skorch

Software Requirements Specifications

Will Harrison
Cody Henderson
Nath Tumlin
Thomas Willingham

03 April 2017

Change Log

Version	Summary	Author	Date
0.1	Document Created	Thomas Willingham	27 January 2017
0.2	Initial Draft	Thomas Willingham	20 February 2017
0.3	Added Glossary, Scope	Cody Henderson	25 February 2017
0.4	Updated Glossary	Cody Henderson	28 February 2017
0.5	Added High-Level Class Diagrams	Nath Tumlin	28 February 2017
0.6	Added Functional Requirements and Non-Functional Requirements	Will Harrison	01 March 2017
0.7	Added Use Case Diagrams, Activity Diagrams	Cody Henderson	01 March 2017
0.8	Updated Class Diagram	Nath Tumlin	01 March 2017
0.9.1	Added introduction, Description, Requirements, Diagrams	Nath Tumlin	01 March 2017
0.9.2	Added System Architecture, User Interface	Thomas Willingham	02 March 2017
0.9.3	Updated Diagrams Descriptions	Will Harrison	02 March 2017
1.0	Finalized Document for Requirements Submission	Cody Henderson, Will Harrison, Nath Tumlin, Thomas Willingham	02 March 2017
1.1.1	Added Join, View Update Game Activity Diagram	Will Harrison	24 March 2017
1.1.2	Added Add/Remove Game Activity Diagram	Will Harrison	24 March 2017
1.2	Updated Glossary	Cody Henderson	24 March 2017
1.3	Updated Scope	Cody Henderson	24 March 2017
1.4	Added Class Diagram	Thomas Willingham	29 March 2017
1.5	Updated and Added UI Images	Thomas Willingham	02 April 2017
1.6	Added Activity Diagram	Thomas Willingham, Nath Tumlin	02 April 2017
1.7	Added New Class Diagram	Thomas Willingham	03 April 2017
2.0	Finalize Document for Design Submission	Cody Henderson, Will Harrison, Nath Tumlin, Thomas Willingham	03 April 2017

Table of Contents

1: Introduction

1.1: Purpose

The purpose of this document is to define the requirements for Skorch. This document describes the problem Skorch attempts to solve, describes functional and nonfunctional requirements for Skorch, and provides use case diagrams, activity diagrams, and a high level class diagram.

1.2: Scope

Skorch primarily provides a GUI interface that will run in modern web browsers. Using this interface, users can create game sessions to keep track of scores for a wide variety of game paradigms. When a game is created, a public and private game phrase will be generated to allow the game session to be shared between users with two different levels of permissions. User account management will allow registered users to manage a list of games with which they previously interacted. Skorch will be a real-time web application that pushes game session updates to connected users as the changes occur. Additionally, Alexa users will be able to use voice commands to modify activate game sessions.

1.3: Glossary

<u>Skorch:</u> the service that allows users to create games and keep track of scores from a variety of devices

<u>Game Center</u>: the place where a user can manage and create games and manage user account settings

<u>Game</u>: a session that allows a user to keep track of scores based on the game type

<u>Game Type</u>: settings imposed on a game that describe the number of players, duration, scoring rules, and winning conditions

<u>Tournament</u>: a set of similar game types set up in a bracket style progression

<u>Game Code</u>: a three- or four-word phrase used to identify a unique session

<u>Public Game Code</u>: the game code that lets any user view a game

<u>Private Game Code</u>: the game code that lets registered users view and control a game

Public Tournament Code: the game code that lets any user view a tournament and all games within it

Private Tournament Code: the game code that lets registered users view and control a tournament and all games within it

GUI Interface: the interface accessible from a desktop or mobile browser that allows users to create, join, query and update a game’s status

Voice Interface: the set of voice commands that use the Skorch API to join, query and update a game’s status

MeteorJS: a complete platform for building web and mobile apps in pure JavaScript

ReactJS: a declarative, efficient, and flexible JavaScript library for building user interfaces

Registered user: a user of Skorch that has created an account to save games

2: Project Description

Skorch will be a management platform for games and tournaments. Users should be able to easily create a game or tournament, with or without logging in. Games can be quickly created by defining its type, how it is scored, the win conditions, and the players. They should then be able to update the status of the game as it is played. A link should be made available to share a live-updating, view-only page for the individual game. Users will also be able to group multiple games into tournaments with similar management and sharing functionalities. Additionally users will be able to associate games with their accounts for easy access to managing and viewing previous results of games.

Skorch will be built to support multiple platforms, including the web and Amazon Alexa, in an attempt to reach as many users as possible. Potential users extend beyond people looking to manage literal games and tournaments, and extends to any sort of tracking of “points”, such as a teacher keeping track of students’ good behavior or participation.

3: Requirements

3.1: Functional Requirements

- The user will be able to create a single-user game (HIGH PRIORITY)
- The user will be able to create a multi-player game (HIGH PRIORITY)
- The user will be able to save a game’s score. (MEDIUM PRIORITY)
- The user will be able to view a game’s score.(HIGH PRIORITY)
- The user will be able to update a game’s score. (HIGH PRIORITY)
- The user will be able to join a game. (MEDIUM PRIORITY)
- The user will be able to create a tournament bracket. (MEDIUM PRIORITY)

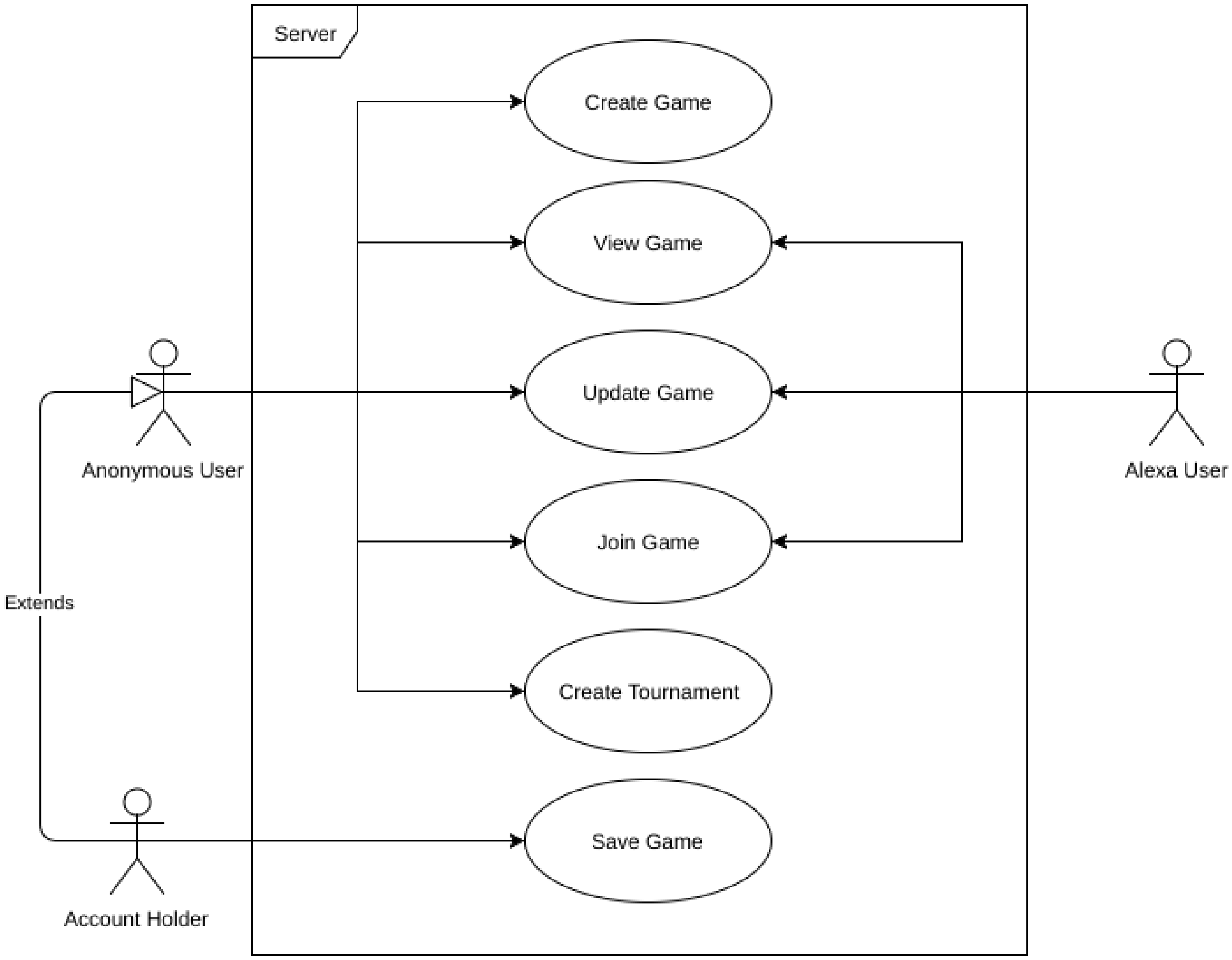
3.2: Non-functional Requirements

- The Alexa must respond to requests within a few seconds. (HIGH PRIORITY)
- The game creation process should be easy, with a user being able to load the website, create a simple game based on provided defaults, and obtain link to share within about 45 seconds. (HIGH PRIORITY)
- Game codes should be easy to memorize and share. (LOW PRIORITY)
- Skorch should be able to support at least 50 concurrent users on an AWS t2.micro EC2 instance. (HIGH PRIORITY)

4: Diagrams

The following diagrams will aid in the understanding of the Skorch system.

