DESIGN REPORT #5

PARTICIPATORY AFFORDANCES - MASS CONTROL

#SOCKETS, #STORYBOARDS, #EMBODIED INTERACTIONS

Elie Zananiri, *Big Screams*

[Inara Rupani]

This is an individual assignment.

[CLOUD 9: https://us-east-2.console.aws.amazon.com/cloud9/ide/7c94b0cd46cc41dabe433772de24dd4a]
[Google Doc: https://docs.google.com/document/d/1CYblH4iqda37EXr4MApXXIK7YKKJL2ZX-Z4JVtuA_X8/edit?usp=sharing]

1 DEADLINES

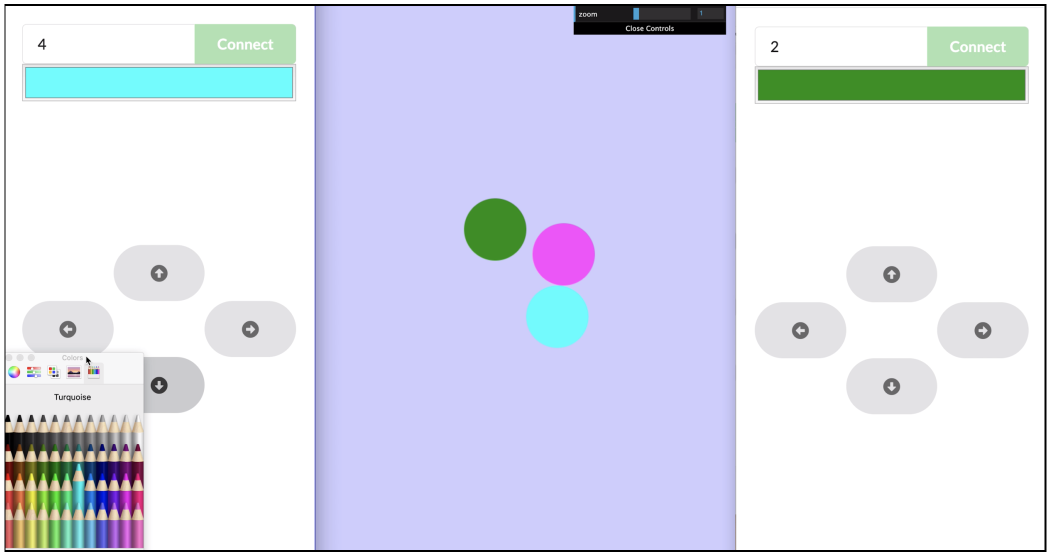
Module Release	April 13
Module Due Date	May 1, End of Day to Canvas
Module Exam	May 1 - May 3

2 TEACHING OBJECTIVES

- Working knowledge of Websockets
- Create an interactive multi-user interface
- Apply and appraise **participatory affordances** in UI design.

3 DESIGN BRIEF (100 PTS TOTAL)

In this assignment, you will develop an interaction that transcends the one device/one person paradigm.



1. **Technical Vignettes (50 pts) -** We will begin by working through some fundamentals around:
   1.1. How to establish a persistent connection with multiple devices through a **multicast websocket server**
   1.2. How to direct communication from one device to another through **event triggers and handling**
   1.3. View the target interaction by clicking the video linked to the app screenshot above.

2. **Design (50 pts) -** You will then propose an interaction through a role prototype -- a storyboard.
   2.1. A websocket-enabled billboard has been installed in your neighborhood. Design a multi-user interaction that maximizes participatory affordances. Communicate this interaction via a storyboard.

4 SETUP

1. Using your current **EC2 Cloud 9** instance, navigate to the Terminal.
    a. In cloud9 Terminal, run:
        i. git clone https://github.com/CSE3392-S2020/module5-mass-control.git
        ii. cd module5-mass-control
        iii. bundle install
    b. In the run configuration within cloud9 (the part that has the rails server command), click on the CWD button and point the current working directory to module5-mass-control.
        i. Change the run config from **rails server** to **rails server -b 0.0.0.0 -p 3000**

2. **Important: For this assignment, be sure to point your browser to http instead of https.**
3. **Each time you reload your Cloud9 (from an inactive state), your IP address will change.**
4. Follow the Cloud9 documentation for exposing port 3000 in your EC2 instance.
    a. Video: https://youtu.be/pUyDViSzp38

```
1. Startup your Cloud9 instance
2. IP: 18.220.156.213 (changes every time you startup)
3. EC2 Manager > EC2 Instance
   Security Groups
     Inbound Rules: Add rules, 3000, Anywhere
4. EC2 Manager> EC2 Instance > Subnet ID
      Network ACL
      Edit inbound rules
      200, Custom TCP, 3000, ALLOW
5. Start server bound to port 3000
   rails server -p 3000 -b 0.0.0.0

Open browser to:
http://<YOUR_IP>:3000
```

5 HELLO PHONE - MOBILE DEBUGGING (5 PTS)

Protocol	**http://**
View	**phone/hello**
Run	**rails s -b 0.0.0.0 -p 3000**

Configure your phone for debugging:

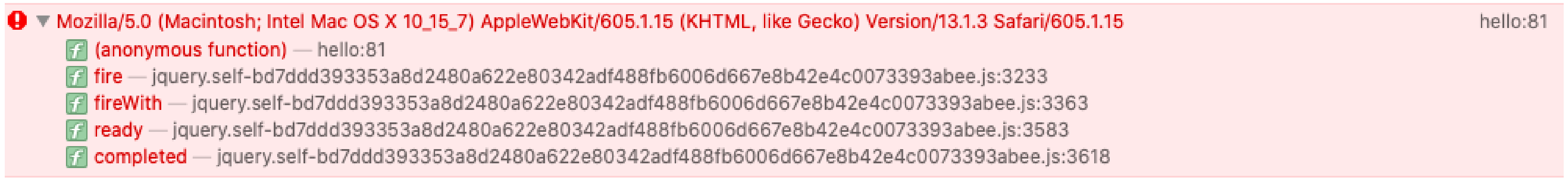
- Safari & iPhone: https://medium.com/@mattcroak718/debugging-your-iphone-mobile-web-app-using-safari-development-tools-71240657c487
- Chrome & Android: https://developers.google.com/web/tools/chrome-devtools/remote-debugging

Depending on your phone OS, you'll need to use the appropriate browser debug tools (see above).

Point your phone browser to <AWS_IP_ADDRESS>:3000/phone/hello
Tip: You can navigate from your browser developer tools by typing window.location = "http://YOUR_IP/phone/hello" into the console.

[TODO: Copy and paste the console message from the mobile debug screen.]

