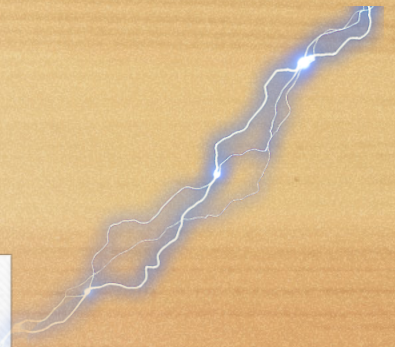


The ARM: Tracking Our Digital Child

A Scrapbook by Saketh Sitaram and Evan Keeton

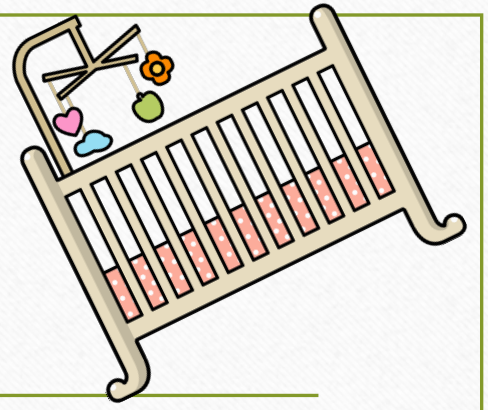
The Proud Parents

Mission Statement

- To address the difficulty musicians face when performing without an assistant or deskmate, we will create an Assistant Robotic Musician (ARM) which can fulfill the same role more efficiently and less intrusively. The ARM will be a free, open-source, cross-platform application into which one can load sheets of music or e-booklets of sheet music, with embedded software to process live audio of the performance and automatically determine when to flip the page. Additionally, it will automatically process scanned sheet music, as well as allow for editing, storing and organizing these processed files in a vast library. Furthermore, the eponymous, friendly AI, "Arm," will manage both the library and the performance software. We hope to make this a collaborative effort, inviting musicians to write sheet music compatible with the ARM. Together, we hope to ease the process of performing with sheet music by removing distractions and difficulties in handling the music.

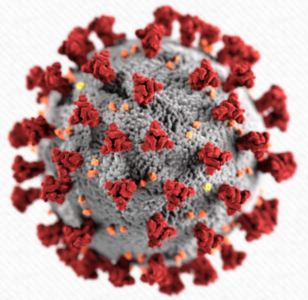
Contents

Summer 2020



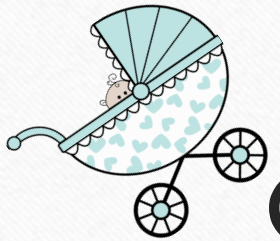
- During the summer, Saketh boosted his Python and HTML skills so that he could give our baby all the nutrition it deserved.
- Evan organized himself and took care of many items, including brushing up on his programming skills, so that he could make time for the baby as it passed through the developmental stages of life.

September 2020

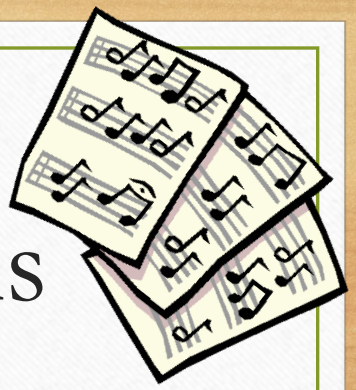


- Little bit of a setback. Due to the relentlessness of businesses in failing to get us our quotes, along with the concerns of fostering our child during a pandemic, we made the decision to have it be purely digital, without including a software component.
- We have started talking to our mentors, who will be guiding us throughout these testy times. Let's hope that the baby remains healthy, without any bugs or viruses! Fingers crossed!

Similac



One Year Down The Line: Predictions



- We expect our child to be well-behaved, who can track when someone plays along with its sheet music live, in real time, and turn the page through intuition.
- We expect it to include a top-notch sheet music editor, all with real-time processing. We are also planning to raise it with the ideals of storage, enabling it with the capability to store sheet music in a vast library.
- We expect it to be cross-platform, working with Windows, Mac OS, and Linux with similar degrees of functionality.

Name!

