CJ

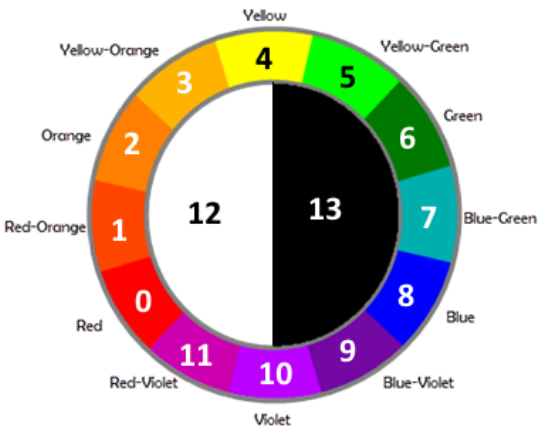
ColorPicker.js Element

+ Contains 12 colors from below color-wheel & black/white with a total of 14 different colors which the user can select.

+ When a color is clicked on the picker, it is highlighted in some way to indicate it being selected.

+ Contain a property/state "selected_color" representing the selected color. Starting at 0 for red going clock-wise on the color wheel.

Ex: 0 for red, 1 for red-orange, 2 for orange, 3 for yellow-orange, etc... all the way to 11 for red-violet. 12 for black, and 13 for white



https://www.usability.gov/how-to-and-tools/methods/color-basics.html

Hemang

Chat.js Element

+ Display chat messages (use states & map?)

+ Display textbox whose text can be submitted by pressing ENTER

+ Use effect to receive socketio emits from server:
  - "chat_update"
  - contains username & new chat message

+ Have a property/state "username" containing string with logged-in username.

+ When ENTER is pressed while textbox is selected, emit a socketio "chat_submit" containing username & message.

Note: When ENTER is pressed, the only thing that should be done is to emit a socketio "chat_submit". The messages should not be updated. This will be done when a "chat_update" emit is received.

+ Have a property/state "enabled" that when True, allows the user to select the textbox and submit messages, and when False, disables the textbox and does not allow message submission.

Naqeeb

Canvas.js Element

+ Have a property/state "data" containing pixel color for each pixel on board. Also have variables for canvas dimensions in pixels.

+ Draw pixels on a HTML canvas based on the "data" property/state.

+ When board is first loaded, emit a socketio "canvas_request" which will request canvas state from the server (wait for app.py to respond with "canvas_state" emit. show a loading circle while waiting).

+ (After sending "canvas_request") Receive socketio emit "canvas_state" which contains board settings, and the color of every pixel. Update/set "data" state w/ this information.

+ Receive socketio emits "canvas_update" which contains information on a pixel and its new color. It also contains time.

+ When user clicks on canvas (when enabled) emit a socketio message: "canvas_set" which contains information on a pixel and its new color. It also contains current time.

+ Have a property/state "enabled" that when True, allows user to click on board with a color selected to modify the board. When False, prevents any modification of the canvas and will not emit any changes/actions.

Elijah

app.py

Handles server & database stuff

+ Function that receives socketio emit: "chat_submit". Receives username/message and emits "chat_update" containing username and message.

+ Function that receives socketio emit: "canvas_request". Uses CanvasState.py to obtain canvas pixel data and sends that information with socketio emit "canvas_state" containing all pixel data as byte array, canvas dimensions and current time.

+ Function that receives socketio emit: "canvas_set" which contains information of pixel, color, and time. Emits a socketio "canvas_update" containing pixel,color, and time, and also sends passes this information to CanvasState.py so that binary files and other variables can be updated.

Use getState() to obtain canvas state.

Use setPixel(...) to set canvas state

Phil

CanvasState.py

Stores canvas state and deals with working with the binary files that store board state and history.

+ Contains constant variable "BoardSize" holding an integer describing the height and width of the canvas in pixels (the canvas is square)

+ Contains variable "CurrentCanvasState[]" that contains pixel data for each pixel in the board. It is a byte array with values ranging from 0 (red) to 11 (red-violet) - see color wheel above - and has the length =

$$BoardSize^2$$

+ Function called "readState()" which reads state from binary file on disk and stores it in CurrentCanvasState[].

+ Function called "getState()" which returns pixel information as byte array.

+ Function called "setPixel(time,pixel, color)" which takes time, pixel index, & integer between 0-11 representing color. Updates CurrentCanvasState[] at specified pixel.

CurrentCanvasState[pixel] = color;

Furthermore, it updates the history binary file writing pixel/color/time.

+ Function called "writeState()" which writes information in CurrentCanvasState to binary file containing current state.

App.js

Display ColorPicker, Canvas, and Chat

+ Has a state called "isLoggedIn" that determines if user is logged in or not.

+ If isLoggedIn is False, do not draw ColorPicker and set Chat and Canvas to disabled (enabled = false)