4.1: Use Case Diagram

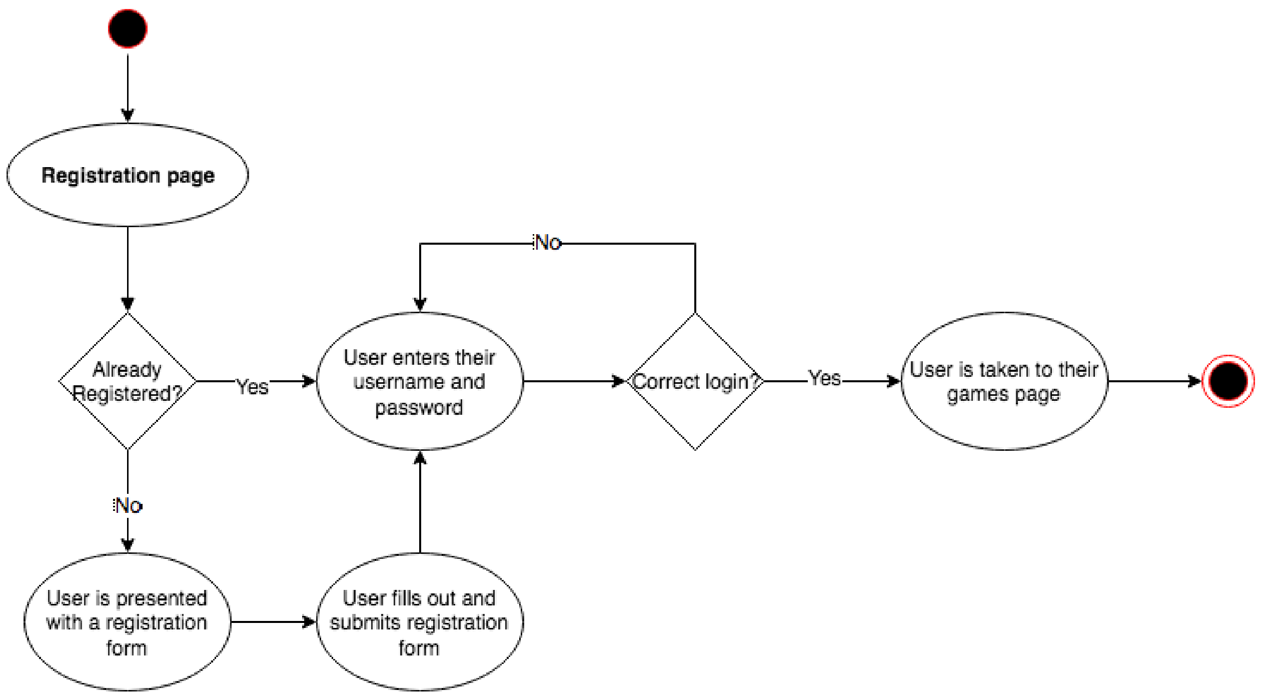


All users, anonymous and registered, are allowed to create games and tournaments. They are also allowed to join, view and update games based on the settings of the individual games. Registered users, or account holders also have the ability to save games to their account to be referenced again in the future if necessary. Alexa users have a smaller subset of actions; these voice-interface users will be able to join, view and update games based on the game settings, but we do not expect them to be able to create games through the voice interface.

4.2: Activity Diagrams

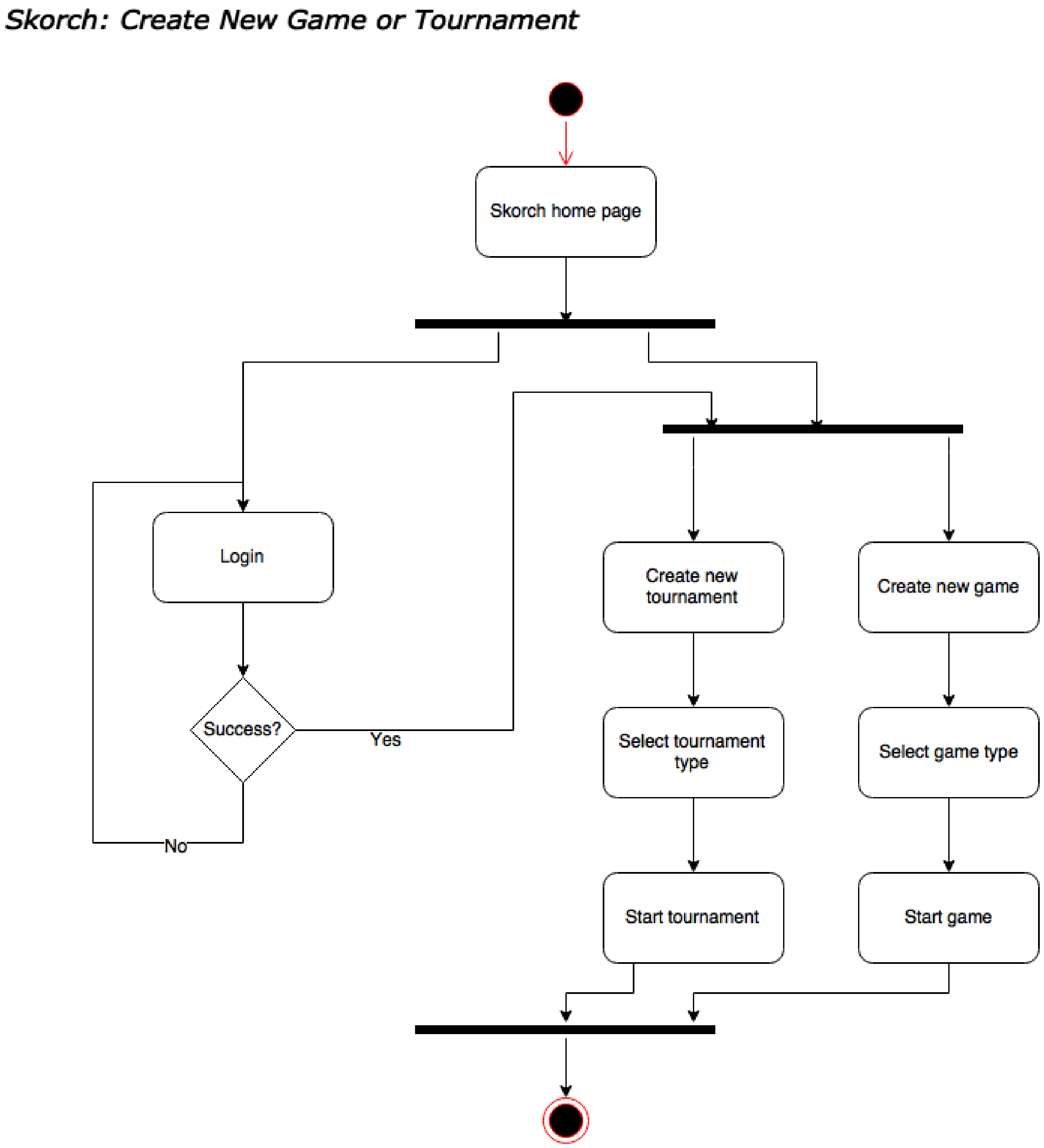
4.2.1 Registration and Login

Skorch: Registration and Login


Registration page
No
Already Registered?
Yes
User enters their username and password
Correct login?
Yes
User is taken to their games page
No
User is presented with a registration form
User fills out and submits registration form

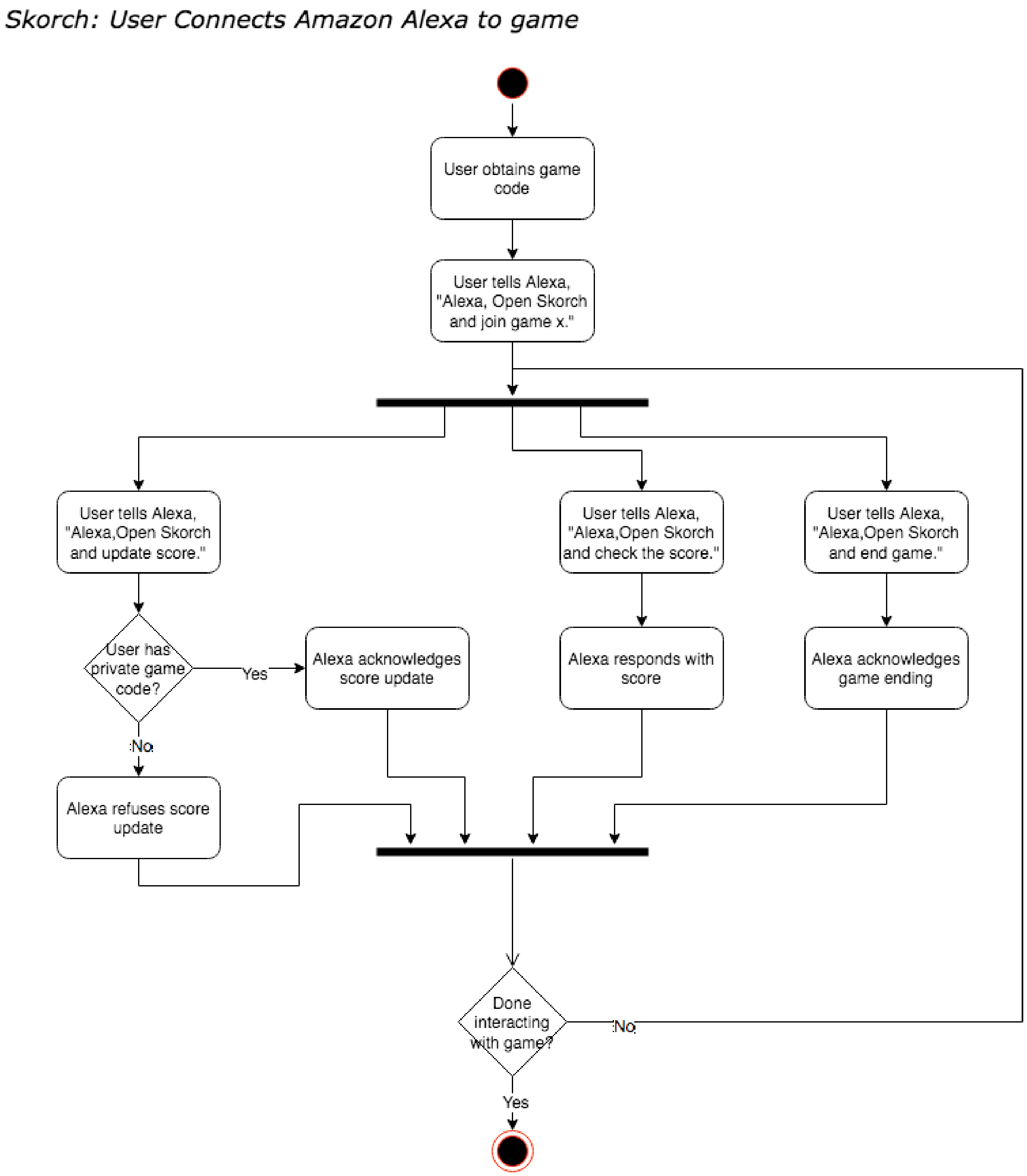
When logging in on a GUI interface, a form allows users to log in if they already have an account. If the user is not already registered, the interface will allow the user to register through a registration form. Once registered, the user can enter their username and password. If the combination matches a user in the system, the user is taken to the game center page.

