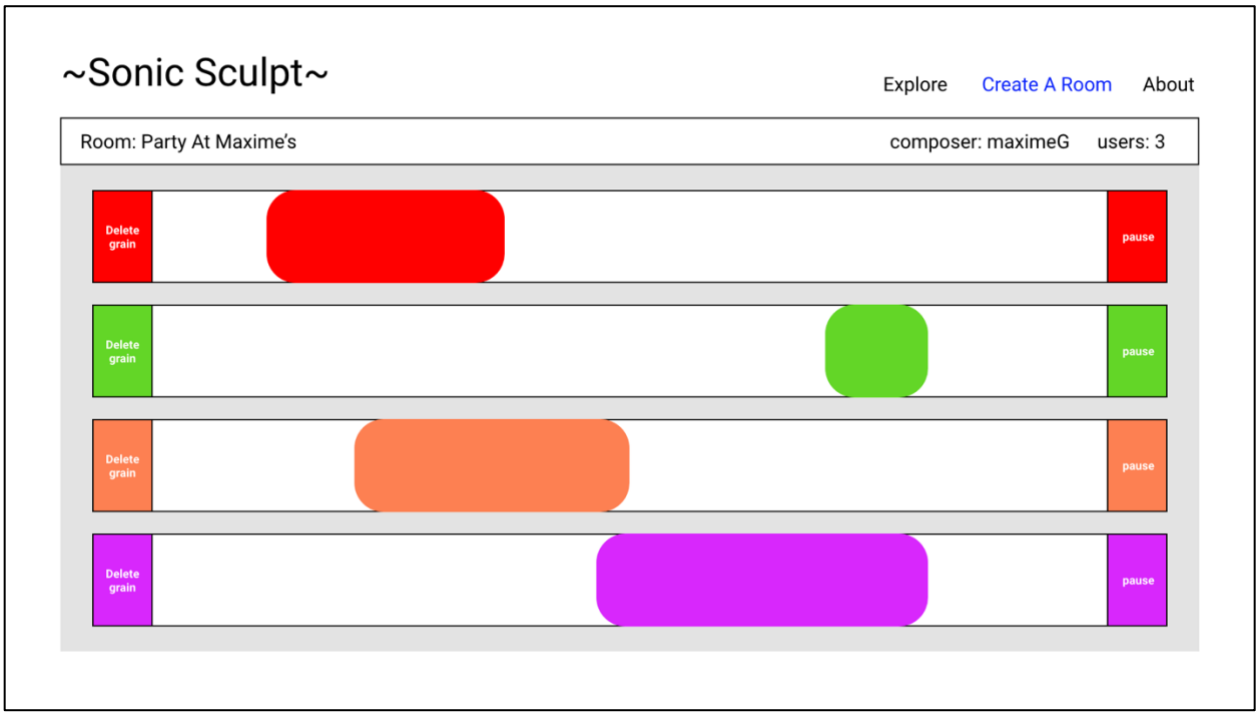
Sonic Sculpt
Prototype Documentation

A web application that let's you share, combine, collaborate on and mix field recordings through granular synthesis



for CART 351 by Maxime Gordon

Original Project Proposal

My original intent for this project was to create a web app for musicians (or those curious about music) that would:

a) encourage collaboration and play between musicians
b) allow recording and playback of found sounds using granular synthesis to encourage discovery and connectedness to a person's surroundings

The main features of the app were based around the concept of 'rooms'. A room would be created by a user and could be joined by other people to collaborate on playing the different channels of granular synths. The features of the app included:

- user login
- ability to save rooms you like
- ability to create rooms
- ability to add many channels of granular synths in a room
- ability to add/delete/play/pause a given channel

I imagined this app to be used on a cellphone for easy portability and to acquire interesting sounds outdoors.

Current Progress

During my prototyping I found that I needed to pare down my original proposal to be feasible for completion by the beginning of December. I have thus made some key design changes which include:

- No more user login
- No more saving rooms by other users
- Limit channels to always be 4 (or 5… not quite decided)
- Focus on making it compatible for a laptop

While I am sad that I have had to take away these features I still believe the intent and integrity of my project is still strong because the core idea of collaborating, playing and discovering with sound is still evident in the UI and UX.

So far I have implemented key sound functionality and have started to work on client, server, and database communication (please see the coded prototype .zip file).