> Mozilla/5.0 (iPhone; CPU iPhone OS 14_4 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0.3 Mobile/15E148 Safari/604.1

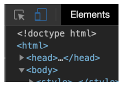


```
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.1.3 Safari/605.1.15    hello:81
    (anonymous function) — hello:81
    fire — jquery.self-bd7ddd393353a8d2480a622e80342adf488fb6006d667e8b42e4c0073393abee.js:3233
    fireWith — jquery.self-bd7ddd393353a8d2480a622e80342adf488fb6006d667e8b42e4c0073393abee.js:3363
    ready — jquery.self-bd7ddd393353a8d2480a622e80342adf488fb6006d667e8b42e4c0073393abee.js:3583
    completed — jquery.self-bd7ddd393353a8d2480a622e80342adf488fb6006d667e8b42e4c0073393abee.js:3618
```

6 PHONE REMOTE BINDING (10 PTS)

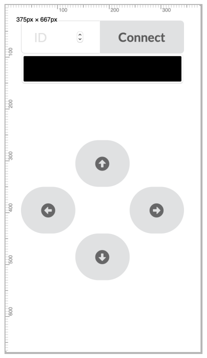
Protocol	**http://**
View	**phone/remote**
Run	**rails s -b 0.0.0.0 -p 3000**

In your browser, configure your setup for mobile development

- Safari: Develop > Enter responsive design mode
- Chrome: Enable Device Toolbar.
    - 

```
<!doctype html>
<html>
 ▶<head>…</head>
 ▼<body>
```

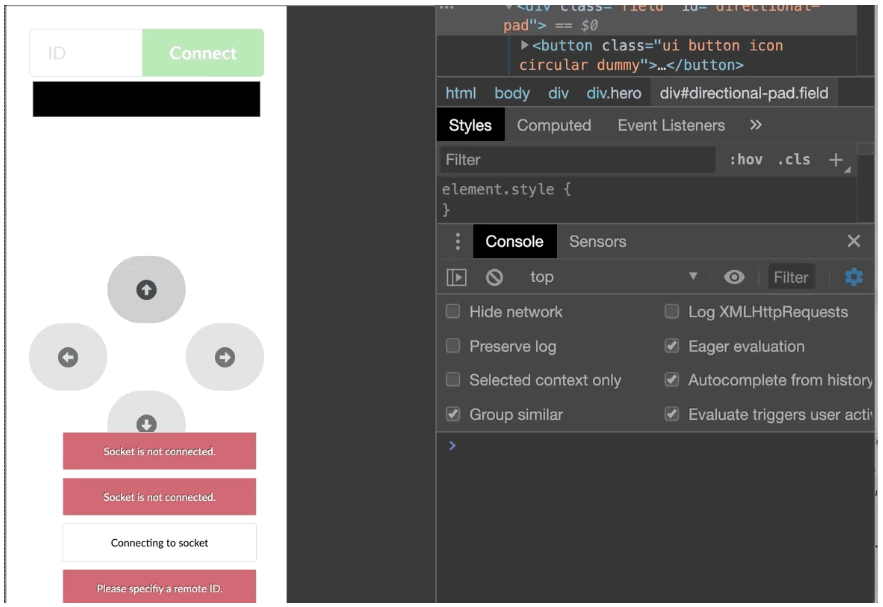
Your screen should look like below:



Object	**Event**	**Behavior**
Connect button	click	Alertify "Connecting to socket"; Add a green and disabled class to the button. Set the global window.socket variable to true.
Directional pad buttons	click	1. **Create** a request object = {} a. Insert key "action" with value "move" b. Insert a key "direction" with value equal to the direction that was clicked e.g., "up" c. Insert a key "color_id" with value equal to the currently selected color 2. **Validate**

		a. Socket object exists i. If not, alertify -- “Socket does not exist. Connect first” ii. return b. remote_id is not a null string i. If not, alertify -- “Please specify a remote ID” ii. Return 3. **Send** the request a. Stringify the object with JSON.stringify i. Add prefix “Server <<”
Note: alertify is a helper UI for debugging. You can pass it a text to display via its .notify or .error methods.		

Expected behavior below…



```
pad"> == $0
  ▶<button class="ui button icon
  circular dummy">…</button>
html  body  div  div.hero  div#directional-pad.field
element.style {
}
```

Socket is not connected.

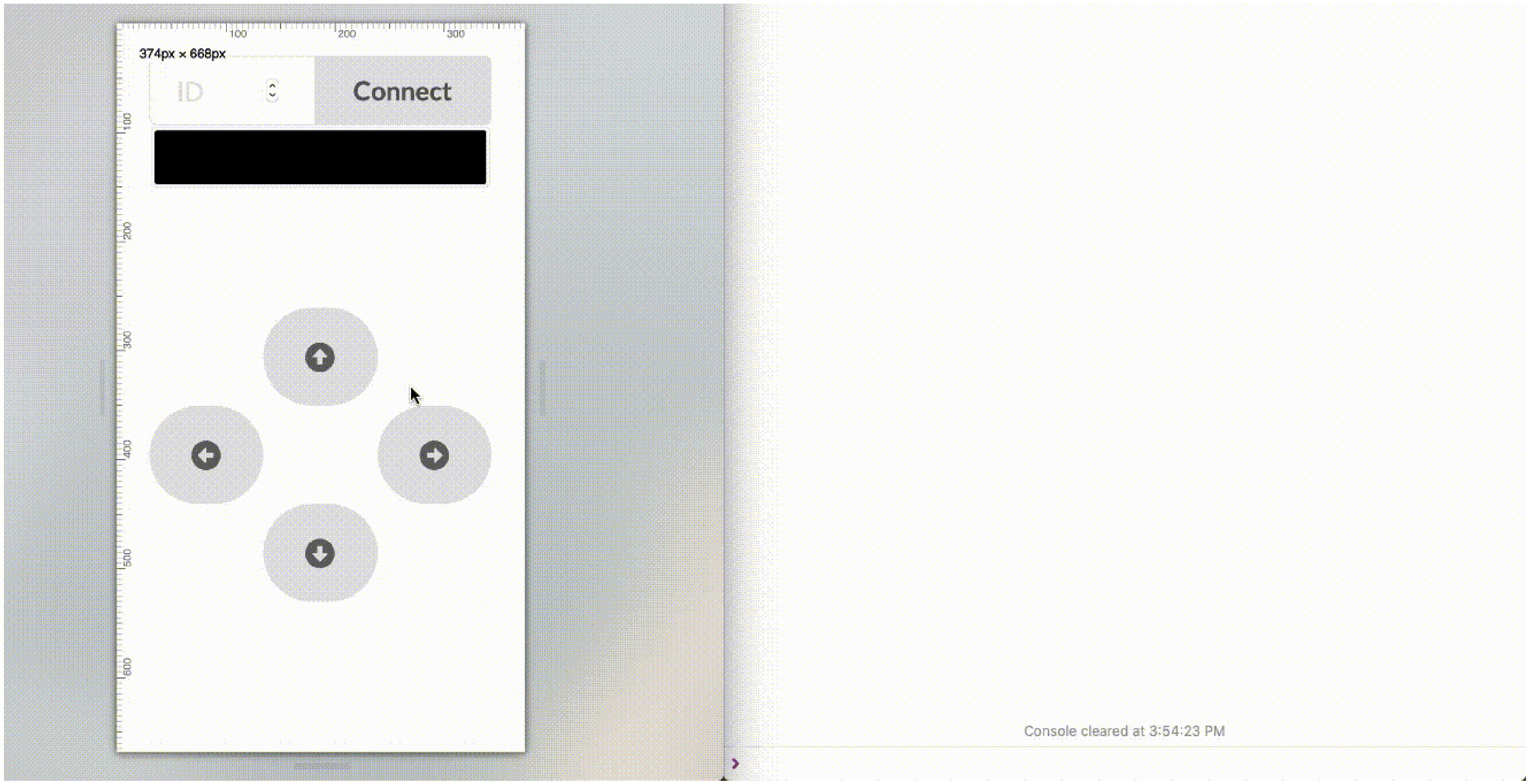
Socket is not connected.