4.2.2 Create New Game or Tournament



Users can use the GUI interface to create new game sessions. Users have the option of logging in to save game sessions. Any user can create a new tournament from the home page, set the tournament type, and start the tournament. Users can also create a new game, select the type, and start the session.

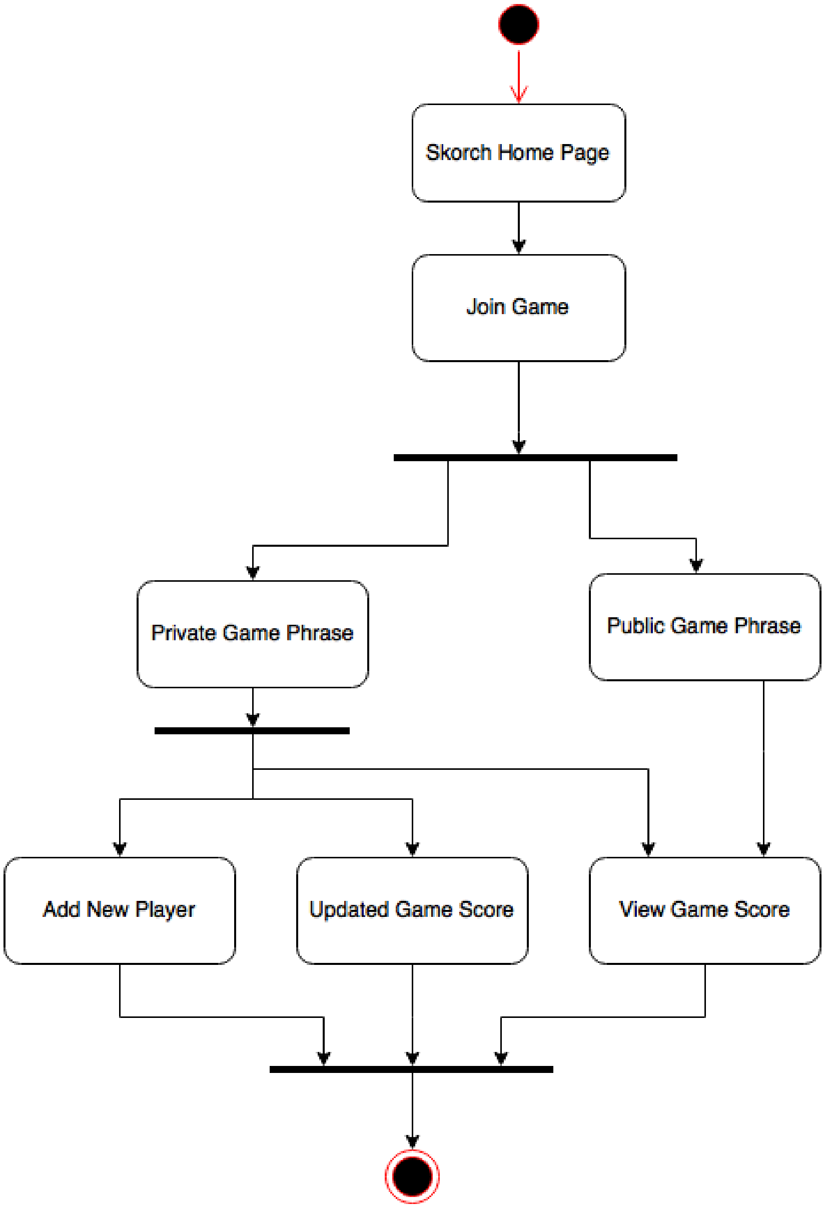
4.2.3 User Connects Amazon Alexa to Game


Skorch: User Connects Amazon Alexa to game
User obtains game code
User tells Alexa, "Alexa, Open Skorch and join game x."
User tells Alexa, "Alexa,Open Skorch and update score."
User tells Alexa, "Alexa,Open Skorch and check the score."
User tells Alexa, "Alexa,Open Skorch and end game."
User has private game code?
Yes
Alexa acknowledges score update
Alexa responds with score
Alexa acknowledges game ending
No
Alexa refuses score update
Done interacting with game?
No
Yes

An Alexa user can interact with Skorch using voice commands. Using the public or private game code, the user can tell Alexa to join a game session. Once joined, the user has a few options. The user can update the game score if the game was joined with a private game code. The user can check on the score of the game, and the user can end the game when complete.

4.2.4 Join, Update, View Game

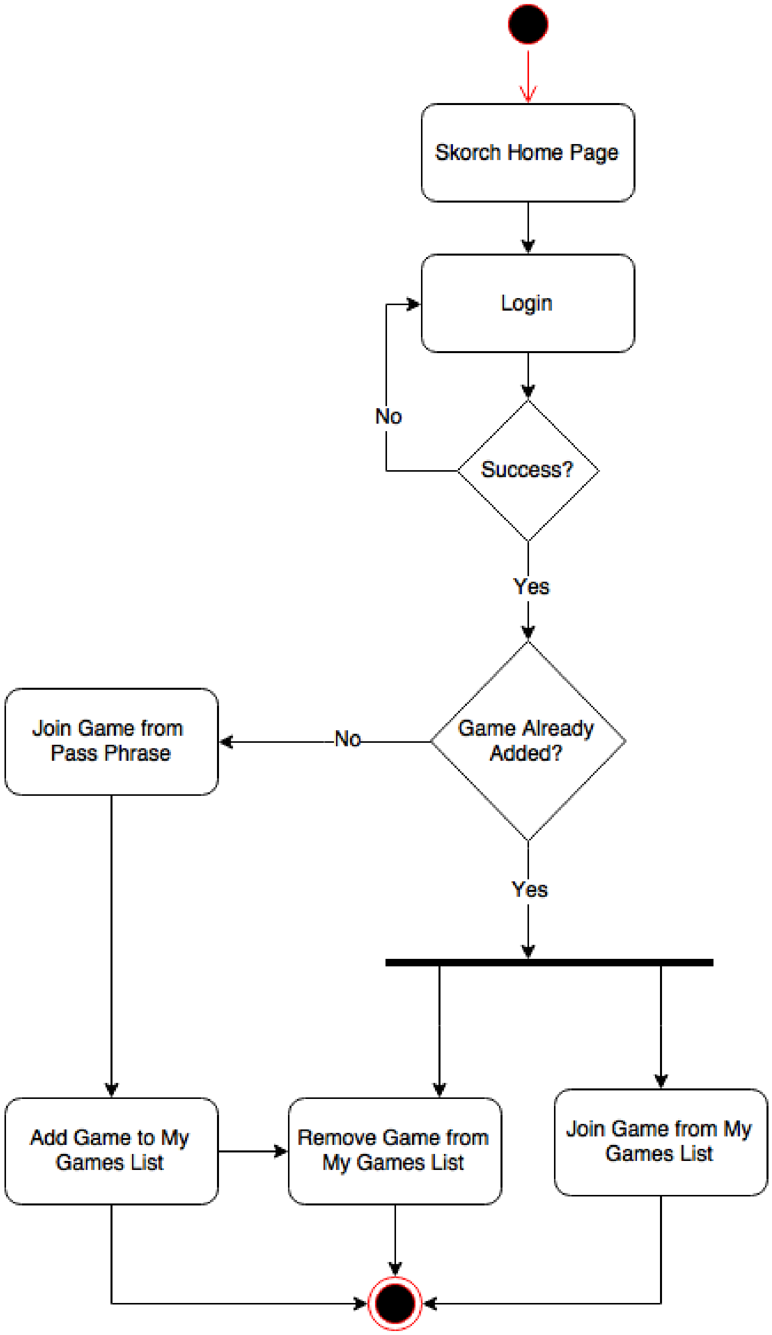
Skorch: Join Game, Update, View Game



User can join their game with a “private game phrase” or a “public game phrase”. A public game phrase allows the user to simply view the game score only. A private game phrase allows a user to view game score as well as add any new players and update the game’s score.

