Cards Against Humanity Online
67-328 Mobile to Cloud
12/10/14

Ian Go
NOTE:RUN LOCALSERVER ON PORT 8000

OpenShift link: cardsagainsthumanity-iango.rhcloud.com
Github link: https://github.com/ianpgo/cardsagainsthumanity

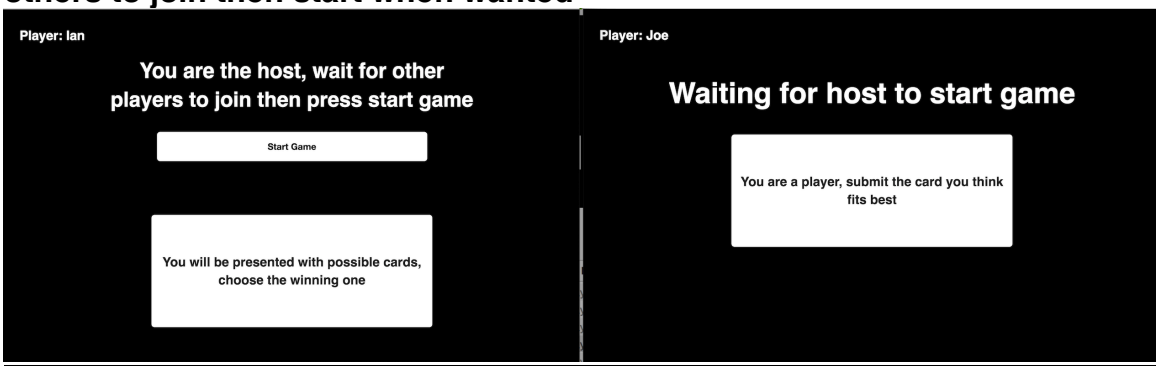
1. Implement a distributed application that does something of value

My application allows people to play the game Cards Against Humanity virtually, using any web device. The game is a somewhat offensive version of Apples to Apples. A host draws a question card that needs to be answered by the other players. The players answer by submitting from the hand of cards drawn from a deck. The host picks the winning card and the host changes. Normally it is a casual game played for laughs (hence why I don't keep score!).

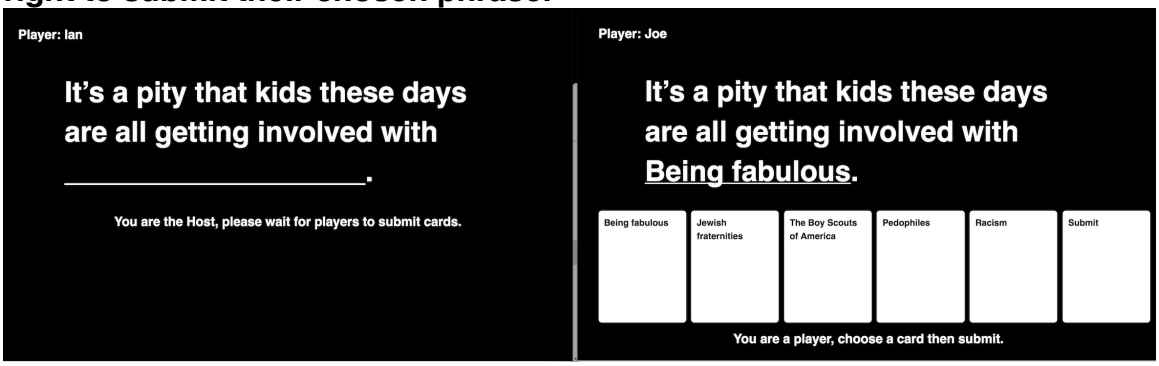
I'll walk you through one round of gameplay

The user first enters their name and joins the game

The first player to join is the host who will start the game. They will wait for others to join then start when wanted