Connecting to socket

Please specify a remote ID.

SUBMISSION

[TODO: Insert GIF]
GIPHY: https://media.giphy.com/media/Ttk63oC4F2EouUKAOE/giphy.gif



```
Console cleared at 3:54:23 PM
```

Record a GIF (recommended GIPHY) with the following actions:

1. Click **UP** button
2. Click **CONNECT** button
3. Click **UP** button
4. Type in ID = 4
5. Click **UP** button
6. Type in ID = 6
7. Click **LEFT, DOWN, RIGHT** button
8. Change the color to non-black
9. Click **RIGHT** button

```
[TODO: Paste your Code from phones/remote - Please format using CodeBlocks]
 $ ->
    alertify.notify "Hello World"
    $("form").submit (event)->
      event.preventDefault()

    # Button Handlers
    $('#connect').on 'click',(event) ->
      console.log('connect button was clicked')
      alertify.notify "Connecting to socket"
      $(this).addClass("disabled")
      $(this).addClass("green")
      window.socket = true

    $('#directional-pad button').on 'click', (event) ->
      requestObj =
        action: "move"
        remote_id: $('[name=remote_id]').val()
        color_id: $('[name=color_id]').val()
        direction: $(this).attr("name")
      #console.log('direction pad is used', $(this).attr("name")
      #validation
      if window.socket == undefined
        alertify.error "Socket does not exist. Connect first"
      if $('[name=remote_id]').val() == null or $('[name=remote_id]').val() == ""
        alertify.notify "Please specify a remote ID"
      else
      #send the object
        ClientReq = JSON.stringify(requestObj)
        RequestString = 'Server << ' + ClientReq
        console.log(RequestString)
```

Check ++ Recommendations:

- Display the dot’s number (using paper.js PointText) on the canvas.
- Add a string input field to alllow users to name their dots.

6 PHONE SOCKET INTEGRATION (10 PTS)

Protocol	**http://**
View	**phone/remote**
Run	**rails s -b 0.0.0.0 -p 3000**

IMPORTANT: BE SURE YOUR BROWSER IS USING **HTTP, NOT HTTPS**
Within a start_socket function:

Create a new WebSocket object variable with websocket address "ws://162.243.120.86:3010". Bind the following events.

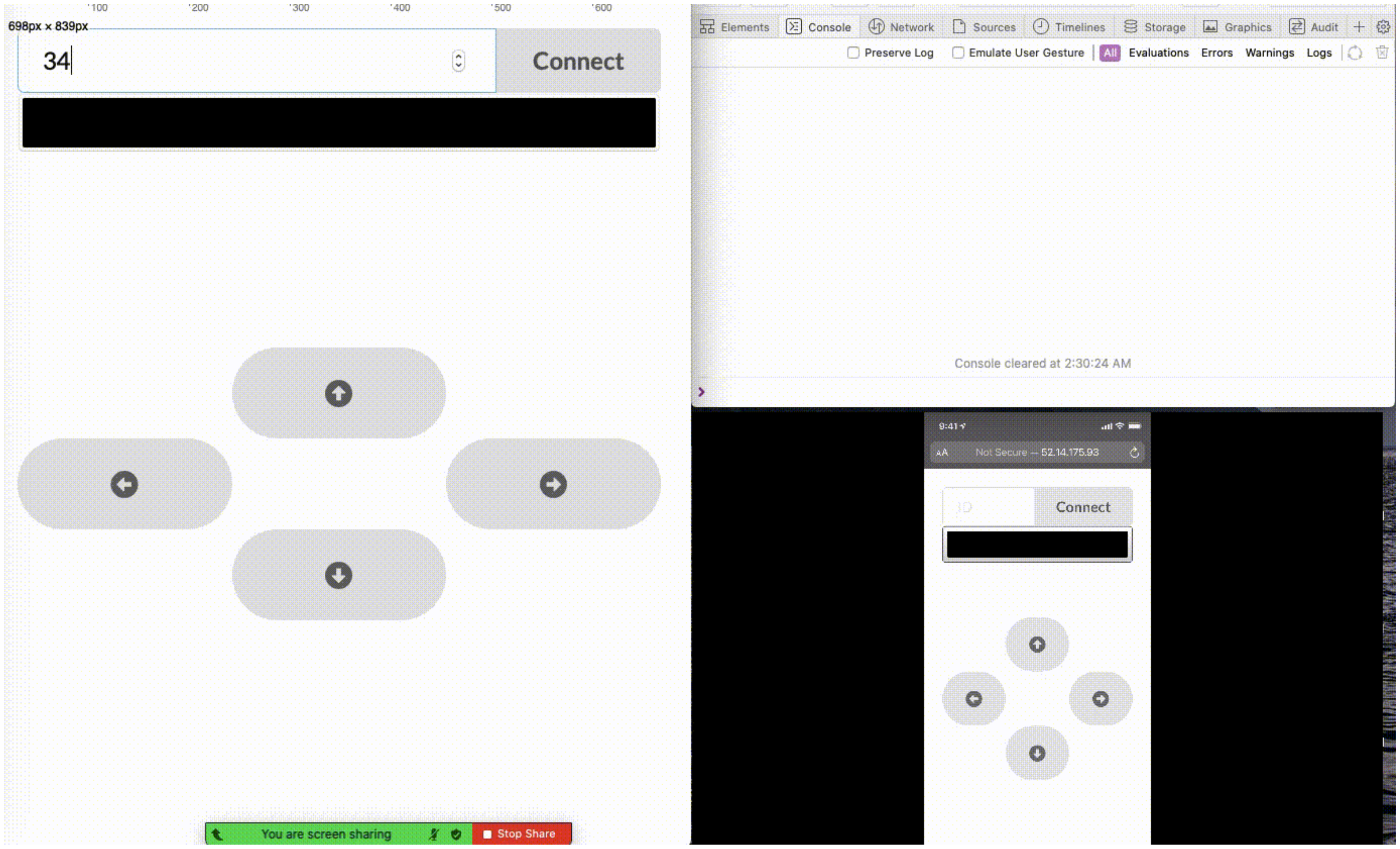
Object	**Event**	**Behavior**
socket	onopen	Add a **disabled** and **green** class to the connect button, remove any **red** class.
socket	onclose	Remove any **disabled** and **green** classes from the connect button, add a **red** class.
socket	onmessage	Parse the event data into a JSON object (JSON.parse). Console log the resulting object. Add a "Server >>" prefix.
socket	onerror	Alertify the error message.