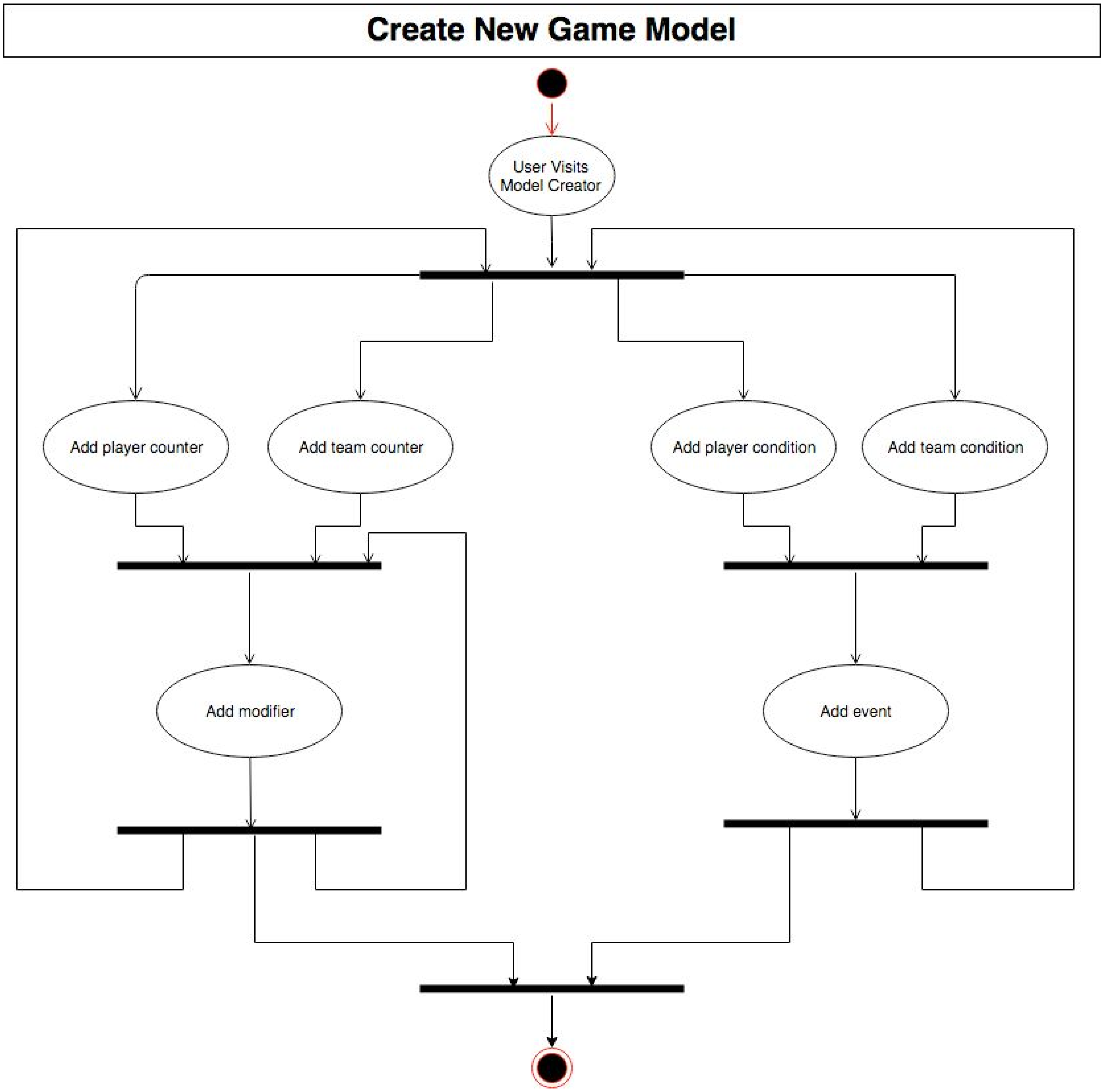
4.2.5 Add/Remove Game from Saved Games

Skorch: Add/Remove Game from My Games


Skorch Home Page
Login
No
Success?
Yes
Game Already Added?
No
Join Game from Pass Phrase
Yes
Add Game to My Games List
Remove Game from My Games List
Join Game from My Games List

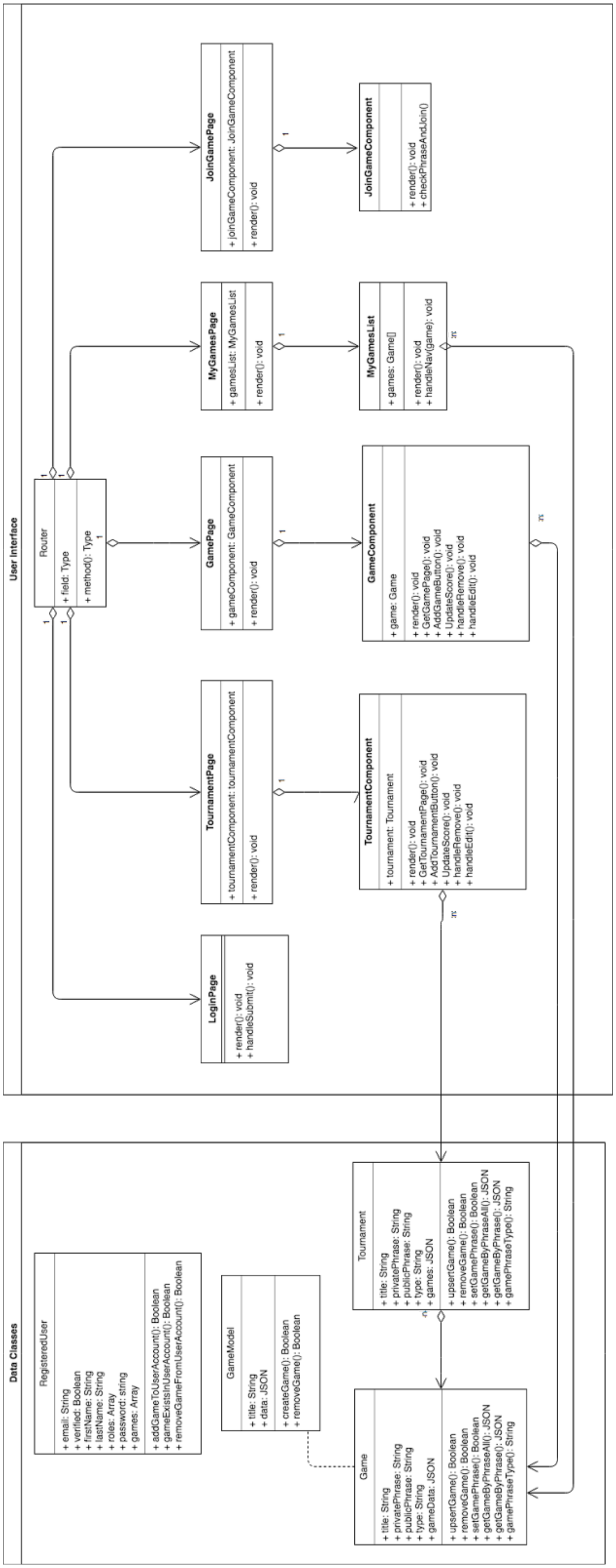
A registered user can login to view their saved games (My Games). If a game has already been added to a user’s saved games, they can join the game or remove the game. If the user has not previously added the game to their saved games list, they must join the game to then add the game to their saved games list.

4.2.6 Create New Game Model



A user can design their own game type by creating a game model. Once the model is created, they will be able to use that game type for any games they create.

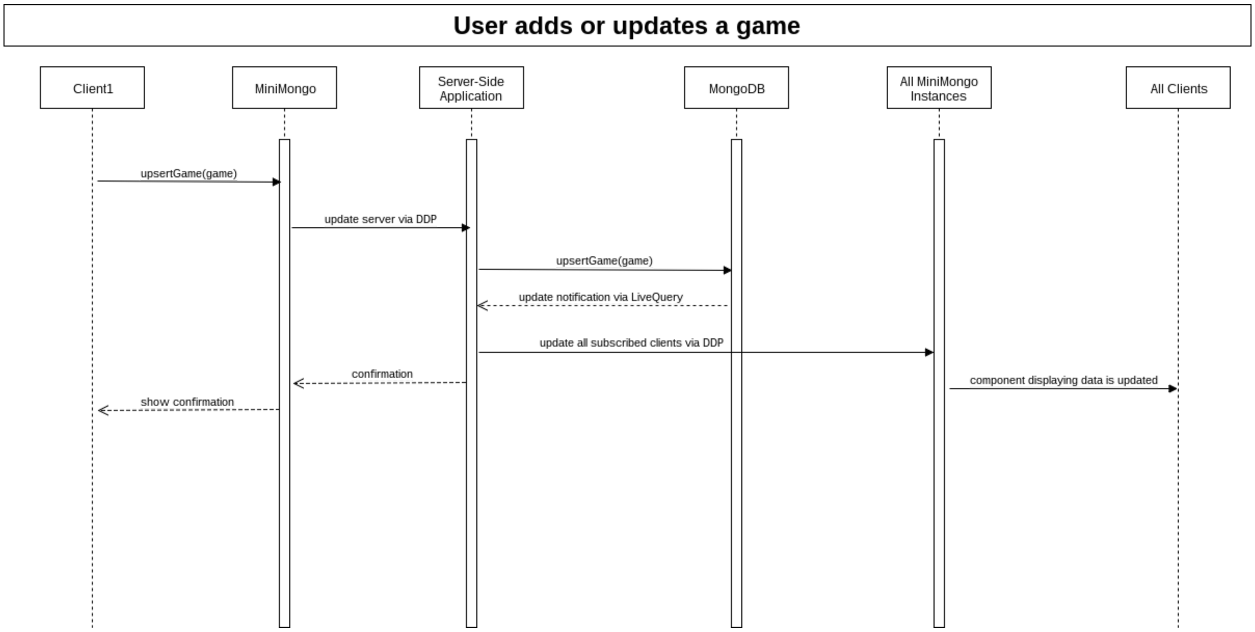
4.3: High-level Class Diagram



In our system, a User will have the most control in the system. A User may create a Game or Tournament, and a User may use a GameCode to find a Game or Tournament. A RegisteredUser has the same functionality as a User, but can also save Games or Tournaments. A Tournament is made up of one or more Games. A Game has a GameType that stores the settings for the game session. Games and Tournaments also have a GameCode with is used to identify the Game.