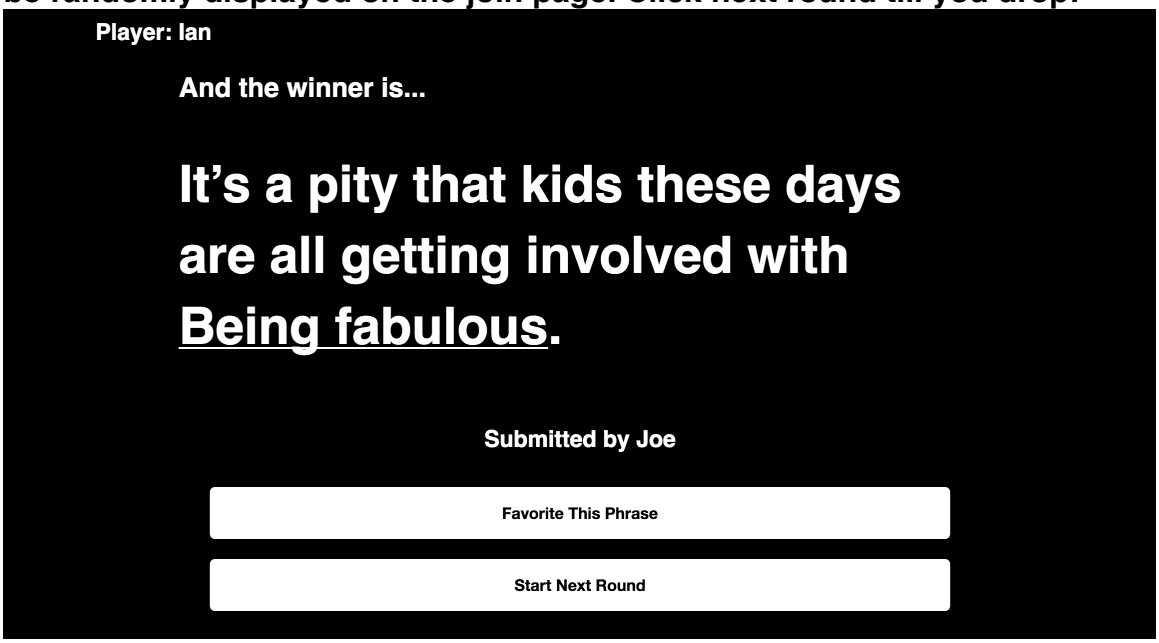


The players see what the question is. The host waits for all players to submit their cards. Players get a view of their current hand and can apply them to the phrase as they see fit. They press the submit card on the far right to submit their chosen phrase.



The host then gets to see all the submitted cards and choose their favorite

The winning card is picked. Winning phrases can be favorited and will then be randomly displayed on the join page. Click next round till you drop!



This walkthrough is shown for only two players, but the game can be played with many more!

2. Update the content displayed on interaction with the server

My server handles most of the logic of cards being moved around and which view is seen. The server draws the players hands and new question cards. This card information is sent to the players for them to see.

The server also tells the players if they are the host, letting them load the view suitably

3. Use Ajax style interaction or **Websockets**

Cards Against Humanity Online uses sockets.io to implement Websockets. Events handled, and messages are sent between clients and the server to make the game interaction happen!

4. The Application interacts live with other users

Players interact with each other by sending cards and judging answers.
This is all done live by juggling the sockets and states of each player.

5. Demonstrate good separation of concerns with MVC

Models

I have two models. One handling the database, the other a player object.

/models/mymongo.js
-This model handles all the mongoDB logic, it connects the app to a local or online mongo storage. It includes insert, find, findall, and update methods to interact with the database.

/models/Player.js
-This model handles the logic behind a single player object. A player object has a username, socketID, hand of cards, and a Boolean of host. Along with this a player has methods that deal with manipulating their hand and state as a host

```
//Class for one player

function Player(username, socketID, hand, host){
    this.username = username;
    this.socketID = socketID;
    this.hand = hand;
    this.host = host;
}

//add one card to player's hand (used on redraw)
Player.prototype.addCard = function(card){
    this.hand.push(card);
}

//pull submitted card from players hand
Player.prototype.submitCard = function(card) {
    var index = this.hand.indexOf(card);
    return this.hand.splice(index, 1);
};

//remove card from players hand
Player.prototype.removeCard = function(card){
    var index = this.hand.indexOf(card);
    this.hand.splice(index, 1);
};

//fill player's hand from cards
Player.prototype.fillHand = function(fillerHand){
    fillerHand.forEach(function(card){
        this.hand.push(card);
    });
}

//make the player the current host
Player.prototype.makeHost = function(){
    this.host = true;
}

//make the player a nonHost player
Player.prototype.unHost = function(){
    this.host = false;
}
```

Views
My views are told by the serverSockets and clientSockets files when to change and what data to show. Check out my separation of HTML, JS, and CSS for more about views!

Routes
/routes/serverSocket.js
-The serverside routing file that handles a lot of routing logic. It uses the models to pull and handle data logic, using this it emits and receives events to and from the users. This all flows out the to views, changing view and taking input form them.