*Note: a large amount of the sound coding is from this site:
https://cm-gitlab.stanford.edu/mherrero/grains4u

Current Implementations

Client side:

User records audio into one of 5 channels. This audio gets saved into a buffer and is played back using granular synthesis. Moving the scroll box over the grain channel changes where in the audio file you are playing. Resizing the scroll box changes how fast the grains play back. You can pause, play, delete and record a new grain for a given grain channel.

APIs:
MediaStream Recording API
https://developer.mozilla.org/en-US/docs/Web/API/MediaStream_Recording_API

I use this API to record audio in from the user's computer microphone. There is a prompt from the browser the first time this happens that the user must accept to be able to record audio.

Server Side:
I've chosen to do a Node.js based server system because my site needs to be able to allow multiple people to interact in real-time within different rooms.

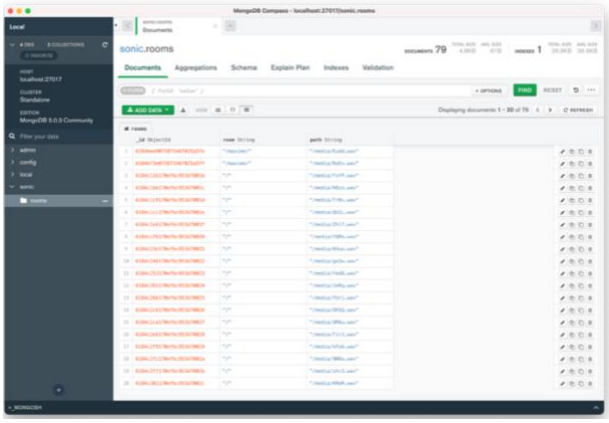
I currently use the Node Modules: Express, Path, Multer, Fs and the database MongoDB and need to still implement socket.io.

Node Modules:
- Express
- Path
- Multer
- Fs
- Socket.io

Database: MongoDB

The part that I have **implemented** on the **server side** in my code is:

Once a user records audio into a grain channel this recorded audio file (not effected by any granular synthesis) is saved into a buffer. Then, JSON data (room_id, and path to this recorded .wav file) is sent to the server and uploaded to MongoDB. Rooms can be changed by typing in the window url. This is basically a functionality test and will be modified to fit the plan that I have created as follows.



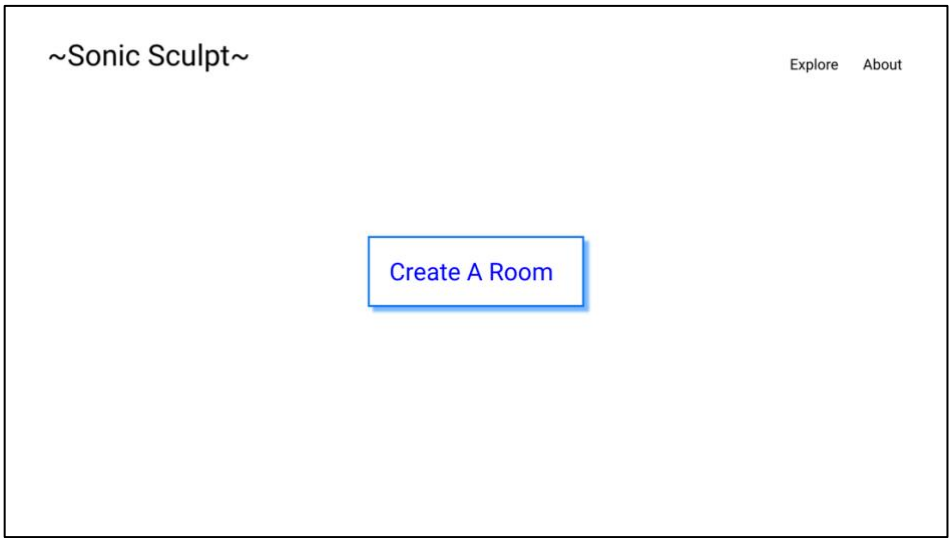
A screenshot of 'Compass' the GUI for using MongoDB. There has been a lot of uploading to the database during my tests…. (79 documents!).