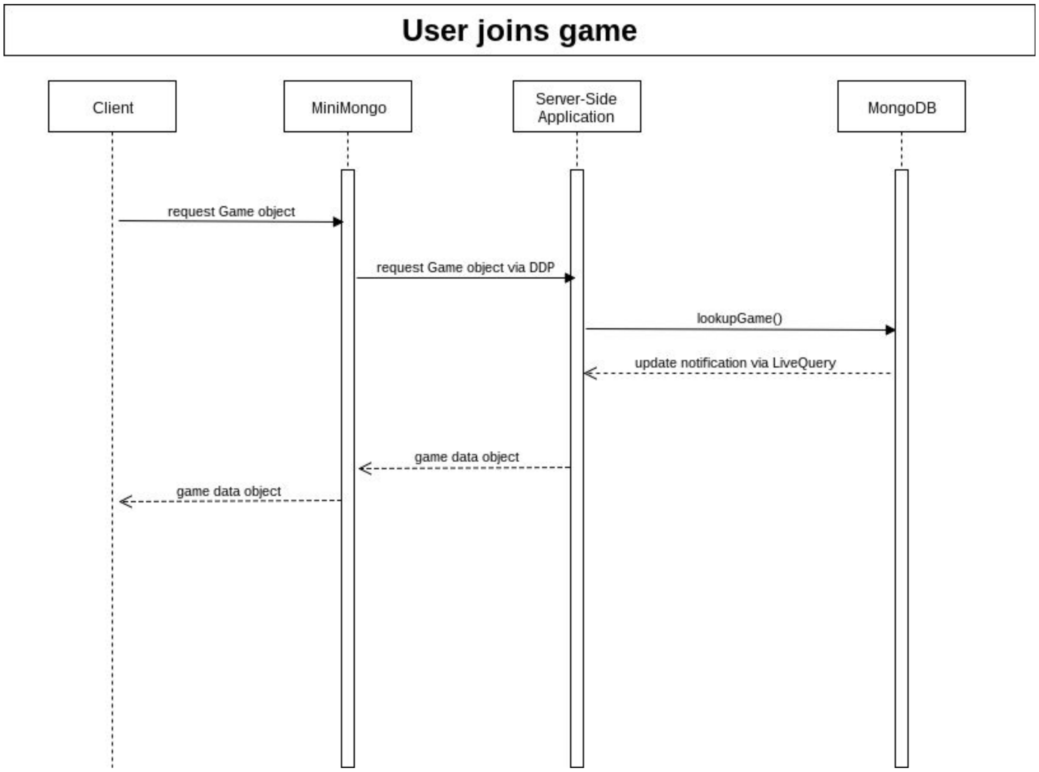
4.4 Sequence Diagrams

4.4.1 User adds or updates a game



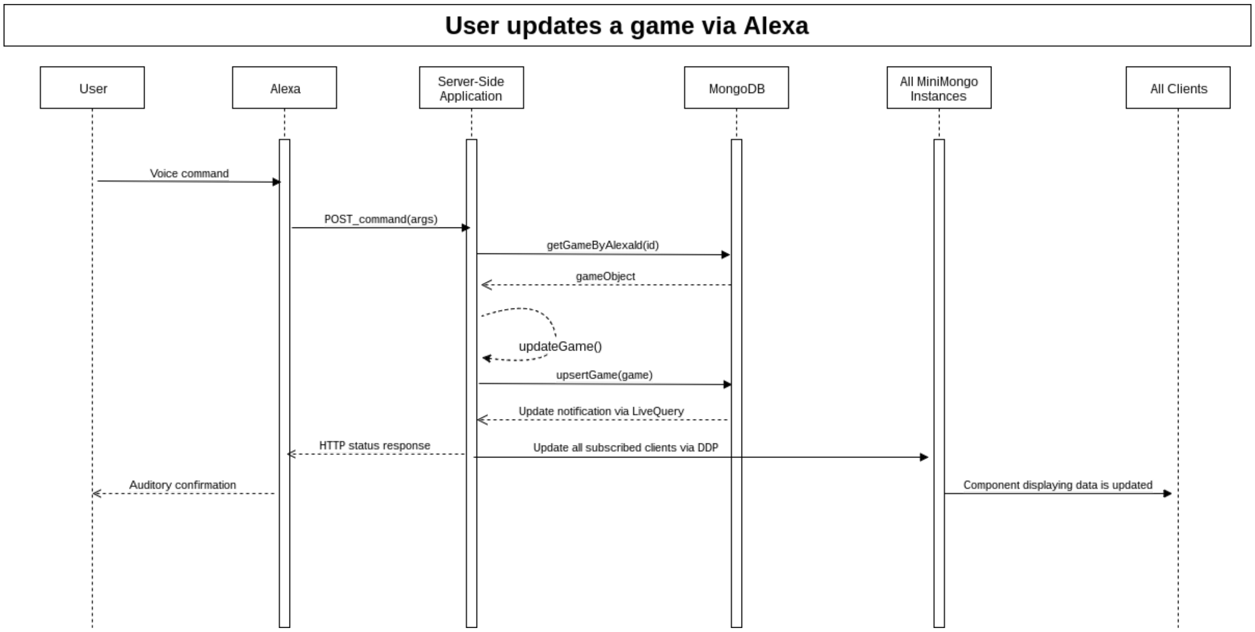
When a client makes a change to a game, the client passes the updated game to the local MiniMongo instance. Once MiniMongo saves the change, it will push the change to the server-side application via the Direct Data Protocol (DDP). The server application will send the updated game to the main Mongo database. Once the database accepts the changes, it reports the change to the server application via LiveQuery. The server application will use DDP to alert any MiniMongo instances that are subscribed to that object who in turn changes the components on the client GUIs. The original client is also alerted of the success of the change.

4.4.2 User joins a game



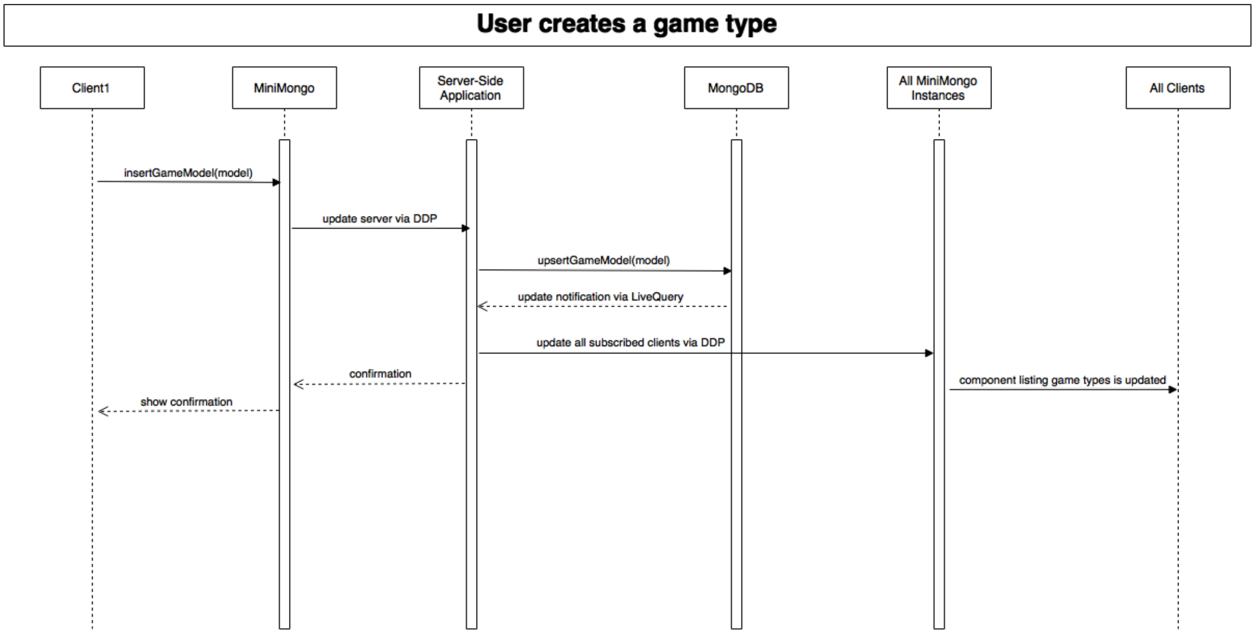
When the client requests a game to join, the client initially contacts the local MiniMongo instance. If MiniMongo does not have a copy of the game object, it will subscribe to the game through the server application. The server requests the object from the main Mongo instance. The data object will be passed down back down to the client and the client will receive future notifications as per diagram 4.4.1.