6. Use Local storage to manage web application state info

On the client-side, each players stores their name, hand, host, and current question. If the player is a host they also store the current cards submitted by each player. This is through a series of variables and arrays.

```
var playerName; //playername
var hand=[]; //the players hand
var host=false; //boolean if player is hosts
var question=""; //stores the current question card
var editQuestion=""; //stores a string formatted version of question card
var selectedcards; //var of submitted cards
```

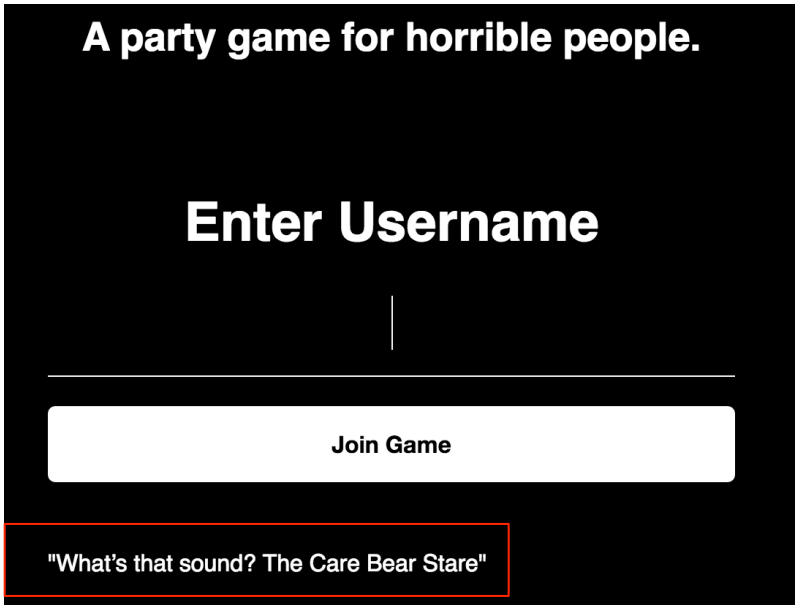
On the server-side the state of the deck of answer and question cards is handled. The decks will reshuffle if they are near depletion. An array of player objects is also stored

```
var currentPlayers = 0; //number of current players in game
var round = 0; //number of rounds played in game
var hostID; //socketID of current host
var players = []; //array of player objects

var questionDeck = []; //empty array for deck of question cards
var answerDeck = []; //empty array for deck of answer cards
var selectionDeck=[]; //deck of player selected cards with players mapped
```

7. Store data persistently using a server-side database

Along with storing the card decks in mongoDB, the application stores favorite phrases chosen by players. When the winning phrase is shown, any player can save it. This will store it server-side in the ‘favorites’ collection.
Favorite phrases are randomly shown on the join game page.



db of favorite phrases

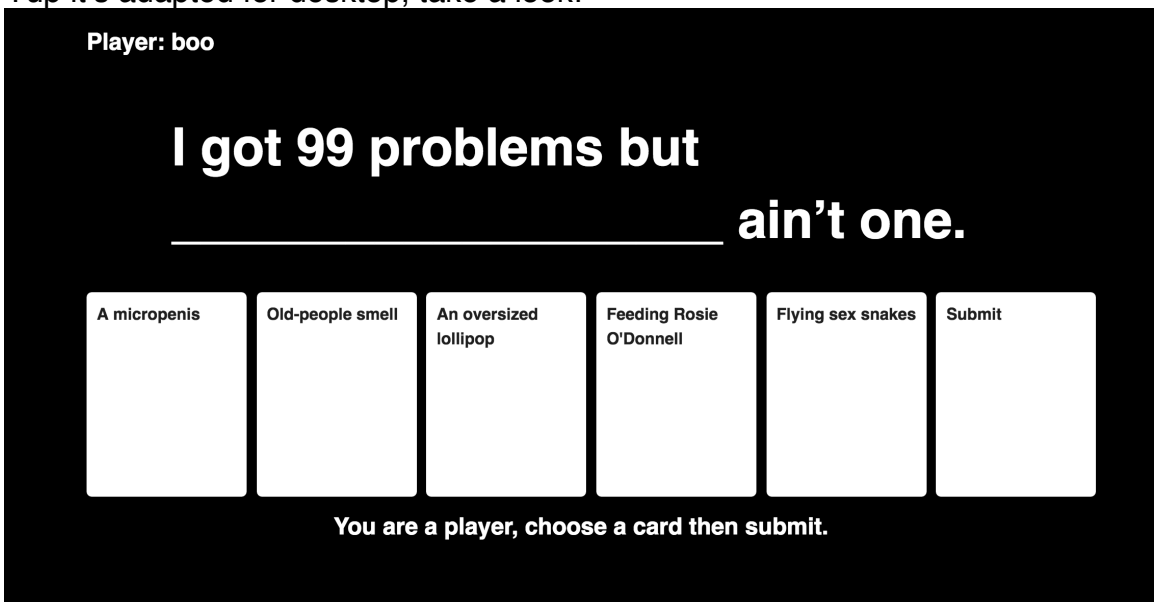
```
> db.favorites.find();
{ "phrase" : "What’s that sound? Sexy pillow fights", "_id" : ObjectId("5488f07096fcb5000053235c") }
{ "phrase" : "Alternative medicine is now embracing the curative powers of A tiny horse.", "_id" : O
{ "phrase" : "Why can’t I sleep at night? What’s that smell?Peeing a little bit", "_id" : ObjectId("
{ "phrase" : "What’s that sound? The Care Bear Stare", "_id" : ObjectId("5488fa9ac3c14600001aaa18") }
```