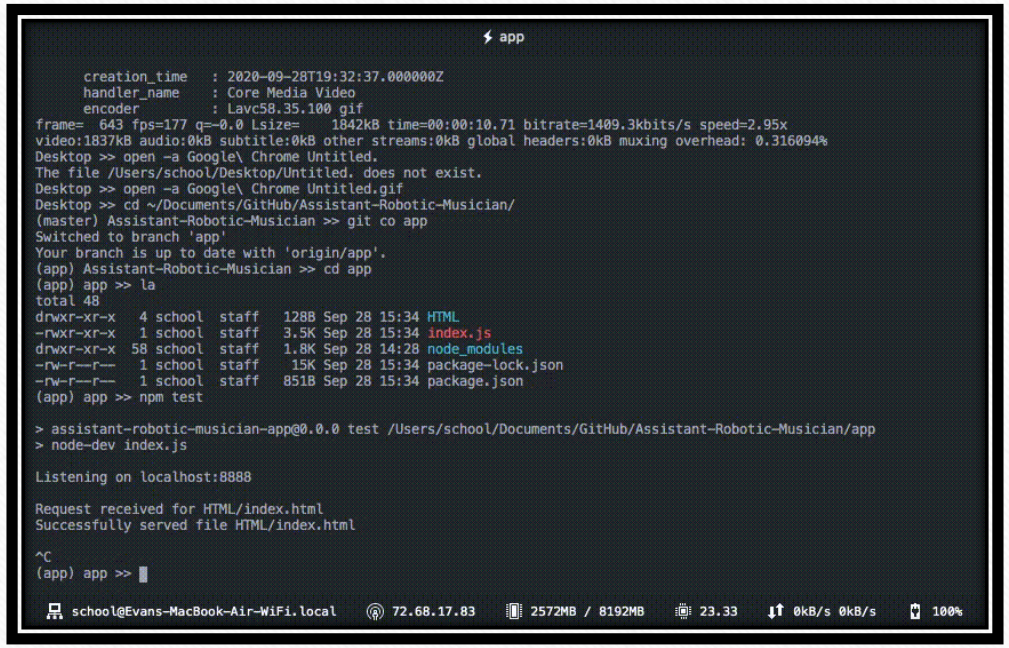
- After months of serious thought, we have finally chosen a name for our child! **Drumroll please**
- In 9 months, we will hopefully welcome to the family ARM Keeton-Sitaram!
- This name will be most directly correlated with the friendly AI “Arm” integrated into the project, who will both guide the user and manage the library.

In-School Mentor

- Our in-school mentor is Mr. Jaworski, the guitar and music technology teacher at Edison High School. He will be guiding us along the way, ensuring that our baby emerges powerful and useful, with an effective user-interface.

Ultrasound!

- We got our ultrasound! Our first look at our sweet little ARM. Isn't it so cute?



October/November 2020

- We have created a formal separation of powers for our project
  - Evan will be handling the back-end work – audio processing, PDF processing (flesh and bones)
    - Main language - Python
  - Saketh will be handling the front-end work – framework and layout (skin and hair)
    - Main languages – HTML, CSS, JS

Out-of-School Advisor

- Our out-of-school advisor is Mr. Vladimir Valjarevic, an esteemed pianist who teaches at the Mason Gross School of the Arts at Rutgers University. His experience with professional classical music makes him an invaluable resource to finalize the various nuances of our software. Also, he would make a perfect test subject!

Advisor Meeting

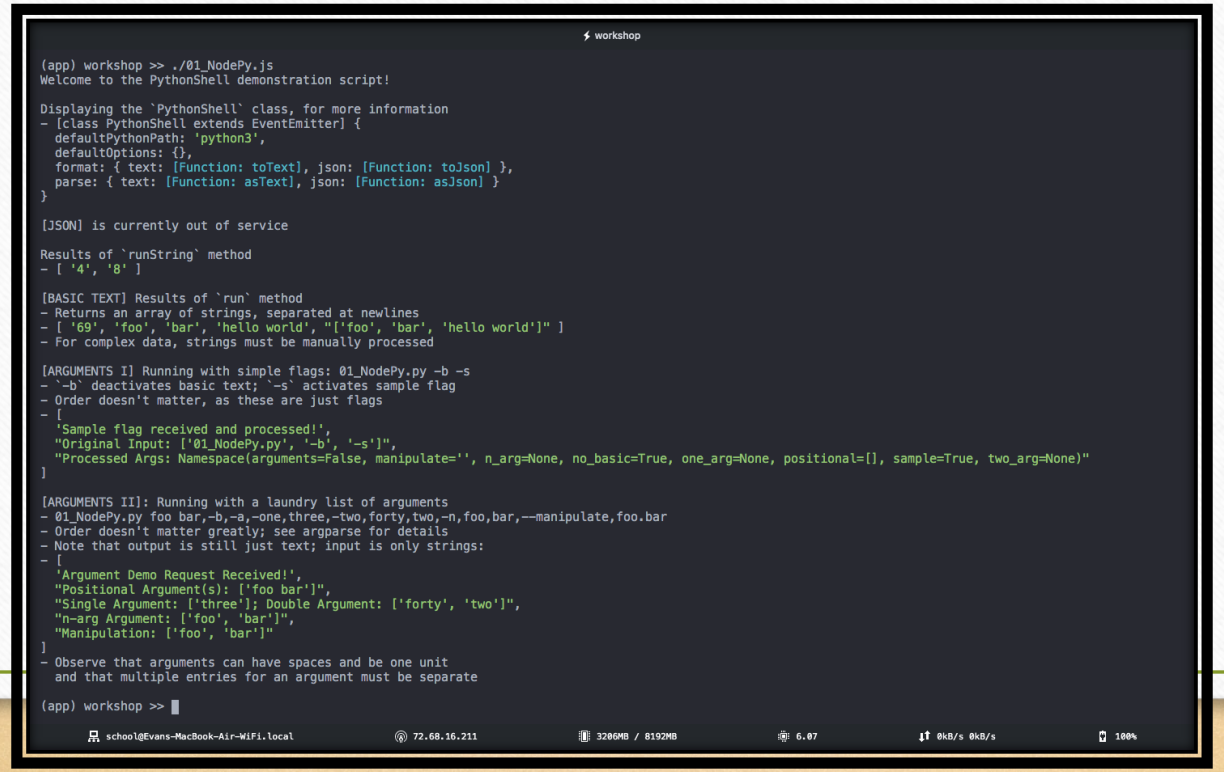
- We held a meeting with our advisor, Mr. Valjarevic, in which we exchanged formalities and gave him a thorough rundown of how the project will work.
- In addition, we discussed our timeline and how we are going to distribute the work across the coming months.