4.4.3 Alexa user updates a game



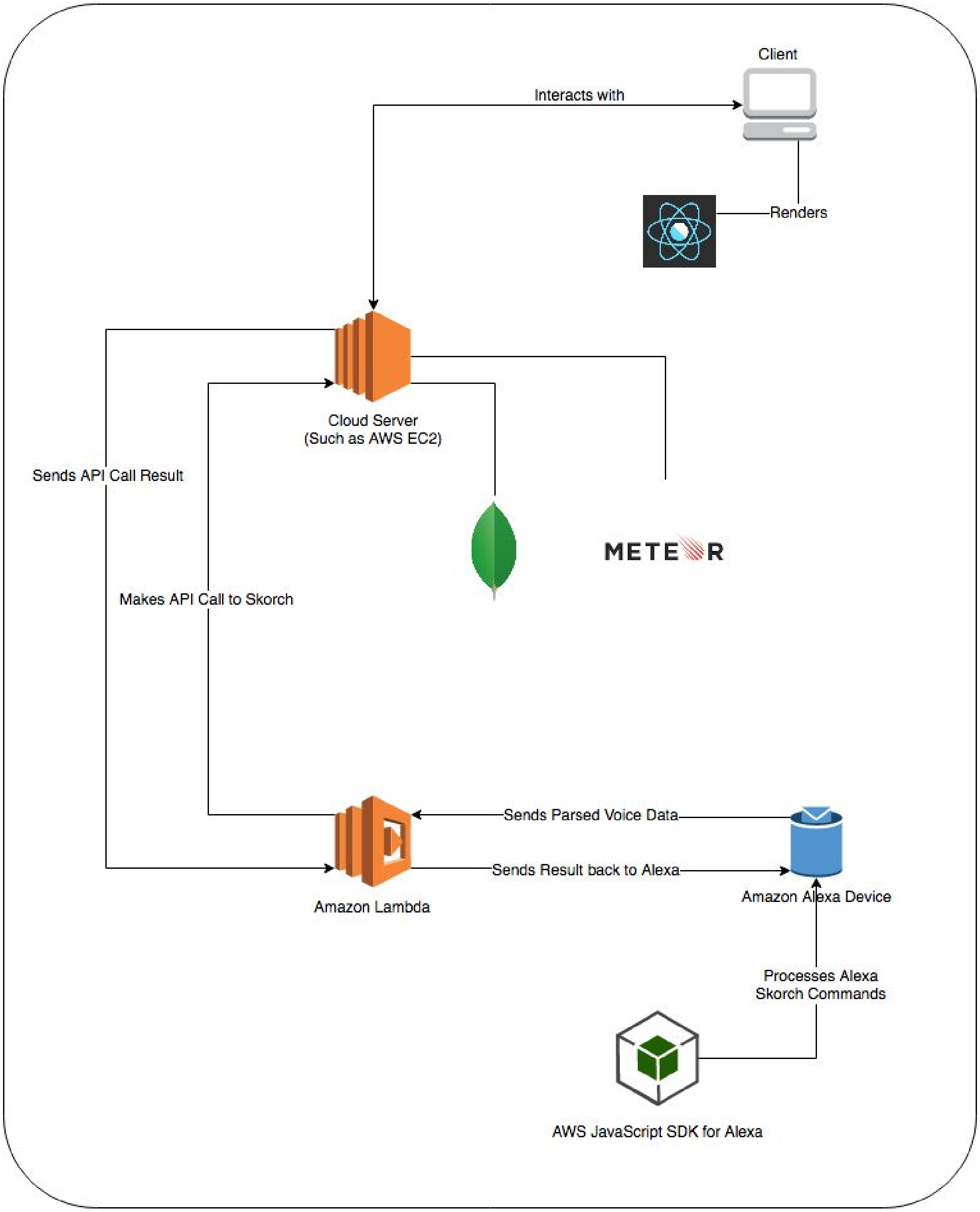
When a user issues a voice command, Alexa will use HTTP post to send the spoken command to the server-side application. The server application will query the main Mongo Database to see which game the Alexa instance is associated with. The server application will perform the correct adjustments to the game data and will push the updated game data to the Mongo instance. Once the database accepts the changes, it reports the change to the server application via LiveQuery. The server application will use DDP to alert any MiniMongo instances that are subscribed to that object who in turn changes the components on the client GUIs. The Alexa will also receive a response and will notify the voice user with an auditory notification.

4.4.4 User creates a game type



When a client makes a change to a game model, the client passes the updated game model to the local MiniMongo instance. Once MiniMongo saves the change, it will push the change to the server-side application via the Direct Data Protocol (DDP). The server application will send the updated game model to the main Mongo database. Once the database accepts the changes, it reports the change to the server application via LiveQuery. The server application will use DDP to alert any MiniMongo instances that are subscribed to that object who in turn changes the components on the client GUIs. The original client is also alerted of the success of the change.

5: System Architecture



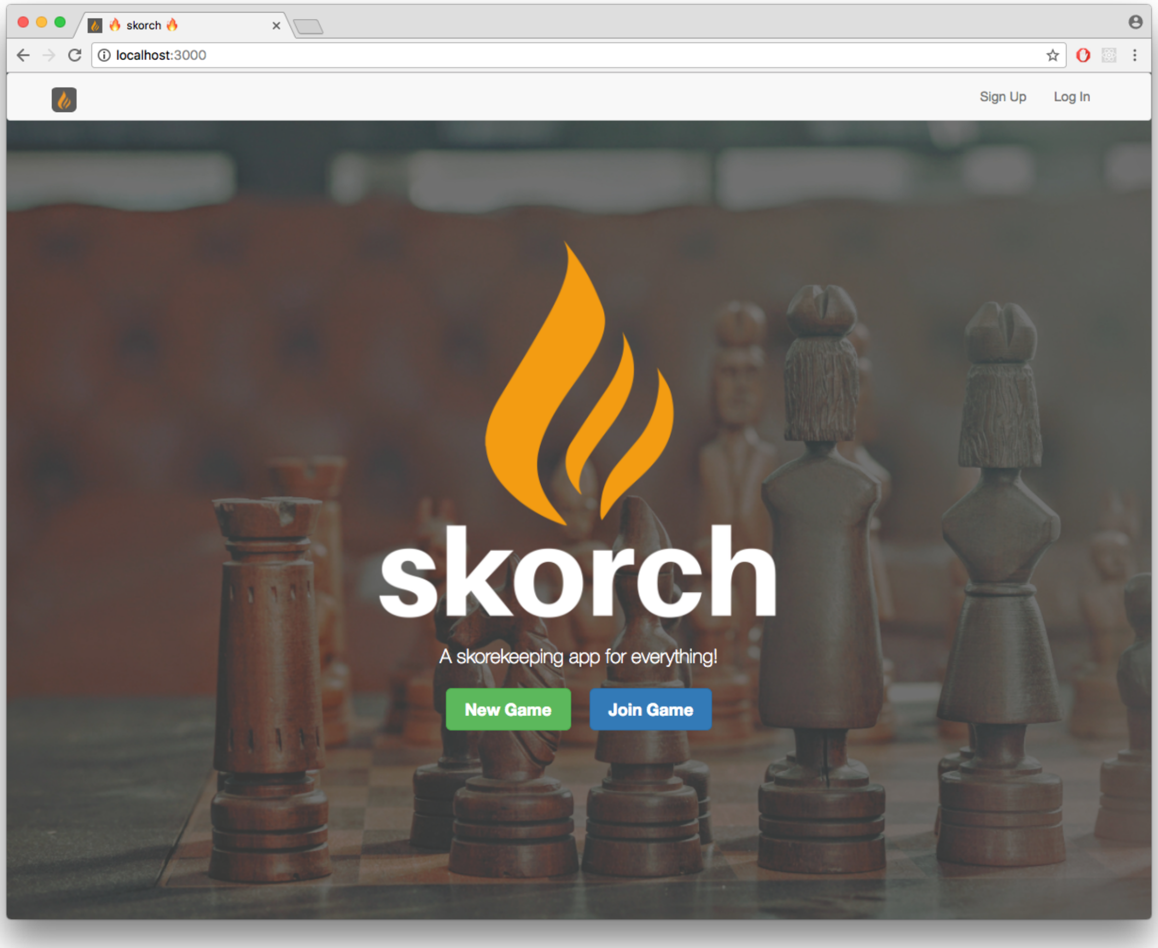
Skorch will employ a number of different technologies and elements in its system architecture. The primary server will run on an Amazon Web Services (AWS) EC2 t2.Micro instance. This server will be a linux server that runs NodeJS, MeteorJS, and MongoDB. All of the data storage and core business logic will be performed on this server. All web and mobile clients will render HTML using ReactJS, and they will connect to the primary server directly.