Most of the work that needs to be completed is do with getting and posting data as well as handling socket.io connection:

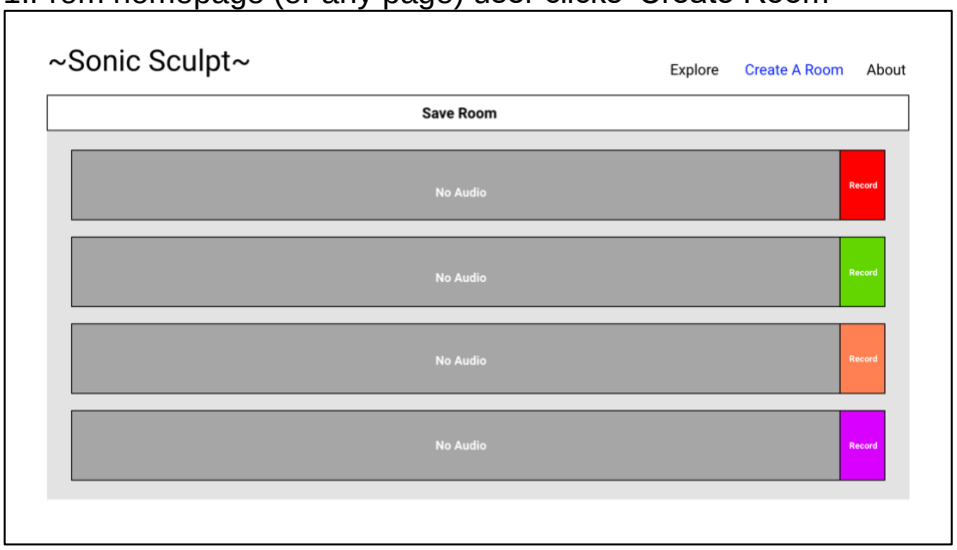
Key Functionality (to be completed):

1. Create a room
2. Explore rooms on the Explore page
3. Join a room (from Explore Page or typing url)
4. Handle people in a room

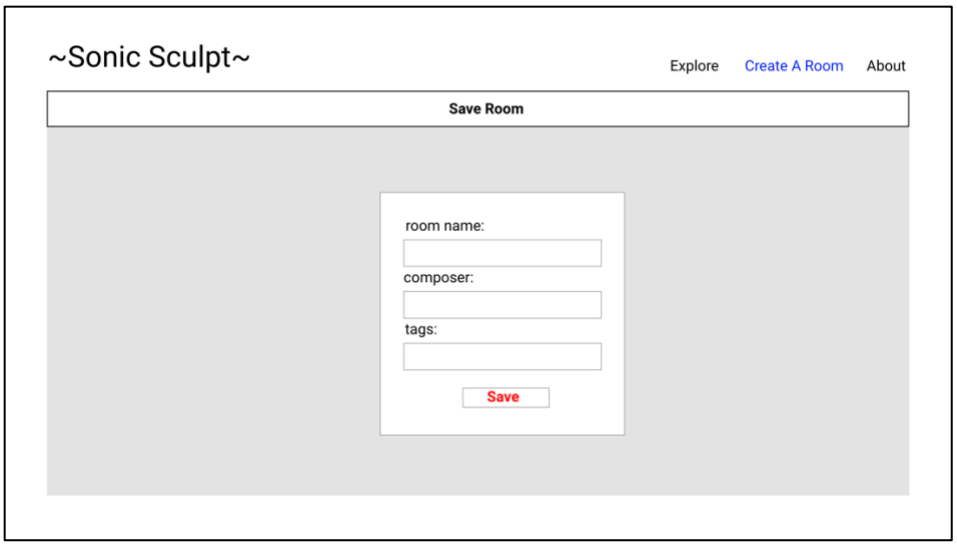
1.Create Room



1.From homepage (or any page) user clicks ‘Create Room’



2. User records various audio samples onto the webpage
3. Once satisfied with sounds the user clicks ‘Save Room’



4. In the Save Room pop-up the user types in:
    1. Room name
    2. composer name
    3. tags

and then clicks ‘Save’

5. Clicking ‘Save’ sends a JSON file to the server that contains:
    -room id
    -username_id
    -tags
    -grain channels(id (1-4), audio file, start of grain from slider, end of grain from slider for saving positions)

This data is submitted to the server and stored on filesystem and mongoDB.

Example of client JSON data to send to server:

```
{
"room": "room_id",
"user": "user_id",
"tags": ["tag1", “tag2”],
"grain_channels": [
{
"id": 1,
"audio": "sample1.wav",
"start": 30,
"end": 50
},
{
"id": 2,
"audio": "sample1.wav",
"start": 30,
"end": 50
},
{
"id": 3,
"audio": "sample1.wav",
"start": 30,
"end": 50
```

}
]
}
Also: whenever you request a room_id (go into a room) the database returns this data to client.

