FINAL PROJECT REPORT

By: GLYNE Gittens, Kevin Espinola, Kevin yan.

hunter college

CAPSTONE CSCI 49900-01

*Abstract*

For our capstone project[[1]](#footnote-1) we decided to merge the idea of music creating (via step-sequencing sounds on a pad) with streaming. The ideal result would be a website where users could join or create sessions, choose to either create their own song or hang out and wait for other users to finish, and once all the users finish making songs all users within the session are directed to the listening/streaming page. We had extra bells and whistles planned like a chat box and live voting for songs (in the case that someone made a not so great song and users wanted to skip it, they could vote to do so!), but due to the small time frame we fell short on a few of those features. We split up the work of our project into three tasks – Frontend by Kevin Espinola, Backend by Glyne Gittens, Streaming by Kevin Yan. Within this paper you can find individual reports that go more into the details of what each team member accomplished. As a team we are pleased with what we ended up with, despite minor setbacks here and there, the struggles of starting from scratch with just a few crazy ideas and having only ourselves and each other for assistance (and of course our generous professor who always asked if we needed help or had any questions!). This class helped us grow tremendously as programmers, teammates, and people.

**Glyne Gittens - Final Progress Report**

I was responsible for the backend and dev ops side of the project. This consisted of setting up

development and deployment environments for the team, modeling and configuring the

database, and using node in order to perform backend tasks.

In the first two weeks, I worked along with the rest of my team to try to figure out the right tools

for the job. We decided that a Node Js backend would be best for starting a project and

Mongodb as a database would us the flexibility to iterate as we went. Also, Mongodb is heavily

in line with a lot of the functionality of javascript.

During week three, I set up Ubuntu virtual machines on Paperspace for each of the team

members. I also began testing the tools necessary to make the application run by equipping

each of the machines with Node Js, Nginx, pm2, and Mongodb.

Week four is when I started work on the backend in Node Js. I used Express Js as the

underlying web framework. This framework allowed me to do routing and keep the application

persistent in an efficient way. I set up some initial routes, and tested sending music data back

and forth for the music portion of the application, in order to learn how to accept and send music

data over the web.

Week five is when I set up the application to make an initial connection to the database before

starting the app. This ensured that collections were created and the database was running

properly before the app would run. I did this hoping to prevent any sessions where the

application crashed because it tried to read/write to the database but failed.

Week six consisted of more backend work, mostly working on the routing logic and updating

pages that Kevin E needed me to serve to him. I also upgraded everyone from Mongodb 3.4 to

4.0.1. There were several useful methods that I didn't have, as well as reliable ways of schema

validation offered in 4.0.1.

During week 7, I was adding more methods to read/write data to the database. I ran into a lot of

issues at this point because I wasn't familiar with a lot of Mongodb's methods and what they

returned. I found out that a lot of the data types that needed to be saved to mongo were specific

mongo data types and not general javascript types like a number or hex string.

During week 8, I mostly worked on the logic for creating directories that corresponded to the

different music creating sessions. This consisted of listening on the new session endpoint,

generating an entry in the database, using the id to make and name a new directory, then

responding with the path of the session. At this stage I also added MulterJs which accepted and

saved a music file which was uploaded to the server and saved it as a part of its corresponding

session.

Weeks nine and ten consisted of making corrections to a lot of the routes that I made in order to

make sure that the front end had the supporting data that it needed. This included sending

objects as responses to POST requests or logged in user data when the front end request a

route using the GET method.

Weeks eleven and twelve consisted of mostly bug fixes and adding routes to support the both

the front end and streaming portion of the project. This was a point where I had a lot of struggles

with the database because of certain Mongodb functions. For example functions like $push had

documentation that wasn't quite in line with what we were doing.

**Glyne Gittens – Conclusion**

In the end, I feel like I did well in my initial setup and execution of our project but I think that over

time I wasn't doing enough rigorous testing of my code. There were a lot of bugs that we came

to as time went on that were simple oversights. I learned a lot about using NoSQL databases

and some of the problems one may run into. I learned a lot more about javascript as a language

and using modular structures in order to pass data through the app. I was super excited to learn

more about programming music services and I feel like I got a lot of experience doing that in this

project. I'm happy that we only focused on the core aspects of the project like the creation

portions and how we would actually save music server side. That helped us to clear up a lot of

questions and assumptions early on. If I had more time, I would make the user models better so

that users could have actual profiles where people could run through their catalogue of songs created and where they ranked.

Kevin Espinola

*Introduction & Team Organization*

For this project my task was to handle the Frontend side of things. I was responsible for the HTML, CSS, and JavaScript that users of our website would see and interact with. What that entails are the website design, website functionality, [and what 80% of my work was spent on](https://codepen.io/kre64/pen/ZjyOEY)[[2]](#footnote-2) – the actual pad that users use to make and record music. I have a small background in music and I felt very passionate about doing something for capstone that was music related so I was very eager to do this part of the project. It was a lot of fun to learn about programming sounds and then combining that newfound knowledge with markup languages like HTML and CSS to slowly figure out how to make my idea come to life.