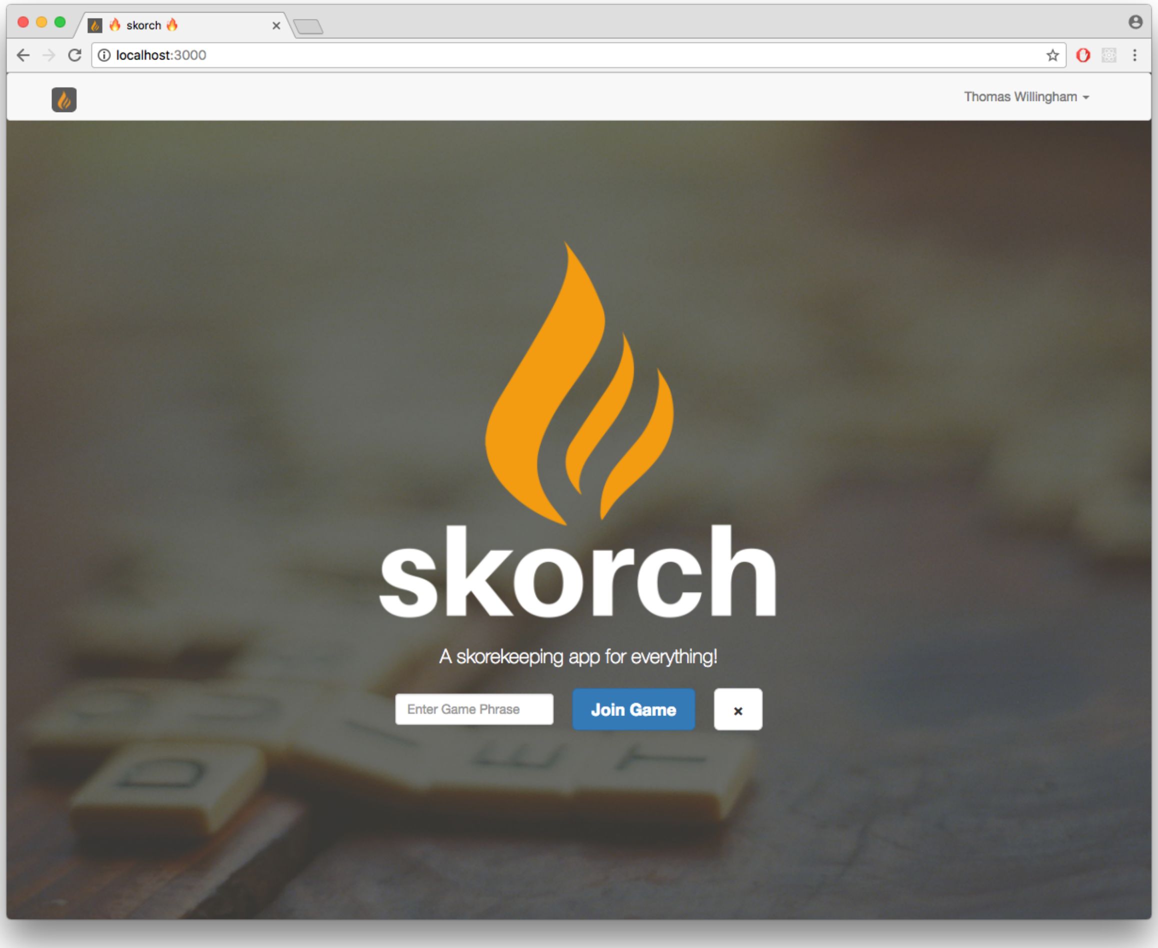
The Amazon Alexa requires a few more pieces to integrate with Skorch. The Amazon Alexa will connect to an Amazon Lambda instance that will host Skorch's AWS Alexa SDK code. All of the parsed voice commands from the Skorch Alexa skill will be sent to the Lambda server, which will in turn determine the proper API calls that need to be made to Skorch. The Lambda server will then make those API calls to Skorch and send the response data back to the Alexa.

6. User Interface

The Skorch application UI will consist of four main components: the home screen, the user account screen, the game screen, and the tournament screen.

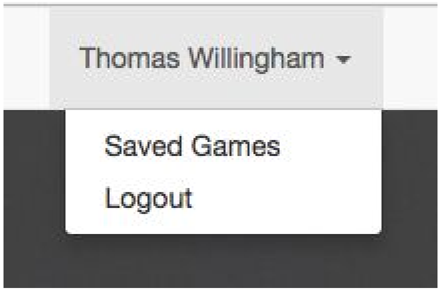
6.1: Home Screen

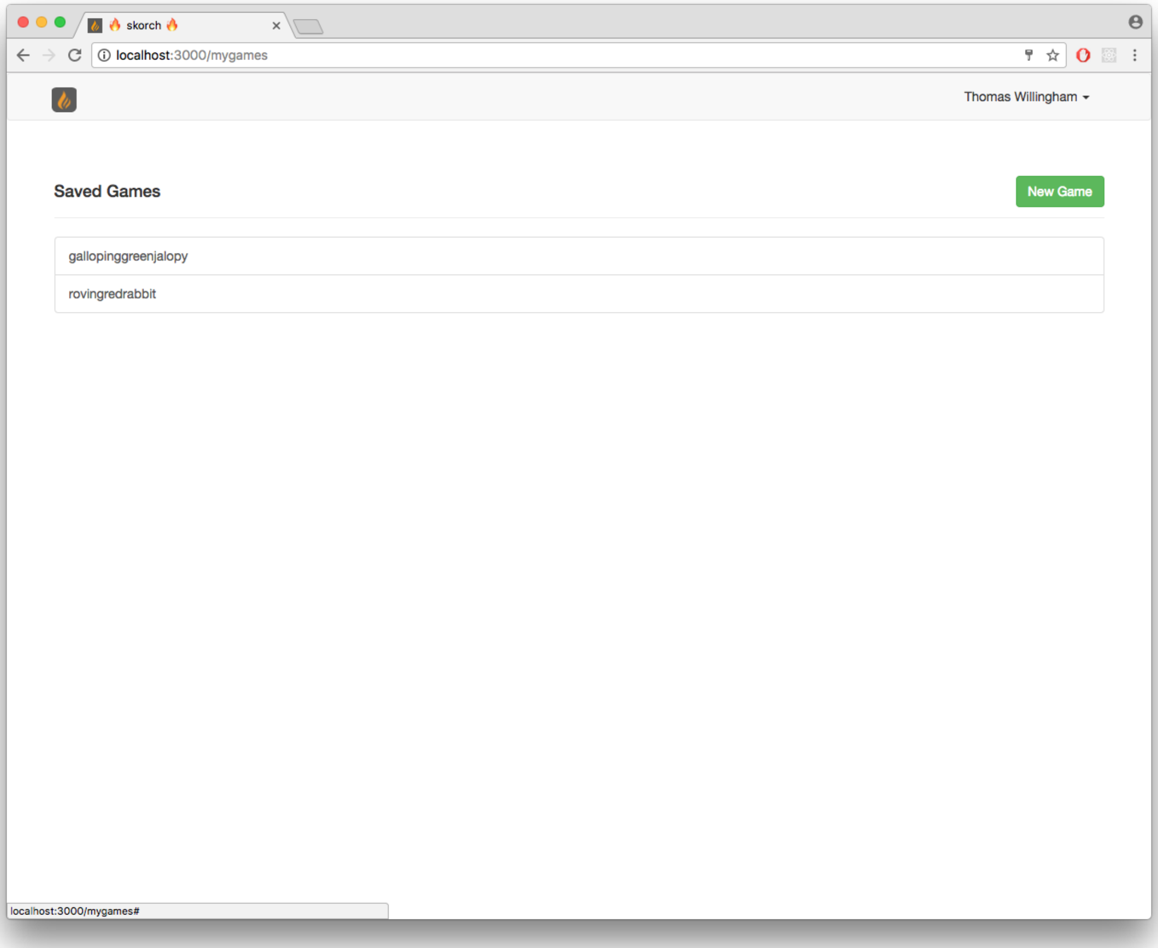




On the home screen, the user will have the option to create a game or tournament, enter game or tournament code, or log in.

6.2: User Account Screen





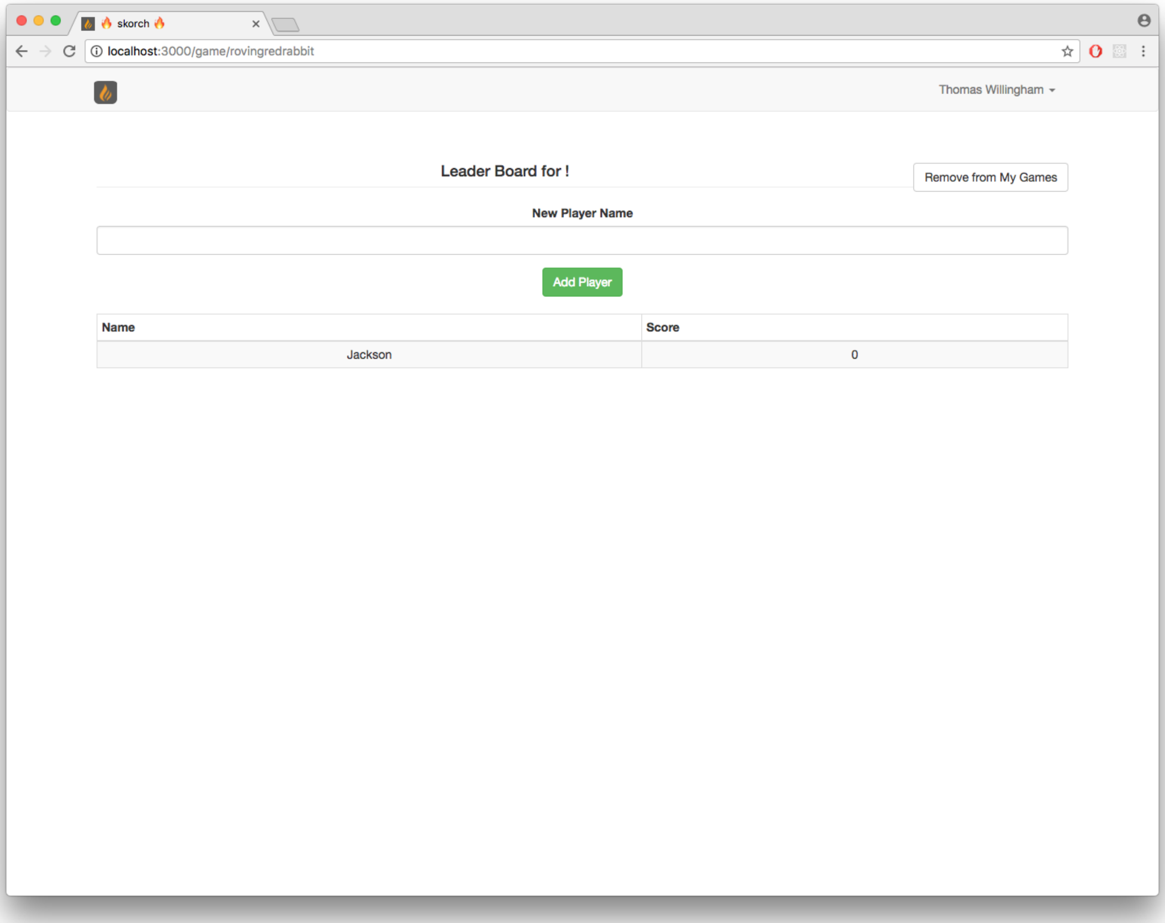
Saved Games

gallopinggreenjalopy

rovingredrabbit

Once logged in, a user can go to their account page to view a list of their saved games. The user will be presented with the scores of in progress games they have saved, and saved completed games will show the game's winner.