Saketh's Progress

- Over the course of October and November, Saketh cultivated his HTML/CSS/JS skills by building many dummy websites. In addition, Saketh got to work on the layout, how the software will look.

Evan's Progress

- Evan started work with NodePy, to embed Python into a NodeJS app.



```
workshop

(app) workshop >> ./01_NodePy.js
Welcome to the PythonShell demonstration script!

Displaying the `PythonShell` class, for more information
- [class PythonShell extends EventEmitter] {
  defaultPythonPath: 'python3',
  defaultOptions: {},
  format: { text: [Function: toText], json: [Function: toJson] },
  parse: { text: [Function: asText], json: [Function: asJson] }
}

[JSON] is currently out of service

Results of `runString` method
- [ '4', '8' ]

[BASIC TEXT] Results of `run` method
- Returns an array of strings, separated at newlines
- [ '69', 'foo', 'bar', 'hello world', "['foo', 'bar', 'hello world']" ]
- For complex data, strings must be manually processed

[ARGUMENTS I] Running with simple flags: 01_NodePy.py -b -s
- `-b` deactivates basic text; `-s` activates sample flag
- Order doesn't matter, as these are just flags
- [
  'Sample flag received and processed!',
  "Original Input: ['01_NodePy.py', '-b', '-s']",
  "Processed Args: Namespace(arguments=False, manipulate='', n_arg=None, no_basic=True, one_arg=None, positional=[], sample=True, two_arg=None)"
]

[ARGUMENTS II]: Running with a laundry list of arguments
- 01_NodePy.py foo bar,-b,-a,-one,three,-two,forty,two,-n,foo,bar,--manipulate,foo.bar
- Order doesn't matter greatly; see argparse for details
- Note that output is still just text; input is only strings:
- [
  'Argument Demo Request Received!',
  "Positional Argument(s): ['foo bar']",
  "Single Argument: ['three']; Double Argument: ['forty', 'two']",
  "n-arg Argument: ['foo', 'bar']",
  "Manipulation: ['foo', 'bar']"
]
- Observe that arguments can have spaces and be one unit
  and that multiple entries for an argument must be separate

(app) workshop >>

school@Evans-MacBook-Air-WiFi.local  72.68.16.211  3206MB / 8192MB  6.07  0kB/s 0kB/s  100%
```

December 2020

- Lots of lab time: extensive tangible progress
- Saketh
- Evan: Developing functions for processing audio data
  - Manipulating audio data files (using `python.wave`)
  - Reading audio data as "chunks" at 30 fps
  - Research on FFT (frequency detection algorithm)

Advisor Meetings

- Met with both our in-school and out-of-school mentors to plan the next phases of the project
- In-depth SMML brainstorming with Mr. Jaworski
  - SMML: Sheet Music Markup Language – Technical language for communicating between music & machine
  - Received sample audio
- In-depth GUI brainstorming with Mr. Valjarevic