Return the socket object.

Adjust the Part 5 code to:

1. Call start_socket when the connect button is pressed.
2. Call socket.send to send the JSON request.
3. Update socket validation to also include checking to see if the socket object is open
   a. socket's readyState == WebSocket.OPEN.

[TODO: GIF showing communication going back and forth.]
GIPHY: https://gph.is/g/4z1mejb

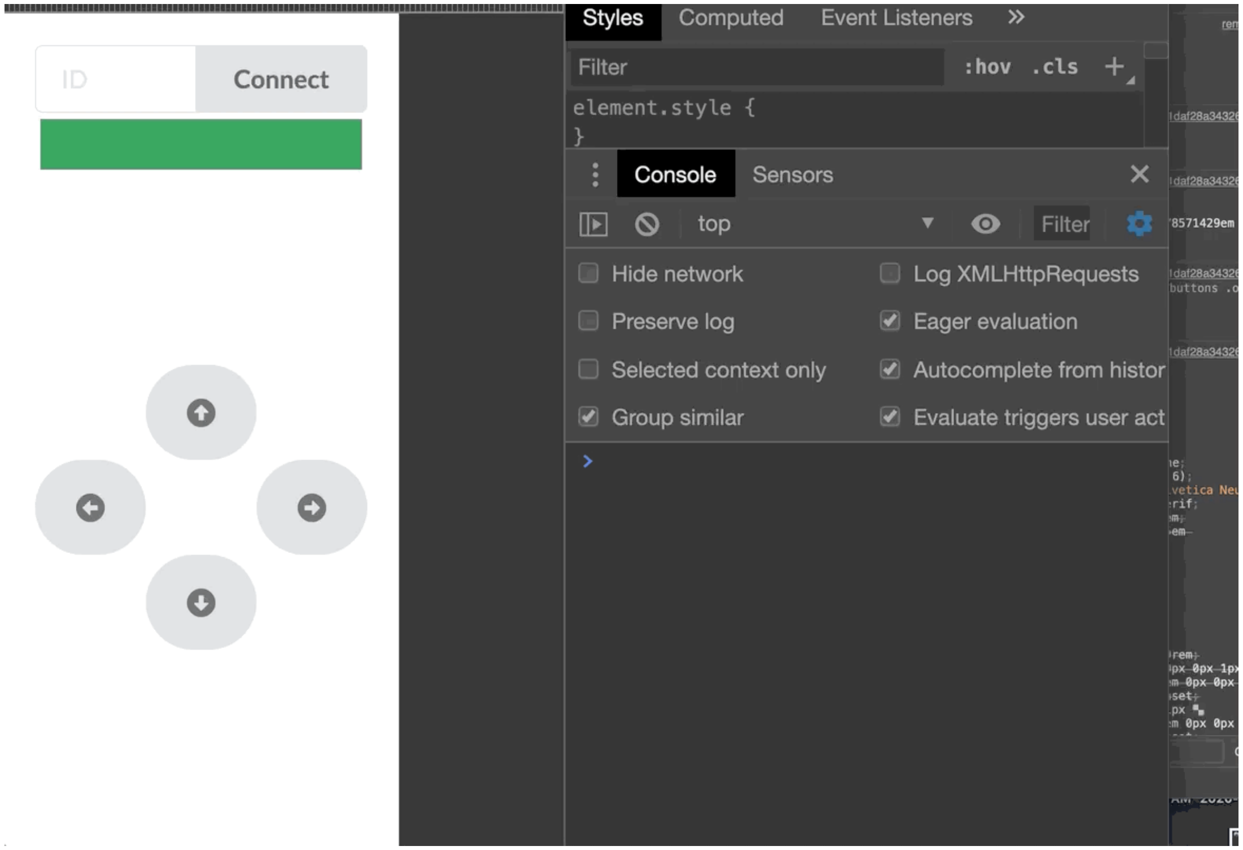


In this gif, I am recording my Desktop Safari and my phone's socket messages. The socket messages sent are seen on the top right side where the desktop Inspect is found. We also see that other people who also connect to AWS are seen on my inspect element

```
:coffeescript
  #aws_wss = "ws://162.243.120.86:3010"
  aws_wss = "ws://162.243.120.86:3010"
  window.socket = undefined;
  # socketConnection = new WebSocket(aws_wss, ["http"])

  start_socket = () ->
    socketConnection = new WebSocket(aws_wss, ["http"])
    console.log("ATTEMPTING CONNECTION ON ws://162.243.120.86:3010")

    socketConnection.onopen = (event) ->
    # $('#connect').on 'click',(event) ->
      alertify.notify "Connecting to socket"
      $('#connect').addClass("disabled")
      $('#connect').addClass("green")
      $('#connect').removeClass("red")
        # window.socket = true

    socketConnection.onclose = (event) ->
      # $('#connect').on 'click',(event) ->
      alertify.notify "Leaving the socket"
      $('#connect').removeClass("disabled")
      $('#connect').removeClass("green")
      $('#connect').addClass("red")

    socketConnection.onmessage = (event) ->
      data = JSON.parse(event.data)
      console.log("Server >>", data)

    socketConnection.onerror = (event) ->
      alertify.error "Error"

    return socketConnection
```

```
$ ->

  alertify.notify "Hello World"
  $("form").submit (event)->
    event.preventDefault()

  # start_socket function enabled as connect button is pressed
  $('#connect').on 'click', (event) ->
    window.socket = start_socket()
    console.log ("WEBSOCKET OPEN")

  $('#directional-pad button').on 'click', (event) ->
    requestObj =
      action: "move"
      remote_id: $('[name=remote_id]').val()
      color_id: $('[name=color_id]').val()
      direction: $(this).attr("name")
    #console.log('direction pad is used', $(this).attr("name")

    #validation
    if window.socket == undefined or window.socket.readyState != WebSocket.OPEN
      alertify.error "Socket does not exist. Connect first"
    if $('[name=remote_id]').val() == null or $('[name=remote_id]').val() == ""
      alertify.notify "Please specify a remote ID"
    else
    #send the object
      ClientReq = JSON.stringify(requestObj)
      RequestString = 'Server << ' + ClientReq
      console.log(RequestString)
      window.socket.send(ClientReq)
```