6.3: Game Screen



Leader Board for !

New Player Name

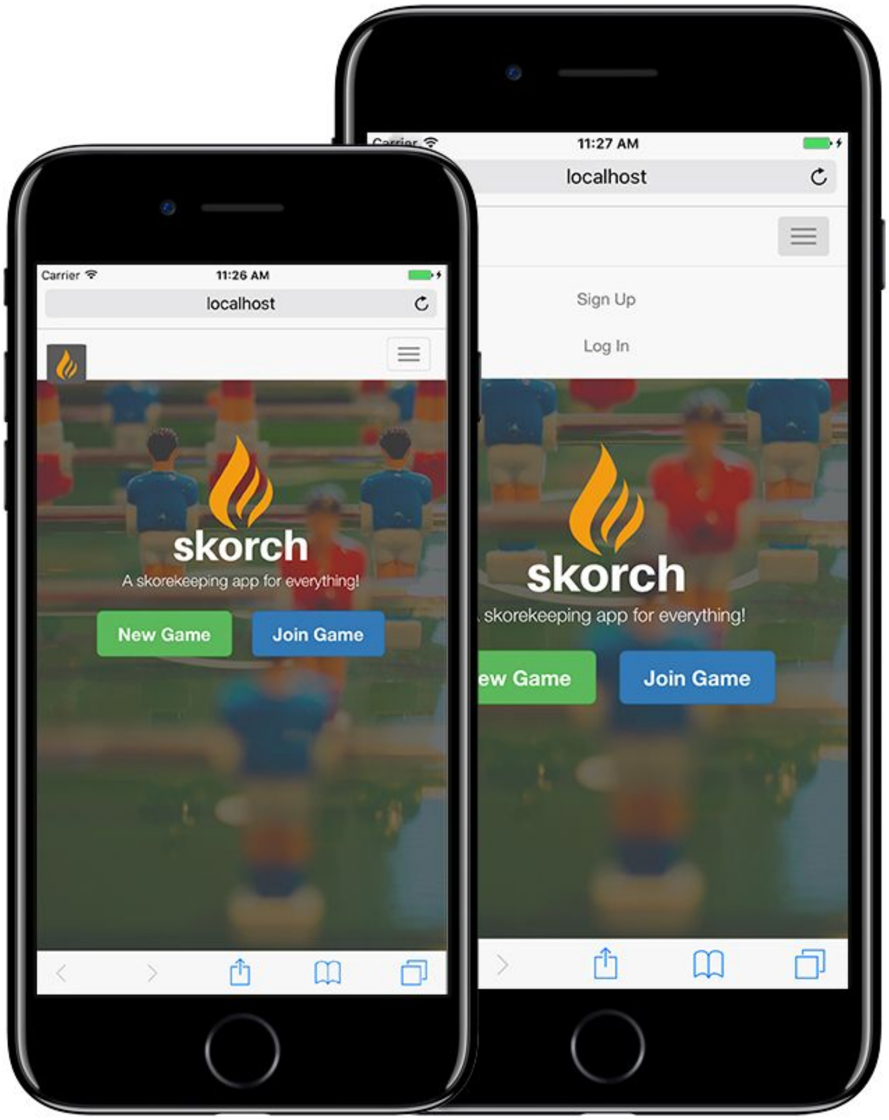
Name	Score
Jackson	0

The game screen will show the scores of the players of the game. If the screen was accessed with a private game code, the user will also have the ability to update the scores of each player and edit the game's configuration such as scoring method, the number of players, and player names.

6.4: Tournament Screen

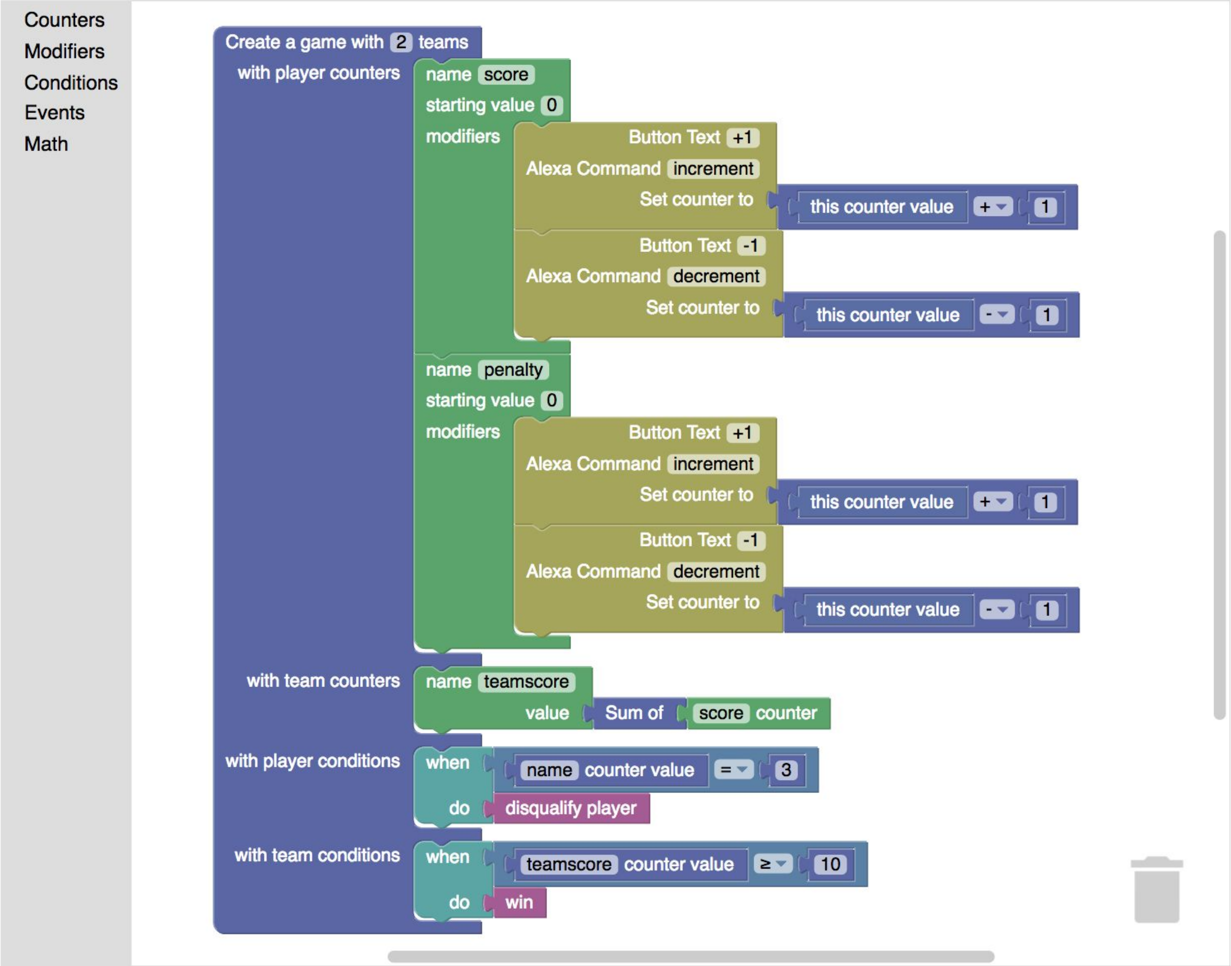
The tournament screen will show the status of a tournament, including scores and results from its individual games and the progression of players or teams through a bracket. Users will be able to navigate to the game screens of individual games in the tournament. If the tournament screen was accessed with a private tournament code, the user will also have the ability to edit the tournament's configuration such as the number of games played, participating players and teams, elimination style, and seeding.

6.5: Skorch on Mobile



Skorch is responsive and will work with screens of all sizes. This is an example of what it might look like on an iPhone 7 and iPhone 7 Plus.

6.6: Game Model Designer



```
Create a game with 2 teams
  with player counters
    name score
    starting value 0
    modifiers
      Button Text +1
      Alexa Command increment
      Set counter to  this counter value + 1
      Button Text -1
      Alexa Command decrement
      Set counter to  this counter value - 1
    name penalty
    starting value 0
    modifiers
      Button Text +1
      Alexa Command increment
      Set counter to  this counter value + 1
      Button Text -1
      Alexa Command decrement
      Set counter to  this counter value - 1
  with team counters
    name teamscore
    value  Sum of  score counter
  with player conditions
    when  name counter value = 3
    do  disqualify player
  with team conditions
    when  teamscore counter value ≥ 10
    do  win
```

The Skorch game model designer is built using Google's Blockly. It allows a user to specify the number of teams in a game, and create counters associated with players or teams. The user then gives counters modifiers, which specify how the counter changes either by button press or Alexa command. Finally, users can specify conditions that will tag teams or players with custom conditions, or preset commands such as winning, losing, or disqualified.