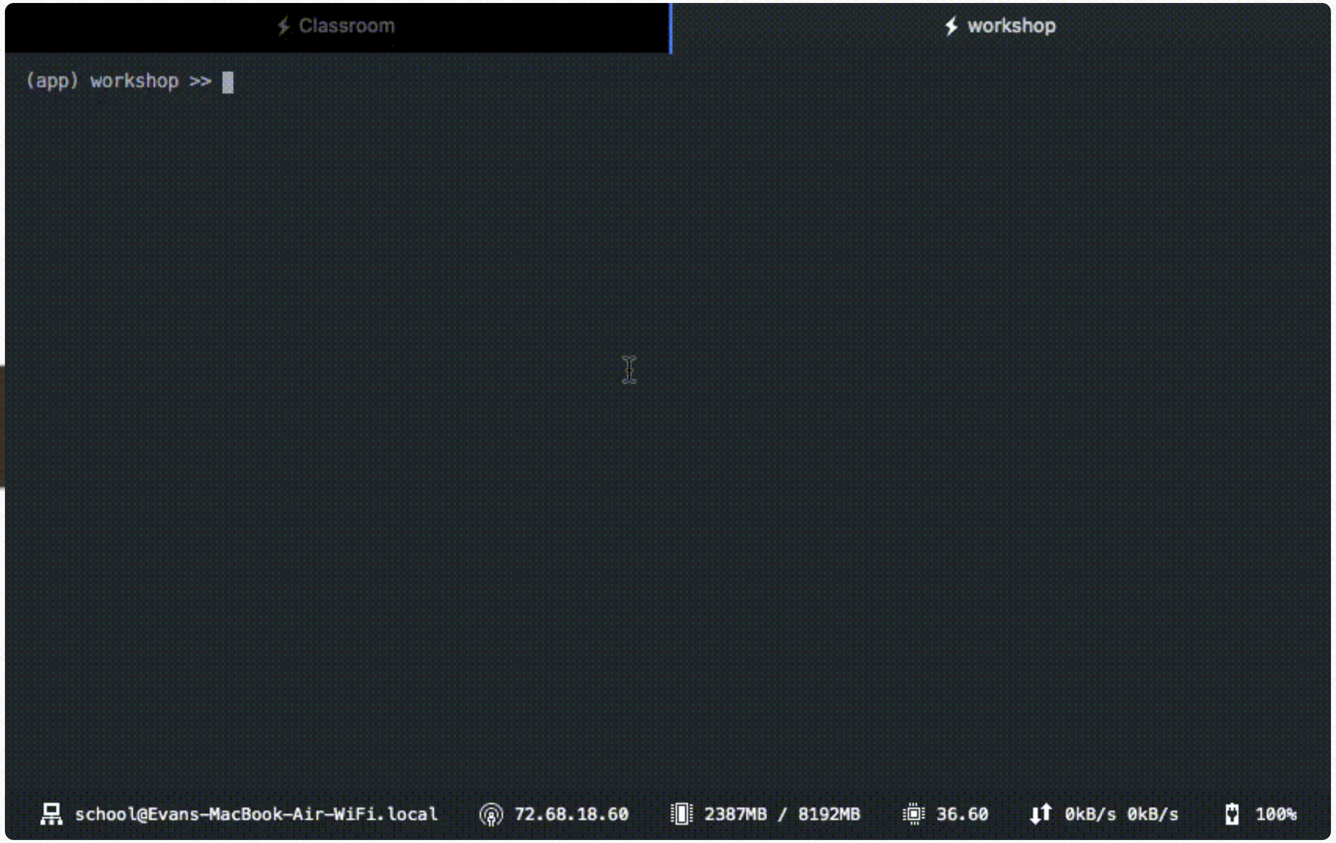
Saketh's Progress

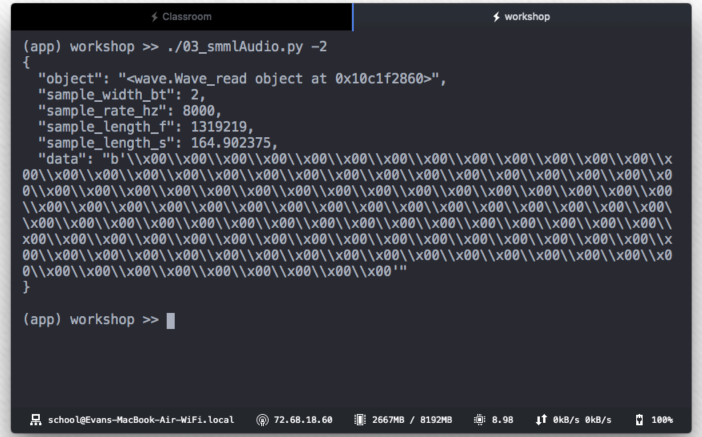
- Over the course of October and November, Saketh cultivated his HTML/CSS/JS skills by building many dummy websites. In addition, Saketh got to work on the layout, how the software will look.