db of answer cards

```
> db.whitecards.find();
{ "_id" : ObjectId("5485cb6fdfc5616fce482304"), "answer" : "Being on Fire" }
{ "_id" : ObjectId("5485cb6fdfc5616fce482305"), "answer" : "Racism" }
{ "_id" : ObjectId("5485cb6fdfc5616fce482306"), "answer" : "Old-people smell" }
{ "_id" : ObjectId("5485cb6fdfc5616fce482307"), "answer" : "A micropenis" }
{ "_id" : ObjectId("5485cb6fdfc5616fce482308"), "answer" : "Women in yogurt commercials" }
{ "_id" : ObjectId("5485cb6fdfc5616fce482309"), "answer" : "Classist undertones" }
{ "_id" : ObjectId("5485cb6fdfc5616fce48230a"), "answer" : "Not giving a shit about the Third World" }
```

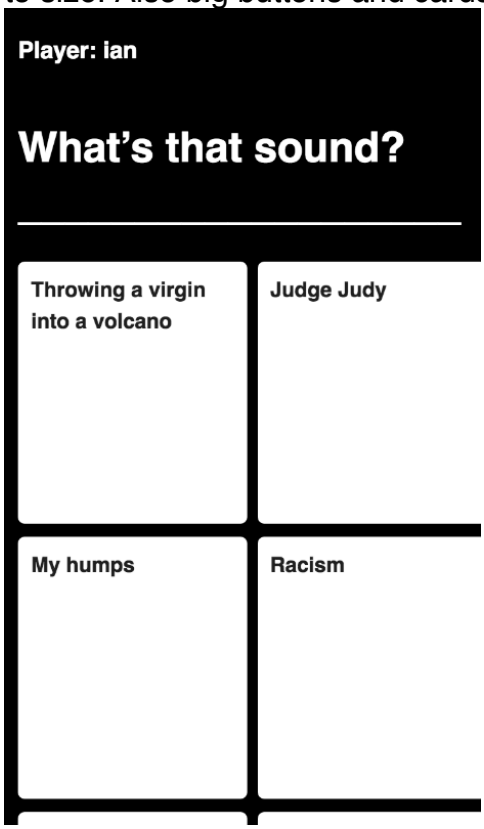
8. Adapt for access by desktop

Yup it's adapted for desktop, take a look!



9. Adapt for access by mobile

Yup, using foundation I made an adaptable framework. Cards and divs adjust to size. Also big buttons and cards are awesome for mobile button presses!



10. Work when deployed in the cloud

Deployed to OpenShift with mongoDB hooked up.

cardsagainsthumanity-iango.rhcloud.com

11. Demonstrate good separation of JavaScript, CSS, HTML

Yup. Here is the breakdown

HTML
Index.html
-Is the main view for the app. Mostly a collection of divs that manipulated according to the users view.

CSS
Foundation.css & normalize.css
-Frameworks I used for grids and responsiveness

App.css
-My edits on the grid to create the Cards of Humanity style I was going for

JavaScript
Plugins
-My app includes Jquery plugin and foundation plugins

clientApp.js
-js file including the animations of the app. I separated this from my clientsSocket.js file because I wanted didn't want any events in the way. Purely stylistic stuff like making a submit card black on hover, filling the answer field, making stuff blink, and making cards move on hover as well!

clientSocket.js
-js file that handles all the socket events on the client side. Includes some local storage and lots of event handling.

12. Make appropriate choices of HTTP methods

I used sockets.io so I don't have much HTTP methods. Even my mongo pulls are handled through sockets. There are a lot of player and server interactions so it made sense this way. You can check out my sparse server.js file, and loaded serverSocket file if you wish!

13. Demonstrate good coding and commenting

I tried to separate concerns, have helper functions, and comment a ton! Check out my code if you don't believe me.

Bonus: Application is fully published to GitHub, with LICENSE and README.md
https://github.com/ianpgo/cardsagainsthumanity

Thank you and goodnight!