         2. Player
      2. Detail View
4. Transactions
   1. List View
      1. Filters
         1. Player
         2. Game
         3. Transaction Type
5. Brawl Street personnel will have access.
6. Security levels: Standard and Admin
7. Standard:
   1. View and search a list of Players
   2. View and search a list of games
      1. Search by game name
      2. List filter:
   3. Suspend or Terminate a Player Account
   4. Reactivate a Player Account
   5. Add Points to an Account
   6. View Player Profile
      1. View Games Created / Participating
      2. View Profile Info
      3. View all Stocks Owned
      4. View Level & Points
   7. View Game Profile
      1. View all Players & Portfolios
      2. Days / Remaining
      3. Pot Size & Basic Game Info
   8. Create a Public Game
   9. View All Transaction History
   10. View All Points Transaction History
   11. View Withdrawal Requests
       1. Process Check Requests / Update Status
   12. Push Notifications
       1. Send Manual Notification to all Players
       2. Send Manual Notification to Specific Players
   13. Dashboard
       1. # of Players
       2. # of Active Games
       3. Total Payouts
       4. List Top Grossing Games
8. Admin:
   1. Log into the Player Account
   2. Update Player profile info