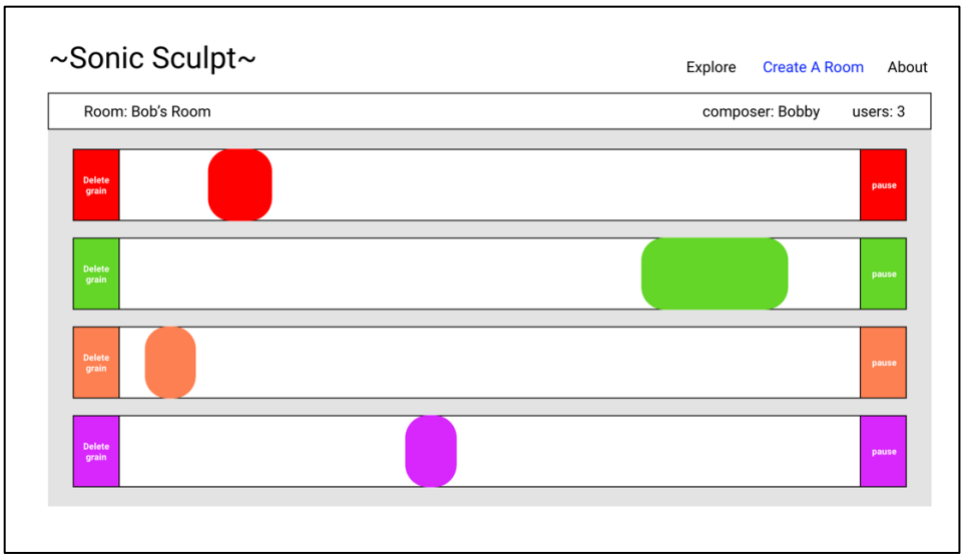
Example of writing to MongoDB:
{
"room": "room_id",
"user": "user_id",
"tags": ["tag1", “tag2”],
"grain_channels": [
{ "audio_url": "/uploads/room_id/sample1.wav", "start": 30,
"end": 50
"id": 1 },
...
]
]

2. Explore Page

1. User clicks ‘Explore’
2. Node.js asks MongoDB for a list of room objects and returns a list of room_id with corresponding url
3. Client renders list of rooms

3. Join A Room

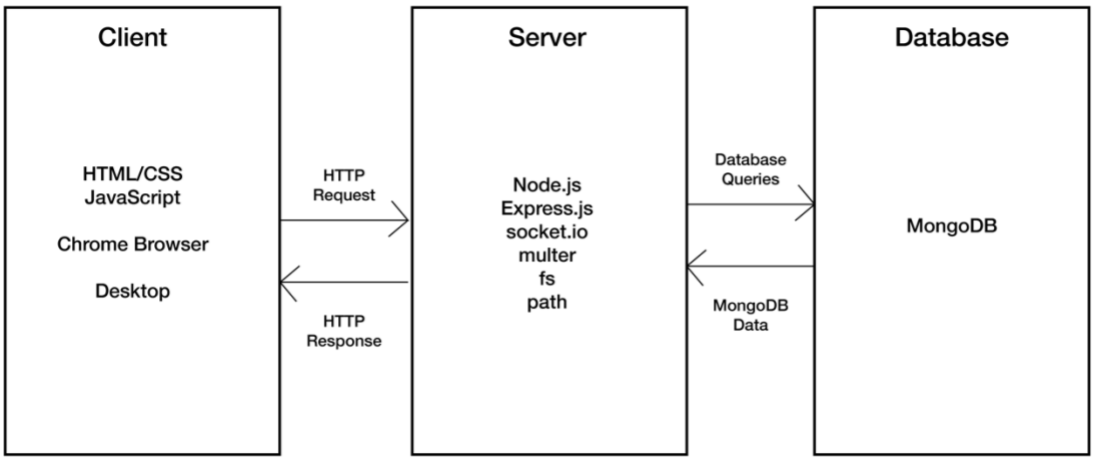
1. Clicks on a room from the Explore Page or enters the url of the room into the browser
2. Node.js asks MongoDB for the room_id
3. Client renders room



4.Handling People in a room (using socket.io)

1. If no socket created, then get the room's state is loaded from MongoDB by Node.js
2. If one person in room a socket is created with a live state by Node.js and socket.io
3. If 2 or more people join the state is loaded from the live state Node.js and socket.io

Simple Data Flow Diagram



Interactive Prototypes

please see code zip file

CLICK HERE FOR WIREFRAME (click anywhere to advance)