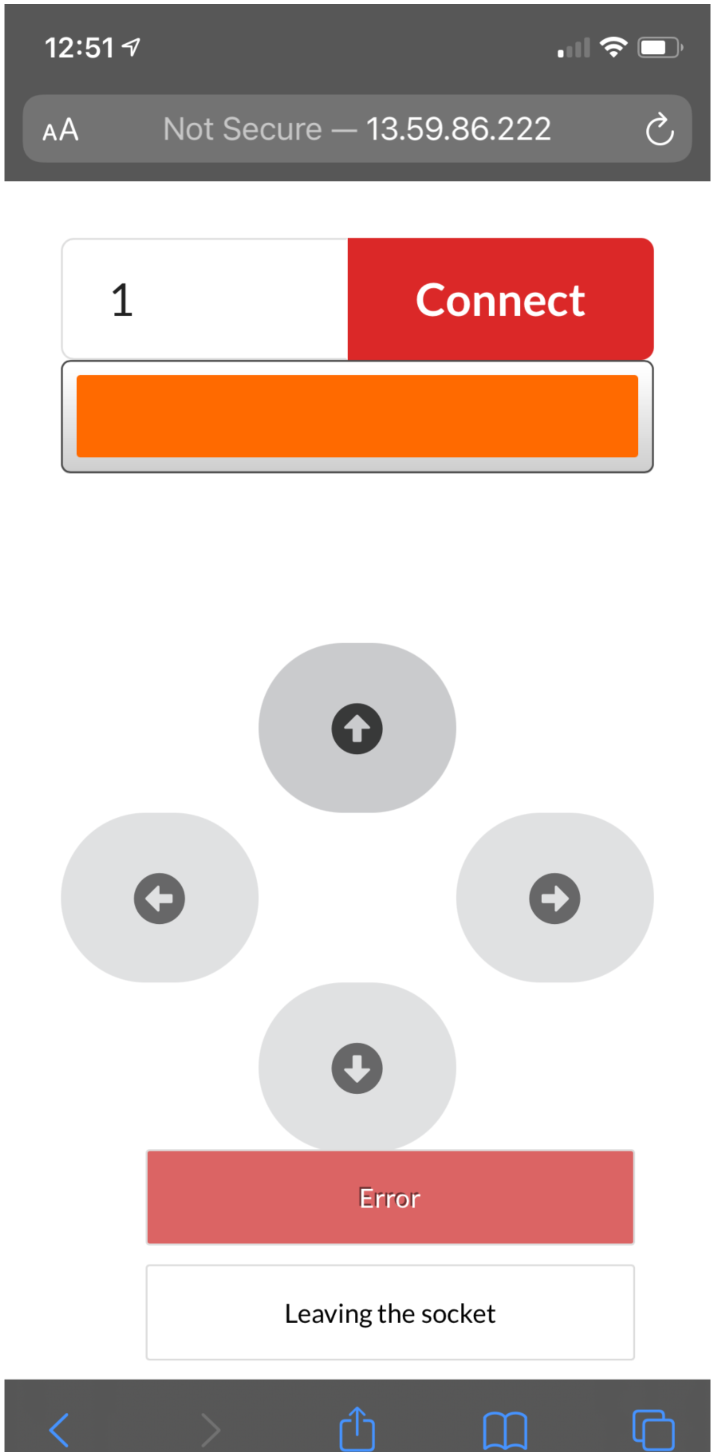
Example output:



```
element.style {
}
```

- [ ] Hide network
- [ ] Log XMLHttpRequests
- [ ] Preserve log
- [✓] Eager evaluation
- [ ] Selected context only
- [✓] Autocomplete from histor
- [✓] Group similar
- [✓] Evaluate triggers user act

#After a long period of inactivity the socket closes and below is the screenshot of how the UI looks like with a red button. (Next page)

	#After a long period of inactivity the socket closes and below is the screenshot of how the UI looks like with a red button. (Next page)

Check ++ Recommendations:

- Give each dot a speed, controlled by the user, that allows the dot to continuously move.
- Detect collisions with other dots and add some indication that a collision had occurred.
- Introduce a game mechanic and keep a leader scoreboard.

7 MOVING DOT (10 PTS)

Protocol	**http://**
View	**board/index**
Run	**rails s -b 0.0.0.0 -p 3000**
Websocket Address	**ws://162.243.120.86:3010**

1. Copy and paste your start_socket function from **Part 6**.
    a. Adjust the onmessage behavior
        i. If the message has an action, trigger that action on the document object. (jquery trigger)
            1. Within the event trigger, add the message as an extra parameter
    b. Add a document "move" listener with **jquery on** method
        i. Query the paper environment for any circles that have remote_id == message's remote_id (paper.project.getItems)
        ii. If so, move the circles 10 pixels in the appropriate direction (message's direction)
        iii. If not, generate a new circle with the make_circle function and move the circle 10 pixels in the appropriate direction (message's direction).
        iv. Change the color of the circle to the message's color_id.
2. In the main function ($ ->), call the start_socket function.

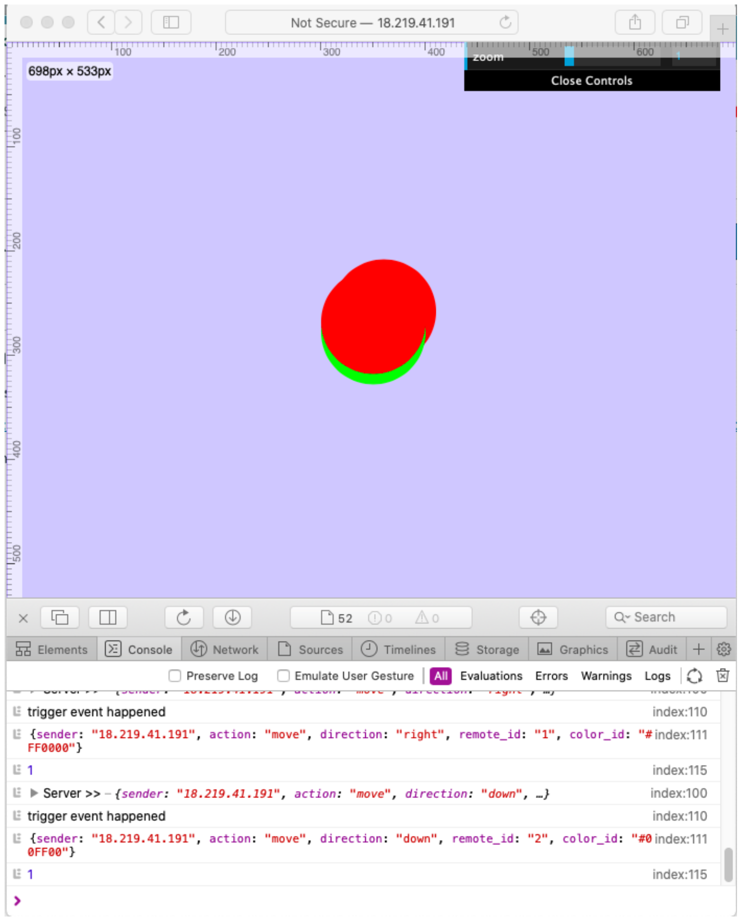
In your Cloud9 terminal, run

```
npm install -g wscat
wscat -c ws://162.243.120.86:3010
```