For the **first week and a half** I spent a majority of my time learning HTML, CSS. It was a little daunting at first, but I relied on Glyne’s experience and he pointed me towards [Bootstrap](https://getbootstrap.com/)[[3]](#footnote-3), a toolkit for developing with HTML, CSS, and JavaScript for creating responsive grids and layouts for our website. During these early periods we established methods of organization and communication via GitHub, Slack, and Google Docs. I also was working on the slides and internal mockup art for the website and what the pad would look like, so my other group members could have a better idea of what we need to accomplish.

**Weeks 2-3** I realized I needed to create a grid using HTML and CSS that would later correspond to a matrix in JavaScript that would be the actual pad.

A close up of a black keyboard

Description generated with very high confidenceA screen shot of a building

Description generated with high confidence  
The pad on the left was a mockup made by me in MS Paint, the pad on the right was one of the early pads I made using HTML and CSS, at this point I had not implemented proper sound because I was still learning how to do that, and it was important that I got sound working properly so I could scale it up to the entire grid. By week three I was learning all about the [WebAudio API](https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API)[[4]](#footnote-4) and started experimenting with sounds in JavaScript (here’s a tip, if you’re going to work with sound make sure your volume isn’t too high, so you don’t end up accidentally deafening yourself!).

**Weeks 3-5** was iteration after iteration of the pad design, sound was working for each button, but I had issues getting the sounds to play properly. Sometimes a sound played, and it muted the entire thing, other times the whole thing would just crash. In the middle of these three weeks I resolved my issues by separating the method through which sound played and added three buttons – PLAY, STOP, RECORD. Only PLAY and STOP worked at the moment and it simply looped through the matrix of buttons and played each column to the tune of a constant internal tempo (which was tied to a JavaScript setInterval() function outside of a big loop). This screenshot is from a version after week 4, the white audio bar at the top was not implemented yet, everything under “more text here” was implemented though.

A screenshot of a cell phone

Description generated with high confidence

**Weeks 6-7**, getting record to work. This was tough, I spent a lot of time getting familiar with the [MediaStream Recording API](https://developer.mozilla.org/en-US/docs/Web/API/MediaStream_Recording_API)[[5]](#footnote-5). I wasn’t sure what kind of magic was going on with the actual data itself (the sound data), I attached a MediaStream to a WebAudio destination node where all the sounds from the pad were being routed to and fired off from. The MediaStream would begin recording when a user hits record and stop recording once the user hits that same button again. After that, the audio recorded was sent to the top audio player (seen in the previous screenshot) where the user could hit the triangle play button and hear their music played back. However, that audio was bugged and for some reason would play back too fast. Eventually I fixed this bug by changing the method through which audio was generated, before it would start at maximum volume of 1.0 and logarithmically decrease to 0.001 volume, after the fix it was a linear gain from 1.0 to 0, for some reason MediaStream Recording wouldn’t record the gaps between sounds when the volume decreased logarithmically, but when it was linear the recording suddenly worked as intended. Perhaps because the linear method continuously plays volume at 0 whilst the logarithmic method would cut off the volume after decreasing to 0.001 – this isn’t documented by either API I used (WebAudio or MediaStream) so it’s just an educated guess on my part.

**Weeks 7-9**, when things had to change. I really wanted to work with samples (audio clips less than 1-2 seconds long) from the backend instead of the WebAudio oscillators I currently had that were generating the sounds. Turns out it just wasn’t feasible and took up way too much of my time. It was time for me to move on and work on other things. Since the backend routing wasn’t finished yet I decided to polish up the frontend design, I added a fancy navbar for when the routing would be implemented and touched up the HTML and CSS on the pad itself. I added a clear button to clear the grid, a Tempo form that would take numbers as input and set the rate at which user music would play, a white glow around the currently

*A picture containing screenshot

Description generated with high confidenceA close up of a computer keyboard

Description generated with high confidence*

Playing column, and a pop animation for when buttons are played, clicked, or hovered over (seen in both screenshots above).

**Weeks 9-11** (and further into the future!). At this point I was working harder than normal, fixing bugs on my end and merging with my other group members (mostly with Glyne at this point, frontend+backend needed to be solid). Most of the big bugs (which I will list known bugs before concluding my section of the report) were squashed in these weeks, record used to overwrite previous recordings which was not ideal, meaning resume and pause were not working as intended. I resolved this by using more methods within MediaStream Recorder API and by only allowing the user to record one grid’s worth of music at a time. Weeks 10,11 in this stretch were very stressful, connecting the frontend with the backend was indeed a challenge. During the week of the final presentation where we had to demo the final “working” product I was making sure we could show off all parts of the project working as one, it didn’t happen, so we demo’d what the functioning website would be like. Glyne and myself were frequently collaborating at this point to fix issues on the front/backend, after the final presentation we fixed the website so most of the site has a working flow

(homepage **->** login/signup **->** list sessions **->** make new session/join a current session

**->** create music/listen to music **->** listen page where music will stream to users)

Currently, at the current time I am writing this (Aug – 12 @3:41PM) the general website flow is still not very logical. Not everything works as it should and I wouldn’t say our project as a collective group effort is finished by my own metrics.