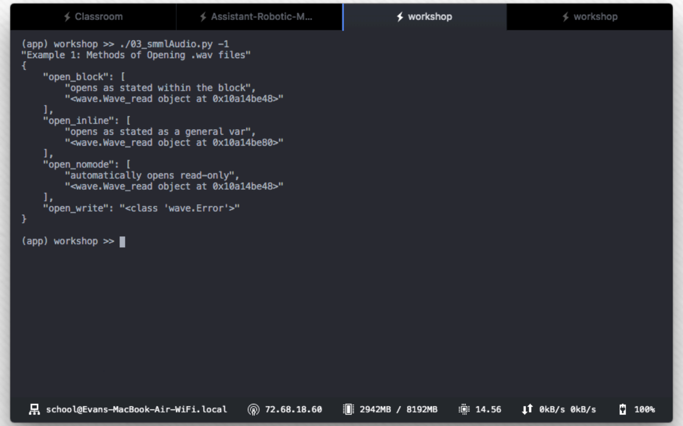
```
workshop

(app) workshop >> ./01_NodePy.js
Welcome to the PythonShell demonstration script!

Displaying the `PythonShell` class, for more information
- [class PythonShell extends EventEmitter] {
  defaultPythonPath: 'python3',
  defaultOptions: {},
  format: { text: [Function: toText], json: [Function: toJson] },
  parse: { text: [Function: asText], json: [Function: asJson] }
}

[JSON] is currently out of service

Results of `runString` method
- [ '4', '8' ]

[BASIC TEXT] Results of `run` method
- Returns an array of strings, separated at newlines
- [ '69', 'foo', 'bar', 'hello world', "['foo', 'bar', 'hello world']" ]
- For complex data, strings must be manually processed

[ARGUMENTS I] Running with simple flags: 01_NodePy.py -b -s
- `-b` deactivates basic text; `-s` activates sample flag
- Order doesn't matter, as these are just flags
- [
  'Sample flag received and processed!',
  "Original Input: ['01_NodePy.py', '-b', '-s']",
  "Processed Args: Namespace(arguments=False, manipulate='', n_arg=None, no_basic=True, one_arg=None, positional=[], sample=True, two_arg=None)"
]

[ARGUMENTS II]: Running with a laundry list of arguments
- 01_NodePy.py foo bar,-b,-a,-one,three,-two,forty,two,-n,foo,bar,--manipulate,foo.bar
- Order doesn't matter greatly; see argparse for details
- Note that output is still just text; input is only strings:
- [
  'Argument Demo Request Received!',
  "Positional Argument(s): ['foo bar']",
  "Single Argument: ['three']; Double Argument: ['forty', 'two']",
  "n-arg Argument: ['foo', 'bar']",
  "Manipulation: ['foo', 'bar']"
]
- Observe that arguments can have spaces and be one unit
  and that multiple entries for an argument must be separate

(app) workshop >>

school@Evans-MacBook-Air-WiFi.local  72.68.16.211  3206MB / 8192MB  6.07  0kB/s 0kB/s  100%
```

Evan's Progress

- Successful Python-NodeJS communication via JSON
- Tested various methods of opening and reading .wav files
- Explored the binary data structure
- Tested ways to create "audio fragment" objects
  - These are the data structures that the algorithm uses
- Next Steps: Applying the FFT Algorithm to Live Audio


Classroom
workshop
(app) workshop >>
school@Evans-MacBook-Air-WiFi.local
72.68.18.60
2387MB / 8192MB
36.60
0kB/s 0kB/s
100%


Classroom
workshop
(app) workshop >> ./03_smmlAudio.py -2
{
"object": "<wave.Wave_read object at 0x10c1f2860>",
"sample_width_bt": 2,
"sample_rate_hz": 8000,
"sample_length_f": 1319219,
"sample_length_s": 164.902375,
"data": "b'\\x00'"
}
(app) workshop >>
school@Evans-MacBook-Air-WiFi.local
72.68.18.60
2667MB / 8192MB
8.98
0kB/s 0kB/s
100%


Classroom
Assistant-Robotic-M...
workshop
workshop
(app) workshop >> ./03_smmlAudio.py -1
"Example 1: Methods of Opening .wav files"
{
"open_block": [
"opens as stated within the block",
"<wave.Wave_read object at 0x10a14be48>"
],
"open_inline": [
"opens as stated as a general var",
"<wave.Wave_read object at 0x10a14be80>"
],
"open_nomode": [
"automatically opens read-only",
"<wave.Wave_read object at 0x10a14be48>"
],
"open_write": "<class 'wave.Error'>"
}
(app) workshop >>
school@Evans-MacBook-Air-WiFi.local
72.68.18.60
2942MB / 8192MB
14.56
0kB/s 0kB/s
100%