Once connected, send:

```
{"action":"move","direction":"up","remote_id":"1","color_id":"#00FF00"}
{"action":"move","direction":"down","remote_id":"2","color_id":"#00FF00"}
{"action":"move","direction":"up","remote_id":"3","color_id":"#0000FF"}
{"action":"move","direction":"right","remote_id":"1","color_id":"#FF0000"}
```

[TODO: Screenshot of resulting dots]

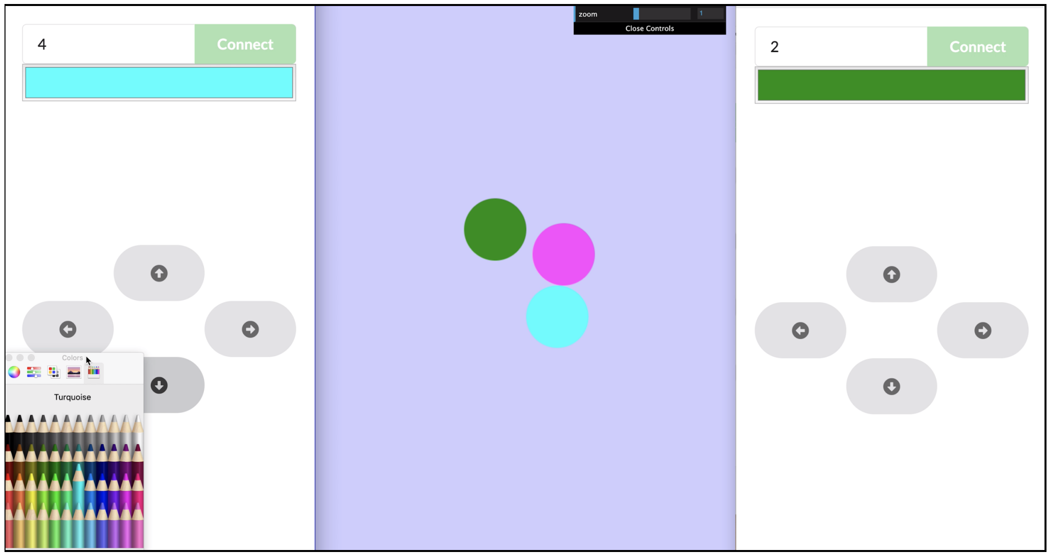


```
trigger event happened                                                                          index:110
{sender: "18.219.41.191", action: "move", direction: "right", remote_id: "1", color_id: "#      index:111
FF0000"}
1                                                                                               index:115
▶ Server >> – {sender: "18.219.41.191", action: "move", direction: "down", …}                  index:100
trigger event happened                                                                          index:110
{sender: "18.219.41.191", action: "move", direction: "down", remote_id: "2", color_id: "#0     index:111
0FF00"}
1                                                                                               index:115
>
```

8 MASS CONTROL (15 PTS)

Views	**board/index** **phone/remote**

Open two browser tabs to phone/remote, and another browser tab to board/index.
Take a GIF of your screen showing you can control the dots with your remotes.

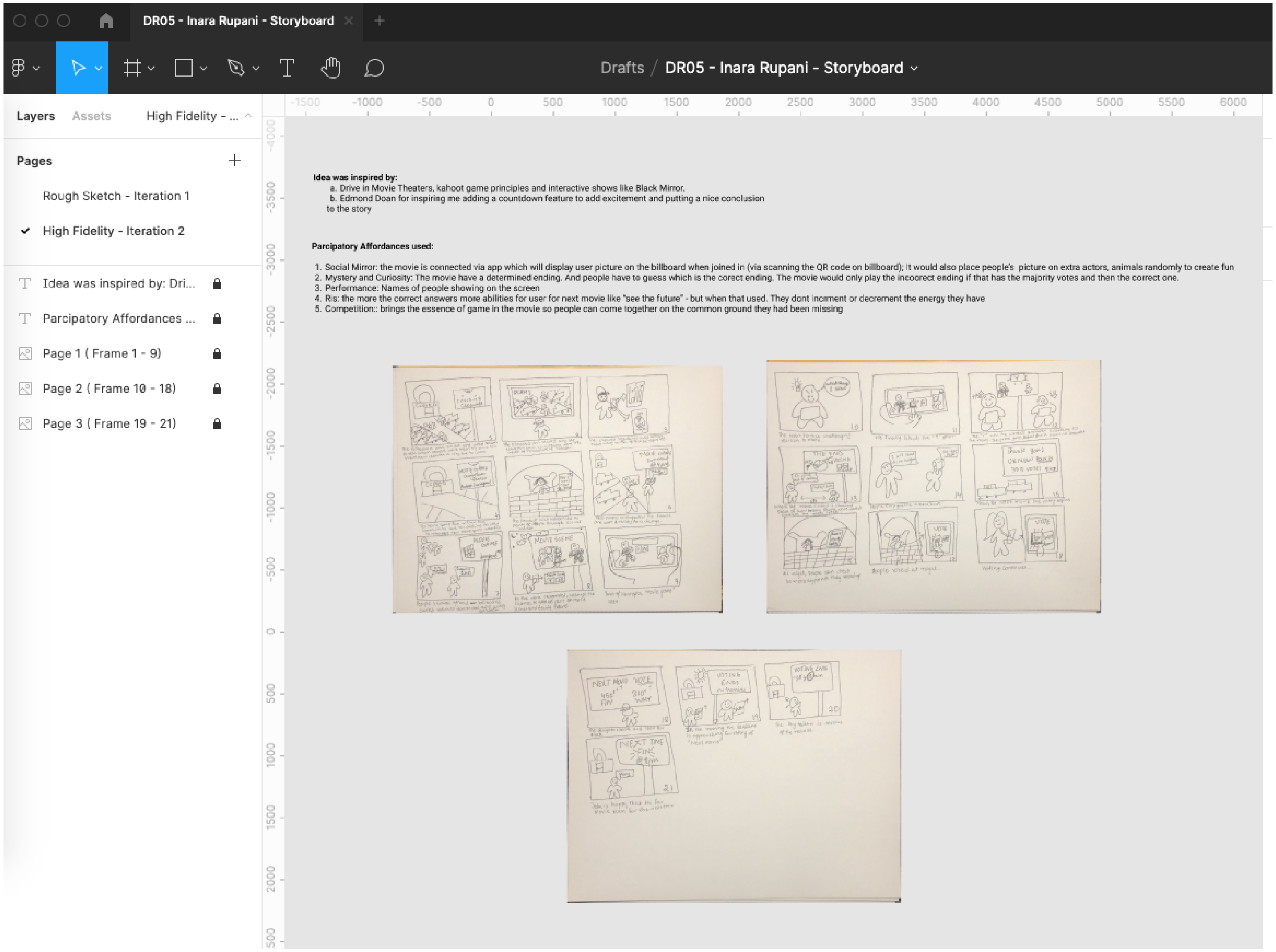


GIF
[https://media.giphy.com/media/kpKbncoEDBwCyWFFBc/giphy.gif] The second dot moving in the gif is through my phone and first i opened the tab.