**Known Bugs:**

* Hitting Play several times makes everything stop working sometimes.
* Activating all the buttons within the pad may or may not freeze everything
* Stop needs to be hit several times to stop playback
* Tempos higher than 800 may sound cool for a while before it mutes everything

Frontend/Backend Bugs

* Users login status is not persistent (seeing the list and going back in browser=not good)
* User signups and logins with non-numeric, non-alphabetic characters break things
* Songs are not being saved properly ->
* Songs are not being loaded properly
* Streaming doesn’t work?

*Conclusion*

I had very little experience developing for the frontend prior to this project, so using JavaScript in conjunction with HTML and CSS was a learning curve but once I got over that it was relatively smooth sailing. I also learned how to generate sounds for web applications using the WebAudio API and how to record audio on websites using the MediaStream Recording API. Working with teammates, building a project from scratch, and how to compartmentalize different parts of a collaborative project were also all very vital skills I cultivated during the project. Lots of things I could have done better, I’ve always had this mindset where I look back at what I’ve done and think about how much better it could have been. Specifically, when it comes to the markup design I did for the website – that could have been done better, I also believe I could have been a better team leader if I had spent a little more time at the beginning of the project outlining every single detail of the website, instead of focusing mostly on my own part. There’s so many things I want to add to the project, better designs, a custom audio player, the ability to record user data and use that for audio instead of the default oscillators currently producing sound, preset audio from the backend that users could load instead of the default sound. I’m very pleased with the work I contributed to the project and hope to keep working on it after the conclusion of this course.

Kevin Yan

Final Project Report

1 Abstract

Our project is basically an application where multiple users start a session where one

would be able to record music that one has made. Each user composes a piece and after

recording the musical piece they are to submit it. After all users are done with uploading their

music they are then to choose the best one out of all the submissions in that session. This

project requires the front end where they are able to make the music and the backend where we

decide the session and where the music is streamed to.

2 Introduction

In this project my job was to work on the streaming aspect of the code. I have

successfully completed that task by sending the stream to the tcp server that i created . Now i

need to combine my code with the other group members. That might be a problem as some of

our code might conflict. Sill working to being able to query where the locations of the composed

music through the database.

3 Team Organization

I was responsible for some of the back-end coding. Specifically the streaming

aspect of this project.

A screenshot of a cell phone screen with text

Description generated with very high confidence

Currently trying to transverse the database and locate the music files and sending them

to the right session

4 Conclusion

On this project i learned that there are many aspects in which i need to know about. I

have never worked on back-end programming. I learned javascript and how convenient node js is and learned a bunch of new API.

*Various Resources & References*

*Used throughout our project*

[Drum Machine Program](https://github.com/maryrosecook/drum-machine)

<https://www.ffmpeg.org/documentation.html>

[SoundJs](https://github.com/CreateJS/SoundJS)

[RecorderJs](https://github.com/mattdiamond/Recorderjs)

[MediaStream HTML5](https://developer.mozilla.org/en-US/docs/Web/API/MediaStream)

[Using ffmpeg for streaming](https://trac.ffmpeg.org/wiki/StreamingGuide)

[Basic Concepts Behind Mozilla Web Audio API](https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Basic_concepts_behind_Web_Audio_API)

[Mozilla Web Audio API](https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API)

[WebAudio Intro](https://www.html5rocks.com/en/tutorials/webaudio/intro/)

[Node ffmpeg library](https://www.npmjs.com/package/ffmpeg)

[Node ffmpeg library](https://github.com/fluent-ffmpeg/node-fluent-ffmpeg)

[Mozilla WebSockets](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API)

[React Bootstrap](https://github.com/react-bootstrap/react-bootstrap)

[React Js](https://reactjs.org/)

[Twitter Bootstrap](https://getbootstrap.com/)

[MongoDB C++ Driver](http://mongodb.github.io/mongo-cxx-driver/)

[PUGjs](https://pugjs.org/api/getting-started.html)

[Receiving Audio In Node Backend](https://stackoverflow.com/questions/31756393/accepting-a-wav-audio-file-over-http-post-in-an-express-node-server)

[Handling Blobs](https://stackoverflow.com/questions/35418921/how-to-handle-xhr-blob-post-in-nodejs)

Links to our hosted content & source code online!

74.82.53.200 - Glyne Gitten's paperspace server

74.82.53.201 - Kevin Yan's paperspace server

74.82.53.202 - Kevin Espinola's paperspace server

All work done via git logged onto our branches

from master-> @ https://github.com/ArtBears/livepad

/branchke = Kevin Espinola's branch

/branchky = Kevin Yan's branch

/webserver = Glyne Gitten's branch

1. Please check the README.txt file inside the zip for instructions on how to see our project [↑](#footnote-ref-1)
2. <https://codepen.io/kre64/pen/ZjyOEY> [↑](#footnote-ref-2)
3. <https://getbootstrap.com/> [↑](#footnote-ref-3)
4. <https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API> [↑](#footnote-ref-4)
5. <https://developer.mozilla.org/en-US/docs/Web/API/MediaStream_Recording_API> [↑](#footnote-ref-5)