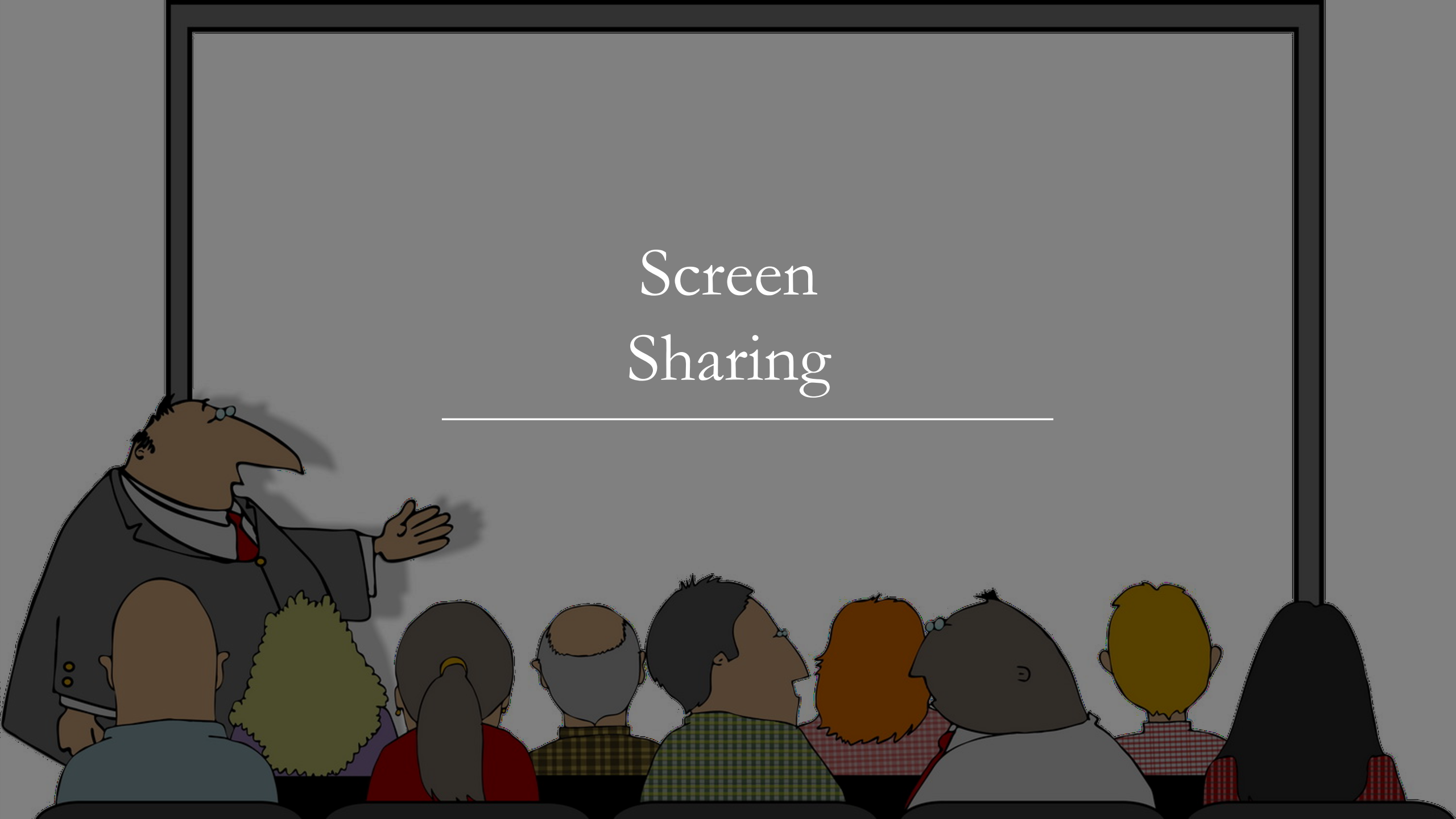
January 2021

- Lots of lab time: extensive tangible progress — tangible products
- Saketh: Main GUI development (with HTML)
  - Translated outline of project into a web display
  - Skeleton, or "husk," of the app itself
  - Developed JavaScript skills for the next part of the project
- Evan: Developing functions for processing audio data
  - Reading live audio at a given frame rate
  - Processing binary files to manipulable arrays
  - Frequency extraction with FFT
  - Volume estimation function

Advisor Meetings

- Met twice with both our in-school and out-of-school mentors to plan the next phases of the project: December and January
- December: Reviewed December progress and discussed GUI (brainstorming before development)
- January: Reviewed January progress and discussed both UI and Backend—now that there is an actual product, something actually visible—and planned the process of integrating the two components

Screen Sharing



Next Steps

- Combine GUI and Backend into a seamless app
- Backend: Interpret Processed Data as Music
  - Develop a script to detect note changes (accounting for error)
  - Develop a script to associate notes with SMML
  - Combine to track the progress of a piece
- Frontend: Display SMML as HTML
  - Display SMML as HTML -> Create sheet music interface with buttons having well-defined and executable functions
  - Create ElectronJS Shell for HTML code -> Turn web display into an app
  - Develop a script that responds to a “page turn” signal