9 STORYBOARD (50 PTS)

A websocket-enabled billboard has been installed in your neighborhood. Design a multi-user interaction that maximizes participatory affordances. Communicate this interaction via a storyboard.

STORYBOARD	**[FIGMA LINK]** **FIGMA: https://www.figma.com/file/tZ71IjW5cYD4lxPMNbSCrM/DR05-Inara-Rupani-Storyboard?node-id=50%3A2** **[Screenshot of FIGMA]** **-- Next Page**


DR05 - Inara Rupani - Storyboard
Drafts / DR05 - Inara Rupani - Storyboard
Layers
Assets
High Fidelity - ...
Pages
Rough Sketch - Iteration 1
✓ High Fidelity - Iteration 2
Idea was inspired by: Dri...
Parcipatory Affordances ...
Page 1 (Frame 1 - 9)
Page 2 (Frame 10 - 18)
Page 3 (Frame 19 - 21)

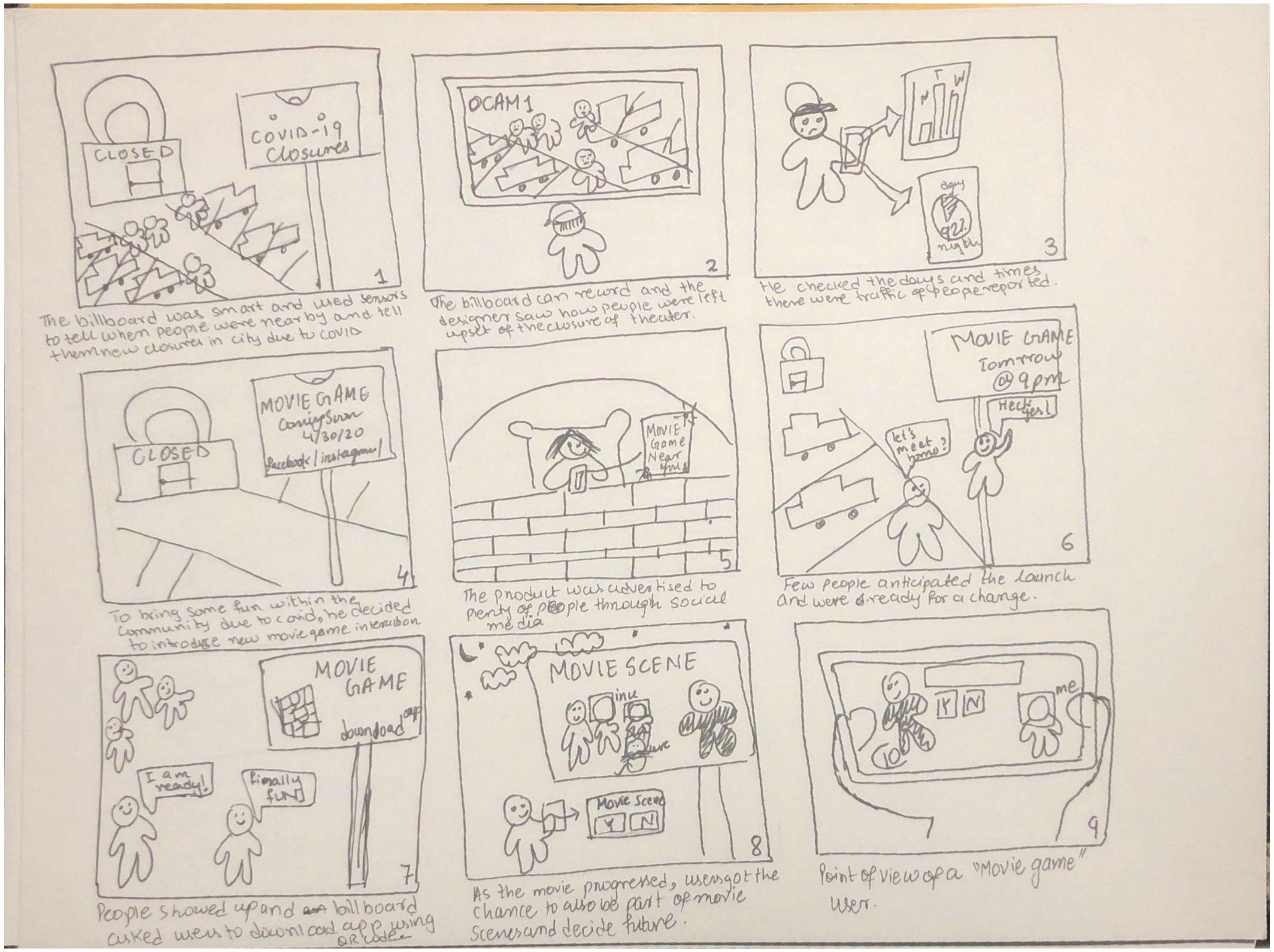
Idea was inspired by:

a. Drive in Movie Theaters, kahoot game principles and interactive shows like Black Mirror.
b. Edmond Doan for inspiring me adding a countdown feature to add excitement and putting a nice conclusion to the story