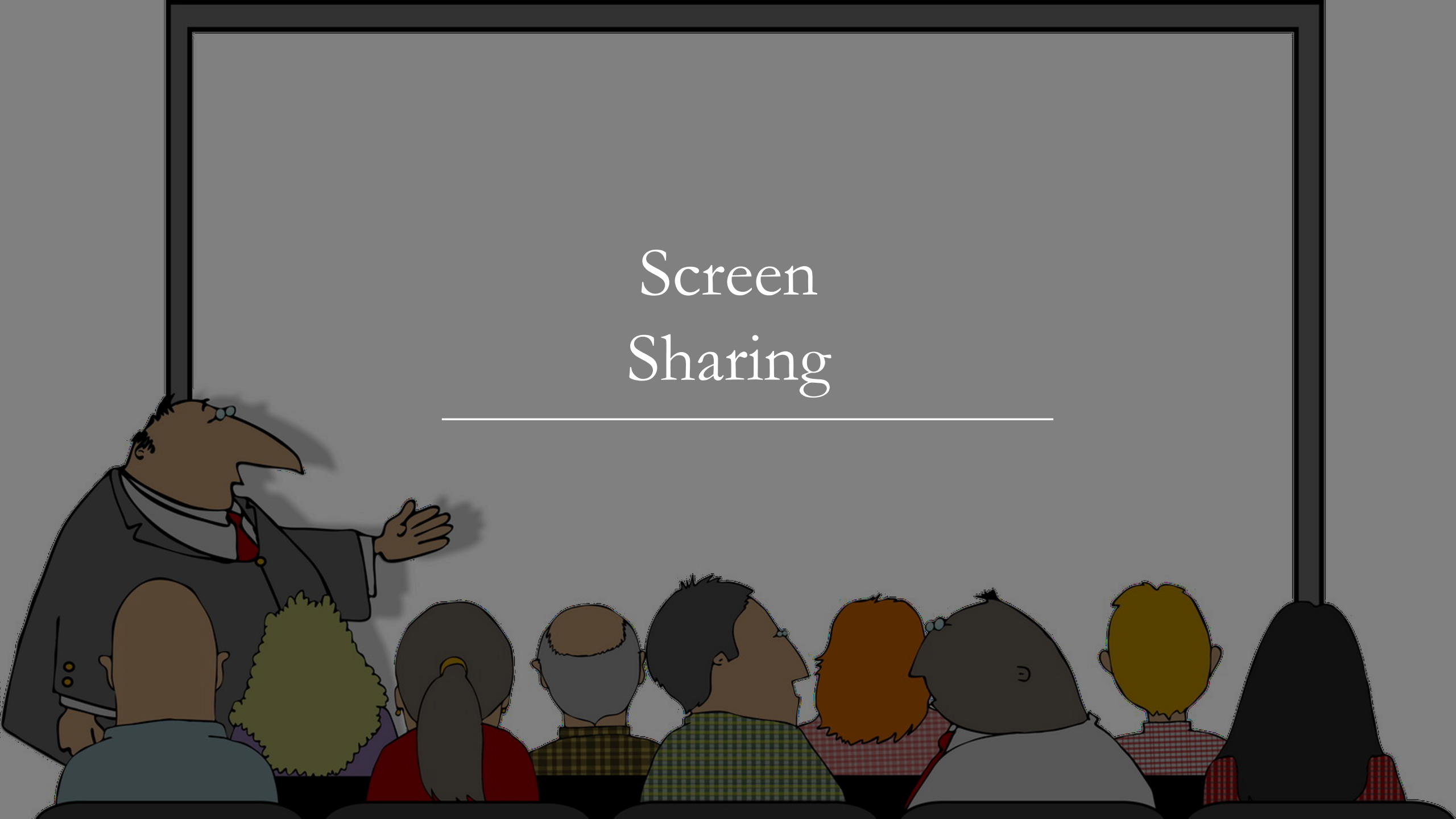
February 2021

- Saketh – continuing GUI – working with the JavaScript aspect of HTML/CSS/JS – gradually filling in previously-made "husk"
  - Created display for the sheet music in the Edit page
  - Created layout for displaying measures through HTML
  - Implemented "Add One Measure" function for one line of measures – however, button is *inside* the sheet music display rather than on the Edit page itself
  - Furthered skills in Jquery and AJAX to work on Play page
- Evan – continuing Python backend
  - Process detected frequencies into pitches
  - Convert SMML into a list of "nodes" ("identifiers" with certain elements) and use the detected pitches to follow this architecture

Advisor Meetings

- With both our in-school and out-of-school mentors in February to plan the next phases of our project: integration + testing
- Reviewed February progress and discussed both UI and Backend separately – now that there is some functionality to components in UI, the integration of frontend and backend seems tangible and doable

Screen Sharing



Next Steps

- Combine GUI and Backend into a seamless app – Goal for March
- Backend: Streamlining and debugging
  - Lots of debugging and testing: Best sample rates and frame rates, best audio format, etc.
  - Streamline the sprawling functionality into a single well-documented package (for open-source purposes)
- Frontend: Display SMML as HTML – Mainly in Play page
  - Create display with clearly defined measures and notes – translate from SMML to sheet music display
  - Improve ElectronJS Shell – should open as its own app and not through a parent terminal
  - Develop a script that responds to a "page turn" signal
- Combined: Streamline the functions into a self-contained, distributable and testable app
  - Goal: Something to distribute to our mentors

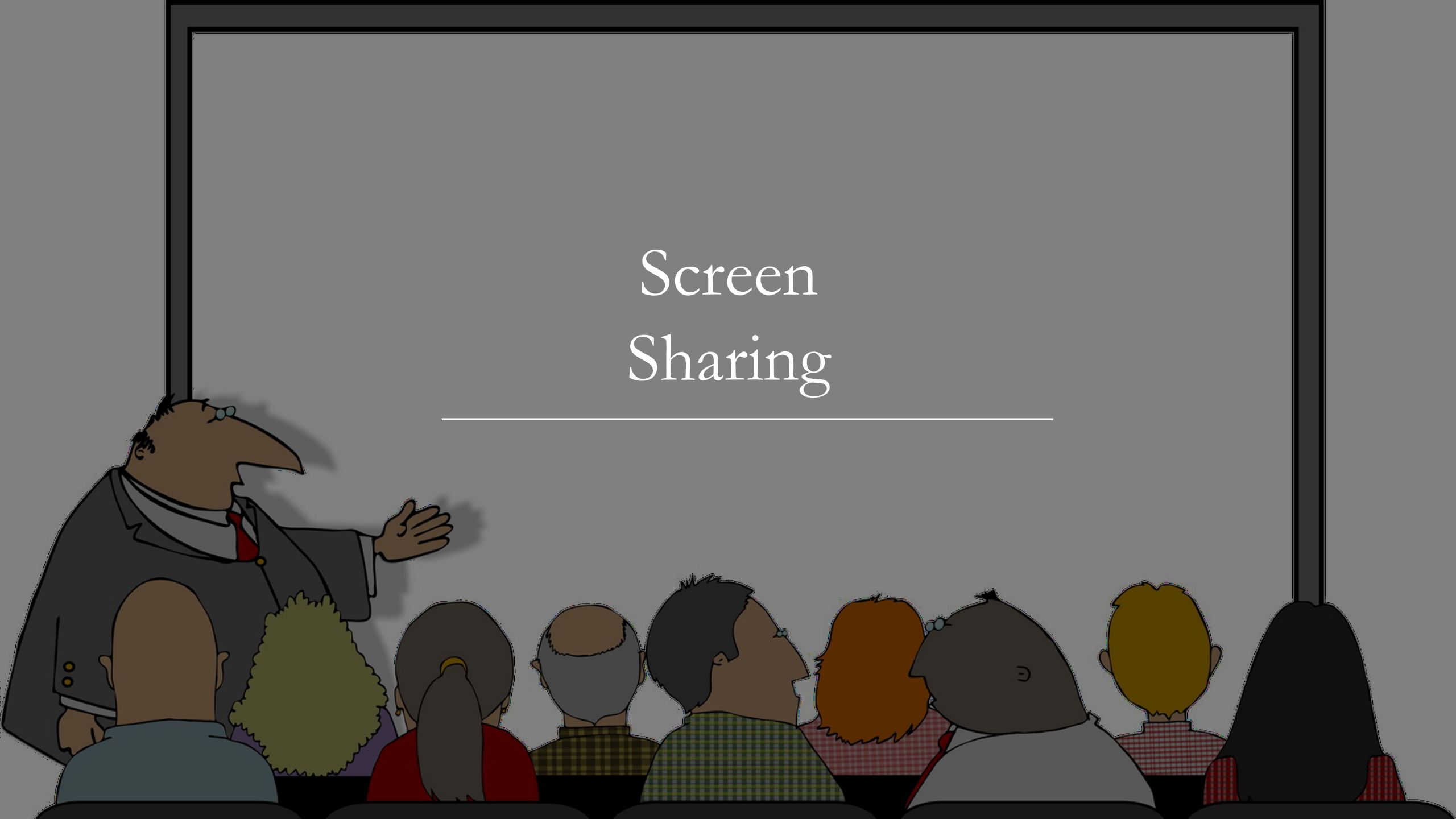
March 2021

- Saketh – continuing frontend – implementing
  - Setback: Spent too much time trying to hand-draw all music elements
    - Replaced with successful VexFlow/EasyScore implementation
  - Developed functions that use JSONified SMML to display sheet music
- Evan – solidifying + packaging backend
  - Packaged disconnected Python scripts into a unified, documented package
    - Provides a variety of functions dealing both with SMML + audio
    - Much easier to implement, and to test comprehensively
  - Took over "app" (backend) part of ElectronJS — implementing Python and communicating with HTML frontend

Advisor Meetings

- Already met with out-of-school mentor: Discussed March setbacks and how to move forward
  - Explained setbacks and demonstrated EasyScore
  - Discussed how to prioritize when "salvaging"
  - Helpful review of our current situation, helping us not feel weighed down, but to move on and be productive
  - Discussed pieces to use for demo pieces
- Plan to meet with in-school mentor Thursday after school (had to be postponed)

Screen Sharing



Next Steps

- Frontend: Refinement and debugging in the app itself
  - Aesthetic improvements? Sheet music display enhancements?
- Backend: Testing — Lots and lots of testing
  - Got pushed back from March due to setback: Variety of tests
  - Testing framerate + sample size to see if pitch accuracy improves
  - Testing pieces of various complexity, to see if tracking can improve
  - Also: Want to compare computer-generated audio with live audio, as a control
- Develop the SMML samples and videos for the final presentation
  - Variety of pieces ranging in complexity, suggested by out-of-school mentor
  - Will need to transcribe to SMML (not too difficult) and demonstrate its usage (more difficult)