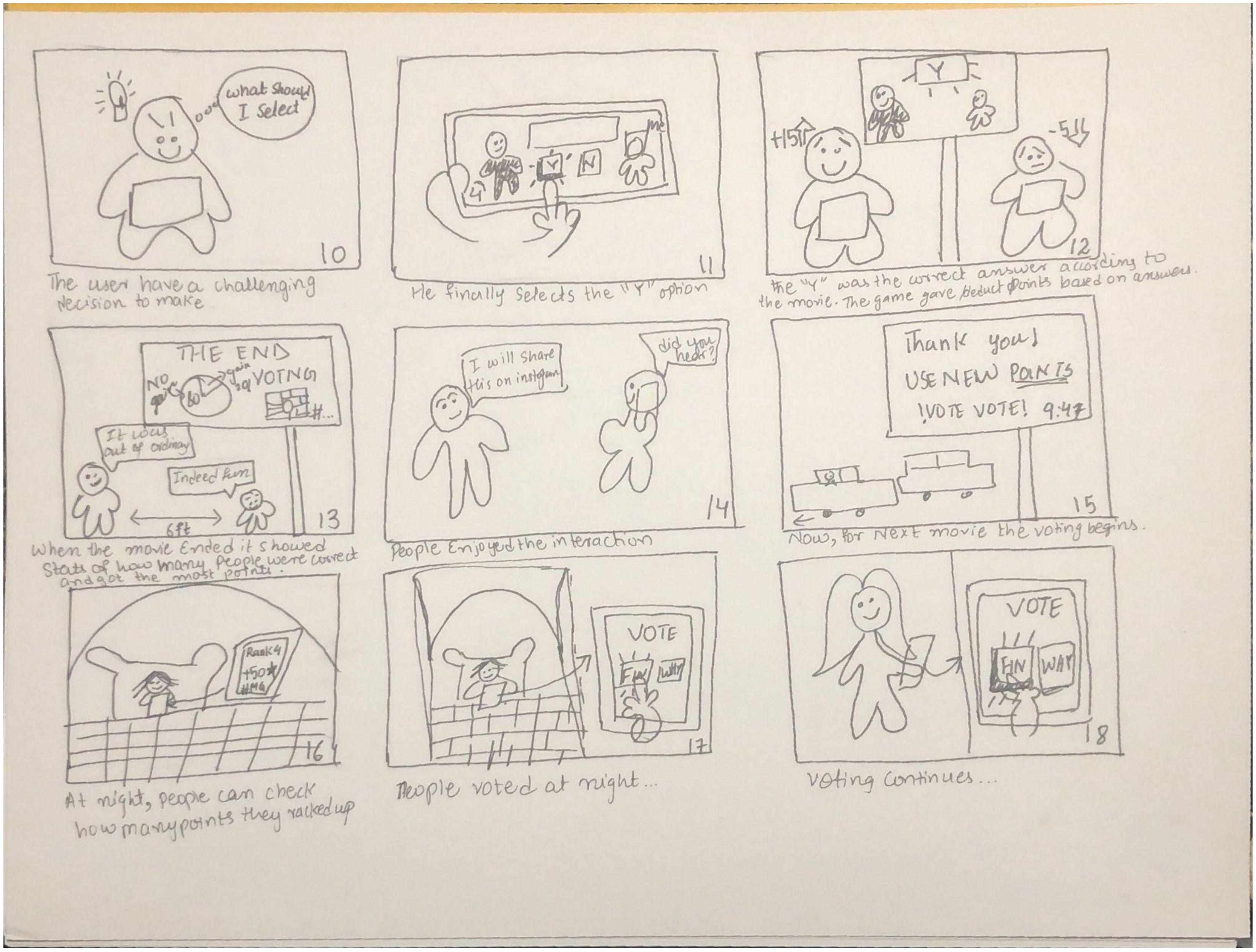
Parcipatory Affordances used:

1. Social Mirror: the movie is connected via app which will display user picture on the billboard when joined in (via scanning the QR code on billboard); It would also place people's picture on extra actors, animals randomly to create fun
2. Mystery and Curiosity: The movie have a determined ending. And people have to guess which is the corect ending. The movie would only play the incoorect ending if that has the majority votes and then the correct one.
3. Performance: Names of people showing on the screen
4. Ris: the more the correct answers more abilities for user for next movie like "see the future" - but when that used. They dont incrment or decrement the energy they have
5. Competition:: brings the essence of game in the movie so people can come together on the common ground they had been missing

[Screenshot of Final Iteration of Storyboard]



1. The billboard was smart and used sensors to tell when people were nearby and tell them new closures in city due to COVID
2. The billboard can record and the designer saw how people were left upset of the closure of theater.
3. He checked the days and times there were traffic of people reported.
4. To bring some fun within the community due to covid, he decided to introduce new movie game interaction
5. The product was advertised to plenty of people through social media
6. Few people anticipated the launch and were already for a change.
7. People showed up and ~~an~~ billboard asked users to download app using QR code.
8. As the movie progressed, users got the chance to also be part of movie scenes and decide future.
9. Point of view of a "Movie game" user.



The user have a challenging decision to make

He finally Selects the "Y" option

the "Y" was the correct answer according to the movie. The game gave /deduct points based on answers.

When the movie Ended it showed Stats of how many People were correct and got the most points.

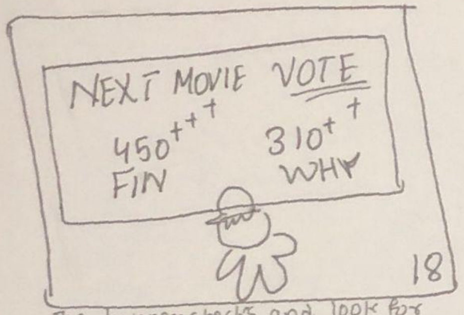
People Enjoyed the interaction

Now, for Next movie the voting begins.

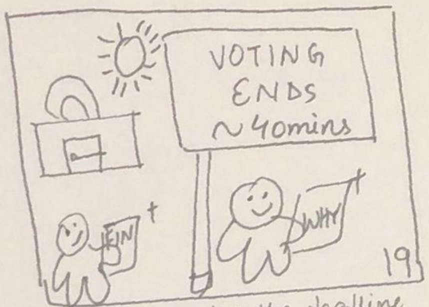
At night, people can check how many points they racked up

People voted at night...

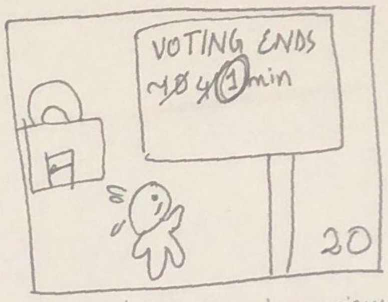
voting continues...



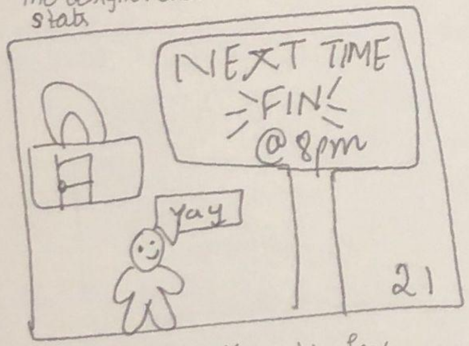
The designer checks and look for stats



~~In~~ the morning the deadline is approaching for voting of "Next movie"



The boy ~~is~~ John is anxious of the results.



John is happy that his fav movie won for the next time.

Check Criteria:

Storyboard Elements

- Setting
- Problem
- Rising Action
- Climax

Participatory Affordances

Craftsmanship

Check ++

- Iteration & Critique

**** MAKE SURE THIS DOCUMENT HAS OPEN SHARING PERMISSIONS BEFORE TURNING IN.****